# pyLM

v1.1

Generated by Doxygen 1.8.6

Wed Nov 9 2016 10:47:46

# Contents

# Chapter 1

# Description

pyLM is a Problem Solving Environment (PSE) for biological simulations [1]. Written in Python, it wraps and extends the highly optimized multi-GPU Lattice Microbes stochastic simulation software [2,3,4]. The PSE is comprised of a base set of functionality to set up, monitor and modify simulations, as well as a set of standard post-processing routines that interface to other Python packages, including NumPy, SciPy, H5py, iGraph to name a few. See [1] for additional information as well as the user guide on the main website. If you use pyLM in your simulations, please cite references [1] and [3] below.

#### References


1. J.R. Peterson, M.J. Hallock, J.A. Cole and Z. Luthey-Schulten. A Problem Solving Environment for Stochastic Biological Simulations. *PyHPC 2013 at Supercomputing 2013*, 2013.

2. E. Roberts, J.E. Stone, L. Sepulveda, W.W. Hwu, and Z. Luthey-Schulten. Long time-scale simulations of in vivo diffusion using GPU hardware. In *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*, 2009.

3. E. Roberts, J. E. Stone, and Z. Luthey-Schulten. Lattice microbes: high-performance stochastic simulation method for the reaction-diffusion master equation. *J. Comp. Chem.*, 32(3), 245-55, 2013.

4. M.J. Hallock, J.E. Stone, E. Roberts, C. Fry and Z. Luthey-Schulten. Simulation of reaction diffusion processes over biologically-relevant size and time scales using multi-GPU workstations *Parallel Comput.* 40:86-99, 2014, doi:10.1016/j.parco.2014.03.009.

# Chapter 2

# Namespace Index

## 2.1 Packages

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

# 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 peekFile Namespace Reference

### Variables

- tuple parser = argparse.ArgumentParser(description="Print info on simulation file")
- tuple args = parser.parse_args()
- tuple tf = float(hdf['Parameters'].attrs['maxTime'])
- list t = sim['LatticeTimes']
- tuple n = len(sim['Lattice'])

### 6.1.1 Variable Documentation

#### 6.1.1.1 tuple peekFile.args = parser.parse_args()

#### 6.1.1.2 tuple peekFile.n = len(sim['Lattice'])

#### 6.1.1.3 tuple peekFile.parser = argparse.ArgumentParser(description="Print info on simulation file")

#### 6.1.1.4 list peekFile.t = sim['LatticeTimes']

#### 6.1.1.5 tuple peekFile.tf = float(hdf['Parameters'].attrs['maxTime'])

## 6.2 pyLM Namespace Reference

The problem solving environement which interfaces to Lattice Microbes simulation software.

### Namespaces

- CME
- ipyInterface
- LMLogger
- RDME
- units

### Variables

- list __all__ = ['LMLogger', 'CME', 'RDME', 'units', 'ipyInterface']

### 6.2.1 Detailed Description

The problem solving environement which interfaces to Lattice Microbes simulation software.

### 6.2.2 Variable Documentation

#### 6.2.2.1 list pyLM.__all__ = ['LMLogger', 'CME', 'RDME', 'units', 'ipyInterface']

## 6.3 pyLM.CME Namespace Reference

### Classes

- class CMESimulation

  *A CME simulation that contains reactions and species.*

### Functions

- def tqdm

### 6.3.1 Function Documentation

#### 6.3.1.1 def pyLM.CME.tqdm ( *x*, *ascii* = `False` )

## 6.4 pyLM.ipyInterface Namespace Reference

### Functions

- def getReactionString

  *Write a reaction to a string with the correct units.*
- def writeTable

  *Write an HTML formatted table to a string.*
- def displayRDMESetup

  *Display an RDME initial conditions in a Jupyter Notebook using ipywidgets.*

### 6.4.1 Function Documentation

#### 6.4.1.1 def pyLM.ipyInterface.displayRDMESetup ( *filename*, *continuous_update* = `True` )

Display an RDME initial conditions in a Jupyter Notebook using ipywidgets.

**Parameters**

| | |
|---|---|
| *filename* | The name of a LM file encoding an RDME simulation |
| *continuous_update* | Whether to update the view continuously or only during; default: True |

#### 6.4.1.2 def pyLM.ipyInterface.getReactionString ( *rct*, *prd*, *rate* )

Write a reaction to a string with the correct units.

**Parameters**

| | |
|---|---|
| *rct* | Reactants tuple or string |
| *prd* | Products tuple or string |
| *rate* | The rate constant |

**Returns**

Three strings: 1) Reaction, 2) rate constant, and 3) units of the rate constant

#### 6.4.1.3 def pyLM.ipyInterface.writeTable ( *columnNames, rows* )

Write an HTML formatted table to a string.

**Parameters**

| | |
|---|---|
| *columnNames* | The name of the columns for the table |
| *rows* | A list of tuples each of the same length as columnNames |

## 6.5 pyLM.LMLogger Namespace Reference

### Classes

- class NullHandler

### Functions

- def setLMLoggerLevel

  *Set the level of the logger for the application.*
- def setLMLogFile

  *Set up file handler to print log to file.*
- def setLMLogConsole

  *Set the logger to write to the console as the code is working.*

### Variables

- tuple LMLogger = logging.getLogger('LMLogger')
- tuple nullHandlerLM = logging.NullHandler()
- tuple LMformatter = logging.Formatter('%(asctime)s: %(levelname)s: %(message)s')

### 6.5.1 Function Documentation

#### 6.5.1.1 def pyLM.LMLogger.setLMLogConsole ( *level* = `logging.DEBUG` )

Set the logger to write to the console as the code is working.

**Parameters**

| | |
|---|---|
| *level* | The level of information to log |

#### 6.5.1.2 def pyLM.LMLogger.setLMLogFile ( *filename*, *level* = `logging.DEBUG` )

Set up file handler to print log to file.

**Parameters**

| | |
|---|---|
| *filename* | The name of the file to log information |
| *level* | The level of information to log |

#### 6.5.1.3 def pyLM.LMLogger.setLMLoggerLevel ( *level* )

Set the level of the logger for the application.

**Parameters**

| | |
|---|---|
| *level* | The level the logger should report (e.g. logger.WARNING, logger.INFO, etc.) |

### 6.5.2 Variable Documentation

#### 6.5.2.1 tuple pyLM.LMLogger.LMformatter = logging.Formatter('%(asctime)s: %(levelname)s: %(message)s')

#### 6.5.2.2 tuple pyLM.LMLogger.LMLogger = logging.getLogger('LMLogger')

#### 6.5.2.3 tuple pyLM.LMLogger.nullHandlerLM = logging.NullHandler()

## 6.6 pyLM.RDME Namespace Reference

### Classes

- class CMESimulation

  *A CME simulation that contains reactions and species.*
- class RDMERegion

  *A class that represents a type region of an RDME simulation.*
- class RDMESimulation

  *A class that contains all regions, reactions, diffusions and rules for a RDME simulation.*

### Functions

- def tqdm

### 6.6.1 Function Documentation

#### 6.6.1.1 def pyLM.RDME.tqdm ( *x*, *ascii* = `False` )

## 6.7 pyLM.units Namespace Reference

### Functions

- def angstrom

*Length Wrapper Functions #.*

- def nm

  *Returns a representation of a number in nanometers.*
- def micron

  *Returns a representation of a number in micrometers.*
- def mm

  *Returns a representation of a number in millimeter.*
- def cm

  *Returns a representation of a number in centimeter.*
- def ns

  *Time Wrapper Functions #.*
- def microsecond

  *Returns a representation of a number in microsecond.*
- def ms

  *Returns a representation of a number in millisecond.*
- def second

  *Returns seconds in seconds.*
- def minute

  *Returns a representation of a number in minutes.*
- def hr

  *Returns a representation of a number in hours.*
- def day

  *Returns a representation of a number in days.*

### 6.7.1 Function Documentation

#### 6.7.1.1 def pyLM.units.angstrom ( *qty* )

Length Wrapper Functions #.

Returns a representation of a number in angstroms

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.2 def pyLM.units.cm ( *qty* )

Returns a representation of a number in centimeter.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.3 def pyLM.units.day ( *qty* )

Returns a representation of a number in days.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.4 def pyLM.units.hr ( *qty* )

Returns a representation of a number in hours.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.5 def pyLM.units.micron ( *qty* )

Returns a representation of a number in micrometers.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.6 def pyLM.units.microsecond ( *qty* )

Returns a representation of a number in microsecond.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.7 def pyLM.units.minute ( *qty* )

Returns a representation of a number in minutes.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.8 def pyLM.units.mm ( *qty* )

Returns a representation of a number in millimeter.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.9 def pyLM.units.ms ( *qty* )

Returns a representation of a number in millisecond.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.10 def pyLM.units.nm ( *qty* )

Returns a representation of a number in nanometers.

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.11 def pyLM.units.ns ( *qty* )

Time Wrapper Functions #.

Returns a representation of a number in nanosecond

**Parameters**

| | |
|---|---|
| *qty* | A list or singleton of a number |

#### 6.7.1.12 def pyLM.units.second ( *qty* )

Returns seconds in seconds.

Seems silly, but for completeness and ability to annotate the unit in code.

## 6.8 pySTDLM Namespace Reference

A standard library of functionality that is commonly used in biological simulations.

### Namespaces

- CellArranger
- Cells
- Distributions
- Fitting
- NetworkVisualization
- OnTheFly
- PostProcessing
- SBMLReader
- StandardCells
- StandardReactions
- StandardReactionSystems

### Variables

- list __all__ = ['StandardReactions', 'StandardReactionSystems', 'StandardCells', 'Cells', 'CellArranger', 'Distributions', 'PostProcessing', 'NetworkVisualization']

### 6.8.1 Detailed Description

A standard library of functionality that is commonly used in biological simulations. Three sets of functionality exist including "StandardReactions" which include standard reactions such as Michealis-Menton, membrane transport, transporters, etc. "StandardReactionSystems" includes published reaction systems from a variety of cells including the MinDE system, Lac switch systems in E. coli etc. Finally, "StandardCells" includes a number of standard cell geometries.

### 6.8.2 Variable Documentation

#### 6.8.2.1 list pySTDLM.__all__ = ['StandardReactions', 'StandardReactionSystems', 'StandardCells', 'Cells', 'CellArranger', 'Distributions', 'PostProcessing', 'NetworkVisualization']

## 6.9 pySTDLM.CellArranger Namespace Reference

### Classes

- class CellArranger

  *A base class for laying out cells in a 3D space.*
- class RandomArranger

  *An arranger that randomly orients a number of cells in a regin until either a count or a volume fraction target is met.*
- class RandomSphericalArranger

  *An arranger that randomly orients a number of cells in a sphere inside a regin until either a count or a volume fraction target is met.*
- class TightGridArranger

  *An arranger that orients cells in square grid.*
- class LooseGridArranger

  *An arranger that orients cells in square grid with padding in between.*

## 6.10 pySTDLM.Cells Namespace Reference

### Classes

- class CellShape

  *Base class for a particular cell type for use in CellArranger (and later more)*
- class SphereCell

  *A representation for a spherical cell.*
- class BoxCell

  *A representation for a spherical cell.*
- class CapsuleCell

  *A representation for a spherical cell.*
- class CapsuleShellCell

  *A representation for a spherical cell.*

## 6.11 pySTDLM.Distributions Namespace Reference

### Classes

- class ParticleGenerator

  *A particle generator that adds particles to a particular region of a simulation.*

## 6.12 pySTDLM.Fitting Namespace Reference

### Classes

- class Comparator
- class MomentComparator

- class TraceComparator
- class Fitter
- class SweepFitter
- class SimulatedAnnealingFitter

### Functions

- def DistributionComparator

### Variables

- counts
- binwidth
- measure

### 6.12.1 Function Documentation

#### 6.12.1.1 def pySTDLM.Fitting.DistributionComparator ( *Comparator* )

### 6.12.2 Variable Documentation

#### 6.12.2.1 pySTDLM.Fitting.binwidth

#### 6.12.2.2 pySTDLM.Fitting.counts

#### 6.12.2.3 pySTDLM.Fitting.measure

## 6.13 pySTDLM.NetworkVisualization Namespace Reference

### Functions

- def plotReactionNetwork

  *Plot the reaction scheme as a network.*
- def plotCMEReactionNetwork

  *Plot the reaction scheme as a network.*
- def plotRDMEReactionNetwork

  *Plot the reaction scheme as a network.*
- def plotCMEDynamicReactionNetwork

  *Plot the dynamics of the species on the network into a dynamic graph file (extension: .gexf)*

### Variables

- hasGexf = True

### 6.13.1 Function Documentation

#### 6.13.1.1 def pySTDLM.NetworkVisualization.plotCMEDynamicReactionNetwork ( *sim, filename, outfile, stride* = `1`, *showMax* = `False`, *showMin* = `False`, *threshold* = `-1`, *replicate* = `1` )

Plot the dynamics of the species on the network into a dynamic graph file (extension: .gexf)

**Parameters**

| | |
|---|---|
| *sim* | A CMESimulation object |
| *filename* | An file from which to read |
| *outfile* | A filename for which to output without the extension (.gexf) |
| *stride* | Stride through the times (Default=1) |
| *showMax* | Show the maximum value achieved over the timecourse on the node (Default=False) |
| *showMin* | Show the maximum value achieved over the timecourse on the node (Default=False) |
| *threshold* | Only show nodes that attain this value for at least one timepoint of the simulation. Set -1 to show all nodes (Default=-1) |
| *replicate* | The replicate to show (Default=1) |

#### 6.13.1.2 def pySTDLM.NetworkVisualization.plotCMEReactionNetwork ( *sim, filename, withRxnNodes =* `False`, *collapseReversible =* `False` )

Plot the reaction scheme as a network.

**Parameters**

| | |
|---|---|
| *sim* | An CMESimulation object |
| *filename* | A file to which to output |
| *withRxnNodes* | Plot the graph with reactions as nodes (default: false) |
| *collapse-Reversible* | Collapsee reversible reactions into one node (default: false) |

#### 6.13.1.3 def pySTDLM.NetworkVisualization.plotRDMEReactionNetwork ( *sim, filename, collapseReversible =* `False` )

Plot the reaction scheme as a network.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation object |
| *filename* | A file to which to output |
| *collapse-Reversible* | Collapsee reversible reactions into one node |

#### 6.13.1.4 def pySTDLM.NetworkVisualization.plotReactionNetwork ( *sim, filename, collapseReversible =* `False` )

Plot the reaction scheme as a network.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation or CMESimulation object |
| *filename* | A file to which to output |
| *collapse-Reversible* | Collapsee reversible reactions into one node |

### 6.13.2 Variable Documentation

#### 6.13.2.1 pySTDLM.NetworkVisualization.hasGexf = True

## 6.14 pySTDLM.OnTheFly Namespace Reference

## 6.15 pySTDLM.PostProcessing Namespace Reference

### Functions

- def openLMFile

  *Open a Lattice Microbes File for reading.*
- def closeLMFile

  *Close a Lattice Microbes File.*
- def showTraceFromFile

  *Show species trace from a particular replicate.*
- def plotTraceFromFile

  *Plot species from an output file.*
- def showTrace

  *Show a specific species trace.*
- def plotTrace

  *Plot a specific species trace.*
- def showAvgVarFromFile

  *Show species from an output file.*
- def plotAvgVarFromFile

  *Plot species from an output file.*
- def showAvgVar

  *Show a specific species average over time and variance.*
- def plotAvgVar

  *Plot a specific species average over time and variance.*
- def getOccupancyKymograph

  *Compute the specie density(occupancy) among a slice of the simulation domain as a function over time for the given direciton.*
- def plotOccupancyKymograph

  *Plot a specific specie density(occupancy) among a slice of the simulation domain as a function of a direction over time.*
- def showOccupancyKymograph

  *Show a specific specie density(occupancy) among a slice of the simulation domain as a function of a direction over time.*
- def plotPhaseSpace

  *Plot the 2D or 3D phase space associated with the given species over the replicates indicated.*
- def getTimesteps

  *Extract the timestep times.*
- def getSpecieTrace

  *Extract data for a particular specie for the specified replicate.*
- def getAvgVarTrace

  *Get the average and variance of the specie trace over time.*
- def getHistogram

  *Get the histogram for a specie.*
- def getPhaseSpace

  *Get the nD phase space associated with the traces of species.*

### 6.15.1 Function Documentation

#### 6.15.1.1 def pySTDLM.PostProcessing.closeLMFile ( *f* )

Close a Lattice Microbes File.

**Parameters**

| | |
|---|---|
| *f* | A previously opened lattice microbes file |

#### 6.15.1.2 def pySTDLM.PostProcessing.getAvgVarTrace ( *f, specie, doublingTime* = `None` )

Get the average and variance of the specie trace over time.

**Parameters**

| | |
|---|---|
| *f* | The HDF5 file handle to extract from or the name of a file to open |
| *specie* | The specie to extract |
| *doublingTime* | An optional doubling time parameter that will normalize average each trace by time in the cell cycle assuming exponentially growing cell (2log2 $*$ x/2^(t/DT)), effectively normalizing against cell size. Default: no averaging is performed |

**Returns**

avg, var, time

#### 6.15.1.3 def pySTDLM.PostProcessing.getHistogram ( *f, species* )

Get the histogram for a specie.

**Parameters**

| | |
|---|---|
| *f* | The HDF5 file handle to extract from or the name of a file to open |
| *specie* | An array of specie/s to extract. returns a nD histogram of the data. |

**Returns**

bins, edges (len(bins)+1)

#### 6.15.1.4 def pySTDLM.PostProcessing.getOccupancyKymograph ( *f, species* = `None`, *replicate* = `1` )

Compute the specie density(occupancy) among a slice of the simulation domain as a function over time for the given direciton.

**Parameters**

| | |
|---|---|
| *filename* | Name of file to extract data from |
| *specie* | A particular specie to plot density for |
| *replicate* | The replicate to show trace for |

#### 6.15.1.5 def pySTDLM.PostProcessing.getPhaseSpace ( *f, species, replicate* )

Get the nD phase space associated with the traces of species.

If a single replicate is specified, a single trace will be returned, otherwise a 2/3D density matrix will be returned.

**Parameters**

| | |
|---|---|
| *f* | The HDF5 file handle to extract from or the name of a file to open |
| *specie* | An iterable of 2 more specie names |
| *replicate* | The replicate for which to extract the phase space |

**Returns**

A numpy array of a trace from a single replicate (specie1, specie2, ...)

#### 6.15.1.6 def pySTDLM.PostProcessing.getSpecieTrace ( *f, specie, replicate* = `1`, *doublingTime* = `None` )

Extract data for a particular specie for the specified replicate.

**Parameters**

| | |
|---|---|
| *f* | The HDF5 file handle to extract from or the name of a file to open |
| *specie* | The specie to extract data |
| *replicate* | The number of the replicate to extract from (default: 1) |
| *doublingTime* | An optional doubling time parameter that will normalize average each trace by time in the cell cycle assuming exponentially growing cell (2log2 $*$ x/2^(t/DT)), effectively normalizing against cell size. Default: no averaging is performed |

**Returns**

The species time trace in a numpy array

#### 6.15.1.7 def pySTDLM.PostProcessing.getTimesteps ( *f* )

Extract the timestep times.

**Parameters**

| | |
|---|---|
| *f* | The HDF5 file handle to extract from or the name of a file to open |

**Returns**

The timestep times in a numpy array

#### 6.15.1.8 def pySTDLM.PostProcessing.openLMFile ( *filename* )

Open a Lattice Microbes File for reading.

**Parameters**

| | |
|---|---|
| *filename* | Name of the file |

**Returns**

a handle to the file

#### 6.15.1.9 def pySTDLM.PostProcessing.plotAvgVar ( *f, species* = `None`, *filename* = `None`, *kwargs* )

Plot a specific species average over time and variance.

**Parameters**

| | |
|---|---|
| *f* | An h5py object handle |
| *species* | A list of species to show |
| *filename* | A filename to print to (default None, gives the same behavior as showAvgVar(...)) |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |

**Returns**

A handle to the figure object created which allows customization of plot attributes

#### 6.15.1.10 def pySTDLM.PostProcessing.plotAvgVarFromFile ( *filename, species, outfile, kwargs* )

Plot species from an output file.

**Parameters**

| | |
|---|---|
| *filename* | The name of an HDF5 output file generated by LatticeMicrobes |
| *species* | A list of species to plot |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |
| *outfile* | A filename to plot to |

#### 6.15.1.11 def pySTDLM.PostProcessing.plotOccupancyKymograph ( *f, species* = `None`, *replicate* = `1`, *filename* = `None` )

Plot a specific specie density(occupancy) among a slice of the simulation domain as a function of a direction over time.

**Parameters**

| | |
|---|---|
| *f* | The name of the LM file |
| *specie* | A particular specie to plot density for |
| *filename* | A filename to print to (default None, gives the same behavior as showAvgVar(...)) |
| *replicate* | The replicate to show trace for |

#### 6.15.1.12 def pySTDLM.PostProcessing.plotPhaseSpace ( *f, species* = `None`, *replicate* = `1`, *withHistogram* = `False` )

Plot the 2D or 3D phase space associated with the given species over the replicates indicated.

**Parameters**

| | |
|---|---|
| *f* | An h5py object handle |
| *specie* | An iterable of 2 or 3 specie names indicating whether to plot in 2D or 3D space |
| *replicate* | The replicate to show trace for |
| *withHistogram* | If set to true, a heatmap of the phase space over all replicates will be plotted in the background. (NOTE: this only works in 2D.) |

#### 6.15.1.13 def pySTDLM.PostProcessing.plotTrace ( *f, species* = `None`, *replicate* = `1`, *filename* = `None`, *kwargs* )

Plot a specific species trace.

**Parameters**

| | |
|---|---|
| *f* | An h5py object handle |
| *species* | A specie name or a list of species to show; can be a single string or a iterable list of species |
| *filename* | A filename to print to (default None, gives the same behavior as showAvgVar(...)) |
| *replicate* | The replicate to show trace for. Can be an integer or an iterable list of replicte numbers |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |

**Returns**

A handle to the figure object created which allows customization of plot attributes

#### 6.15.1.14 def pySTDLM.PostProcessing.plotTraceFromFile ( *filename, species, replicate, outfile, kwargs* )

Plot species from an output file.

**Parameters**

| | |
|---|---|
| *filename* | The patch to an HDF5 output file generated by LatticeMicrobes |
| *species* | A list of species to plot |
| *replicate* | The replicate to show trace for |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |
| *outfile* | A filename to plot to |

#### 6.15.1.15 def pySTDLM.PostProcessing.showAvgVar ( *f, species, kwargs* )

Show a specific species average over time and variance.

**Parameters**

| | |
|---|---|
| *f* | An h5py object handle |
| *species* | A list of species to show |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |

#### 6.15.1.16 def pySTDLM.PostProcessing.showAvgVarFromFile ( *filename, species, kwargs* )

Show species from an output file.

**Parameters**

| | |
|---|---|
| *filename* | The name of an HDF5 output file generated by LatticeMicrobes |
| *species* | A list of species to plot |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |

#### 6.15.1.17 def pySTDLM.PostProcessing.showOccupancyKymograph ( *f, species* = `None`, *replicate* = `1` )

Show a specific specie density(occupancy) among a slice of the simulation domain as a function of a direction over time.

**Parameters**

| | |
|---|---|
| *f* | An h5py object handle |
| *specie* | A particular specie to plot density for |
| *replicate* | The replicate to show trace for |

#### 6.15.1.18 def pySTDLM.PostProcessing.showTrace ( *f, species, replicate, kwargs* )

Show a specific species trace.

**Parameters**

| | |
|---|---|
| *f* | An h5py object handle |
| *species* | A list of species to show |
| *replicate* | The replicate to show trace for |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |

#### 6.15.1.19 def pySTDLM.PostProcessing.showTraceFromFile ( *filename, species, replicate, kwargs* )

Show species trace from a particular replicate.

**Parameters**

| | |
|---|---|
| *filename* | The patch to an HDF5 output file generated by LatticeMicrobes |
| *species* | A list of species to plot |
| *replicate* | The replicate to show trace for |
| *kwargs* | Additional arguments to be passed on to matplotlib.plot. These are any arguments that are valid with matplotlib.plot |

## 6.16 pySTDLM.SBMLReader Namespace Reference

### Functions

- def importSBMLModel

  *Helper Functions ##.*
- def importSBMLModelL3V1
- def getRate
- def readSBMLtoCME

  *Read an SBML file for the reaction model for a CME simulation.*
- def readSBMLtoRDME

  *Read an SBML file for the reaction model for a RDME simulation.*
- def readSBMLtoRegion

  *Read an SBML file for the reaction model into a specific region of the RDME simulation.*

### 6.16.1 Function Documentation

#### 6.16.1.1 def pySTDLM.SBMLReader.getRate ( *rxn, rcts, prds, sim, globalP, globalPV* )

#### 6.16.1.2 def pySTDLM.SBMLReader.importSBMLModel ( *filename* )

Helper Functions ##.

#### 6.16.1.3 def pySTDLM.SBMLReader.importSBMLModelL3V1 ( *model, sim, region* = None )

#### 6.16.1.4 def pySTDLM.SBMLReader.readSBMLtoCME ( *sim, filename* )

Read an SBML file for the reaction model for a CME simulation.

**Parameters**

| | |
|---|---|
| *sim* | A CMESimulation object |
| *filename* | A SBML filename/filepath |

**Returns**

The simulation object

#### 6.16.1.5 def pySTDLM.SBMLReader.readSBMLtoRDME ( *sim, filename* )

Read an SBML file for the reaction model for a RDME simulation.

**Parameters**

| | |
|---|---|
| *sim* | A RDMESimulation object |
| *filename* | A SBML filename/filepath |

**Returns**

The simulation object

#### 6.16.1.6 def pySTDLM.SBMLReader.readSBMLtoRegion ( *sim, region, filename* )

Read an SBML file for the reaction model into a specific region of the RDME simulation.

**Parameters**

| | |
|---|---|
| *sim* | A RDMESimulation object |
| *region* | A region name that already exists in the RDMESimulation |
| *filename* | A SBML filename/filepath |

**Returns**

The simulation object

## 6.17 pySTDLM.StandardCells Namespace Reference

### Functions

- def buildEColiCell

  *Build a standard E coli cell 0.5 microns across and 2 microns long.*
- def buildDividingEColiCell

  *Build a standard E coli cell 0.5 microns across and 4 microns long.*
- def buildSphericalEColiCell

  *Build a Spherical E coli cell, one that lacks the gene for elongation.*
- def buildFilamentousEColiCell

  *Build a long filamentous E coli cell where the user can specify the length.*
- def packFastGrowingEcoli

*Pack a cell with the protein distribution for fast growing E coli.*

- def packSlowGrowingEcoli

  *Pack a cell with the protein distribution for fast growing E coli.*

### 6.17.1 Function Documentation

#### 6.17.1.1 def pySTDLM.StandardCells.buildDividingEColiCell ( *sim*, *crowded* = `False`, *crowdedMembrane* = `False` )

Build a standard E coli cell 0.5 microns across and 4 microns long.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation object |
| *crowded* | Should the cytosol be crowded (default: false) |
| *crowded-Membrane* | Should the membrane be crowded (default: false) |

#### 6.17.1.2 def pySTDLM.StandardCells.buildEColiCell ( *sim*, *crowded* = `False`, *crowdedMembrane* = `False` )

Build a standard E coli cell 0.5 microns across and 2 microns long.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation object |
| *crowded* | Should the cytosol be crowded (default: false) |
| *crowded-Membrane* | Should the membrane be crowded (default: false) |

#### 6.17.1.3 def pySTDLM.StandardCells.buildFilamentousEColiCell ( *sim*, *length* = `8.0`, *crowded* = `False`, *crowdedMembrane* = `False` )

Build a long filamentous E coli cell where the user can specify the length.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation object |
| *length* | The length of the cell in microns (default: 8) |
| *crowded* | Should the cytosol be crowded (default: false) |
| *crowded-Membrane* | Should the membrane be crowded (default: false) |

#### 6.17.1.4 def pySTDLM.StandardCells.buildSphericalEColiCell ( *crowded* = `False`, *latticeSpacing* = `16`, *crowdedMembrane* = `False` )

Build a Spherical E coli cell, one that lacks the gene for elongation.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation object |
| *crowded* | Should the cytosol be crowded (default: false) |
| *crowded-Membrane* | Should the membrane be crowded (default: false) |

#### 6.17.1.5 def pySTDLM.StandardCells.packFastGrowingEcoli ( *sim* )

Pack a cell with the protein distribution for fast growing E coli.

**Parameters**

| | |
|---|---|
| *sim* | A RDMESimulation object |

#### 6.17.1.6 def pySTDLM.StandardCells.packSlowGrowingEcoli ( *sim* )

Pack a cell with the protein distribution for fast growing E coli.

**Parameters**

| | |
|---|---|
| *sim* | A RDMESimulation object |

## 6.18 pySTDLM.StandardReactions Namespace Reference

### Functions

- def addMembraneTransporter

  *Adds a membrane transport system to an RDME system.*
- def addPassiveTransport

  *Adds a passive transport system to an RDME system.*
- def addMichaelisMenten

  *Adds a Michaelis Menten Reaction.*
- def addReversibleMichaelisMenten

  *Add a Reversible Michaelis Menten Reaction.*
- def createExpressionModel

  *Create a set of gene/mRNA/protein reactions based on a genebank file and a set of rates assuming constitutive expression.*

### Variables

- hasBio = True

### 6.18.1 Function Documentation

#### 6.18.1.1 def pySTDLM.StandardReactions.addMembraneTransporter ( *sim, transporter, number, name1, name2, dRate, kf, kr, region1 =* `'cytoplasm'`, *region2 =* `'default'`, *intoRegion =* `'membrane'` )

Adds a membrane transport system to an RDME system.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation to add the transporter system to |
| *transporter* | Name of the transporter particle |
| *number* | The number of transporter molecules |
| *name1* | Name of the transported particle in region1 |
| *name2* | Name of the transported particle in region2 |
| *dRate* | The rate of diffusion from region1<->intoRegion and region2<->intoRegion |
| *kf* | The forward reaction rate |

| | |
|---|---|
| *kr* | The reverse reaction rate |
| *region1* | The region on the first side of the membrane |
| *region2* | The region on the second side of the membrane |
| *intoRegion* | The region representing the membrane |

**Returns**

The simulation object so this can be a chained call

#### 6.18.1.2 def pySTDLM.StandardReactions.addMichaelisMenten ( *sim, reactant, enzyme, product, k1f, k1b, k2, region =* `'cytoplasm'` )

Adds a Michaelis Menten Reaction.

**Parameters**

| | |
|---|---|
| *sim* | The RDME or CME reaction |
| *reactant* | The reactant that reacts with the enzyme |
| *enzyme* | The enzyme catalyzing the reaction |
| *product* | The product of the reaction |
| *k1f* | The forward reaction rate |
| *k1b* | The backward reaction rate |
| *k2* | The second forward rate |
| *region* | The region in which the reaction should occur (RDME only) |

**Returns**

The simulation object so this can be a chained call

#### 6.18.1.3 def pySTDLM.StandardReactions.addPassiveTransport ( *sim, specie, dRate, region1 =* `'cytoplasm'`, *region2 =* `'default'`, *region3 =* `'membrane'` )

Adds a passive transport system to an RDME system.

**Parameters**

| | |
|---|---|
| *sim* | An RDMESimulation to add the passive transport to |
| *specie* | The species that can diffuse across the membrane |
| *dRate* | The rate of diffusion across the membrane |
| *region1* | The region on the first side of the membrane |
| *region2* | The region on the second side of the membrane |
| *region3* | The region representing the membrane |

**Returns**

The simulation object so this can be a chained call

#### 6.18.1.4 def pySTDLM.StandardReactions.addReversibleMichaelisMenten ( *sim, reactant, enzyme, product, k1f, k1b, k2f, k2b, k3f, k3b, region =* `'cytoplasm'` )

Add a Reversible Michaelis Menten Reaction.

**Parameters**

| | |
|---|---|
| *sim* | The RDME or CME reaction |
| *reactant* | The reactant that reacts with the enzyme |
| *enzyme* | The enzyme catalyzing the reaction |
| *product* | The product of the reaction |
| *k1f* | The forward reaction rate |
| *k1b* | The backward reaction rate |
| *k2f* | The second forward rate |
| *k2b* | the second backward rate |
| *k3f* | The third forward rate |
| *k3b* | the third backward rate |
| *region* | the region in which the reaction should occur (RDME only) |

**Returns**

The simulation object so this can be a chained call

#### 6.18.1.5 def pySTDLM.StandardReactions.createExpressionModel ( *sim, gb, kt, kd, kr, kdil* = `None`, *regions* = `None` )

Create a set of gene/mRNA/protein reactions based on a genebank file and a set of rates assuming constitutive expression.

**Parameters**

| | |
|---|---|
| *sim* | The RDME or CME simulation |
| *gb* | The genbank filename. File should be readable by BioPython. |
| *kt* | mRNA transcription rate dictionary { locusTag -> rate } |
| *kd* | mRNA degradation rate dictionary { locusTag -> rate } |
| *kr* | Protein transcription rate dictionary {locusTag -> rate } |
| *kdil* | Protein dillution/degradation rate dictionary {locusTag -> rate} (Optional; Default "None", meaning no dillution reaction will be specified) |
| *regions* | Regions for the reactions to occur. Degradation is allowed in both regions. {locusTag -> (transcriptionRegion, translationRegion)} (required for RDMESimulations) |

**Returns**

SeqIO representation of the Genbank file with qualifiers added: qualifiers["dna_id"->(str,int),"rna_id"->(str,int), "protein_id"->(str,int)]

### 6.18.2 Variable Documentation

#### 6.18.2.1 pySTDLM.StandardReactions.hasBio = True

## 6.19 pySTDLM.StandardReactionSystems Namespace Reference

### Functions

- def addMinDESystem

  *Adds the MinDE system in Ecoli as a standard reaction network.*
- def addLacTwoStateSystem

  *Adds the Lac switch reaction with two states of the DNA.*
- def addPTSPathway

  *Adds the Phosphoenolpyruvate-depenedent phosphotransferase reaction system.*

### 6.19.1 Function Documentation

#### 6.19.1.1 def pySTDLM.StandardReactionSystems.addLacTwoStateSystem ( *sim, inducerType = 'TMG'* )

Adds the Lac switch reaction with two states of the DNA.

Reference: E. Roberts, A. Magis, J.O. Ortiz, W. Baumeister, Z. Luthey-Schulten. Noise Contributions in an Inducible Genetic Switch: A Whole-Cell Simulation Study. PLoS Comput. Bio. 7(3): 2011, e1002010.

**Parameters**

| | |
|---|---|
| *sim* | A CMESimulation or RDMESimulation object with the "cytoplasm" region defined |
| *sim* | inducerType The type of inducer used for the switch, either TMG or IPTG |

**Returns**

A pointer to the simulation object that was passed in

#### 6.19.1.2 def pySTDLM.StandardReactionSystems.addMinDESystem ( *sim* )

Adds the MinDE system in Ecoli as a standard reaction network.

Reference:

**Parameters**

| | |
|---|---|
| *sim* | A RDMESimulation object with the "cytoplasm" and "membrane" regions defined |

**Returns**

A pointer to the simulation object that was passed in

#### 6.19.1.3 def pySTDLM.StandardReactionSystems.addPTSPathway ( *sim* )

Adds the Phosphoenolpyruvate-depenedent phosphotransferase reaction system.

Reference: J.V. Rodriguez, J.A. Kaandorp, M. Dobrzynski, J.G. Blom Spatial stochastic modelling of the phosphoenolpyruvate-dependent phosphotransferase (PTS) pathway in Escherichia coli. Bioinform. 22:15 (2006), pp. 1895-1901.

**Parameters**

| | |
|---|---|
| *sim* | A RDMESimulation object with the "cytoplasm", "default" and "membrane" regions defined |

**Returns**

A pointer to the simulation object that was passed in

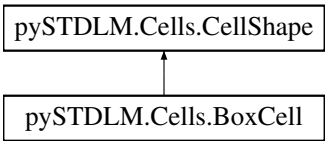
# Chapter 7

# Class Documentation

## 7.1 pySTDLM.Cells.BoxCell Class Reference

A representation for a spherical cell.

Inheritance diagram for pySTDLM.Cells.BoxCell:




### Public Member Functions

- def __init__
- def internalModelCell
- def computeVolume
- def internalAddToSim
- def internalTranslateCell

### Public Attributes

- name
- p1
- p2
- mins
- maxs
- width
- height
- depth
- membraneThickness

### 7.1.1 Detailed Description

A representation for a spherical cell.

This cell requires an attribute dictionary that includes "width", "height", "depth" and "membraneThickness"

### 7.1.2 Constructor & Destructor Documentation

#### 7.1.2.1 def pySTDLM.Cells.BoxCell.__init__ ( *self, p1 =* `3*[0.], p2 =` `3*[0.]` )

### 7.1.3 Member Function Documentation

#### 7.1.3.1 def pySTDLM.Cells.BoxCell.computeVolume ( *self* )

#### 7.1.3.2 def pySTDLM.Cells.BoxCell.internalAddToSim ( *self, sim* )

#### 7.1.3.3 def pySTDLM.Cells.BoxCell.internalModelCell ( *self, attr* )

#### 7.1.3.4 def pySTDLM.Cells.BoxCell.internalTranslateCell ( *self, point* )

### 7.1.4 Member Data Documentation

#### 7.1.4.1 pySTDLM.Cells.BoxCell.depth

#### 7.1.4.2 pySTDLM.Cells.BoxCell.height

#### 7.1.4.3 pySTDLM.Cells.BoxCell.maxs

#### 7.1.4.4 pySTDLM.Cells.BoxCell.membraneThickness

#### 7.1.4.5 pySTDLM.Cells.BoxCell.mins

#### 7.1.4.6 pySTDLM.Cells.BoxCell.name

#### 7.1.4.7 pySTDLM.Cells.BoxCell.p1

#### 7.1.4.8 pySTDLM.Cells.BoxCell.p2

#### 7.1.4.9 pySTDLM.Cells.BoxCell.width

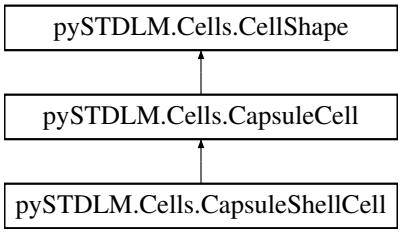
The documentation for this class was generated from the following file:

- Cells.py

## 7.2 pySTDLM.Cells.CapsuleCell Class Reference

A representation for a spherical cell.

Inheritance diagram for pySTDLM.Cells.CapsuleCell:



### Public Member Functions

- def __init__
- def internalModelCell
- def computeVolume
- def internalAddToSim
- def internalTranslateCell

### Public Attributes

- name
- p1
- p2
- radius
- length
- mins
- maxs
- membraneThickness

## 7.2.1 Detailed Description

A representation for a spherical cell.

This cell requires an attribute dictionary that includes "radius", "length", and "membraneThickness"

## 7.2.2 Constructor & Destructor Documentation

### 7.2.2.1 def pySTDLM.Cells.CapsuleCell.__init__ ( *self, p1* = `3*[0.]`, *p2* = `3*[0.]`, *radius* = `0.`, *length* = `0.` )

## 7.2.3 Member Function Documentation

### 7.2.3.1 def pySTDLM.Cells.CapsuleCell.computeVolume ( *self* )

### 7.2.3.2 def pySTDLM.Cells.CapsuleCell.internalAddToSim ( *self, sim* )

### 7.2.3.3 def pySTDLM.Cells.CapsuleCell.internalModelCell ( *self, attr* )

### 7.2.3.4 def pySTDLM.Cells.CapsuleCell.internalTranslateCell ( *self, point* )

## 7.2.4 Member Data Documentation

### 7.2.4.1 pySTDLM.Cells.CapsuleCell.length

### 7.2.4.2 pySTDLM.Cells.CapsuleCell.maxs

### 7.2.4.3 pySTDLM.Cells.CapsuleCell.membraneThickness

### 7.2.4.4 pySTDLM.Cells.CapsuleCell.mins

### 7.2.4.5 pySTDLM.Cells.CapsuleCell.name

### 7.2.4.6 pySTDLM.Cells.CapsuleCell.p1

### 7.2.4.7 pySTDLM.Cells.CapsuleCell.p2

#### 7.2.4.8 pySTDLM.Cells.CapsuleCell.radius

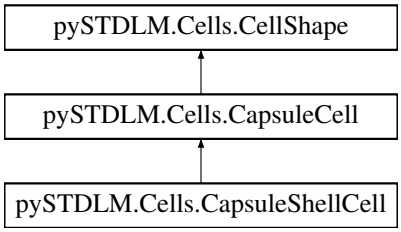
The documentation for this class was generated from the following file:

- Cells.py

## 7.3 pySTDLM.Cells.CapsuleShellCell Class Reference

A representation for a spherical cell.

Inheritance diagram for pySTDLM.Cells.CapsuleShellCell:




### Public Member Functions

- def __init__
- def internalModelCell
- def computeVolume
- def internalAddToSim

### Public Attributes

- name
- radius
- length
- p1
- p2
- membraneThickness
- mins
- maxs

### 7.3.1 Detailed Description

A representation for a spherical cell.

This cell requires an attribute dictionary that includes "radius", "length" and "membraneThickness"

### 7.3.2 Constructor & Destructor Documentation

#### 7.3.2.1 def pySTDLM.Cells.CapsuleShellCell.__init__ ( *self*, *p1* = `3*[0.]`, *p2* = `3*[0.]`, *radius* = `0.`, *length* = `0.` )

### 7.3.3 Member Function Documentation

#### 7.3.3.1 def pySTDLM.Cells.CapsuleShellCell.computeVolume ( *self* )

#### 7.3.3.2 def pySTDLM.Cells.CapsuleShellCell.internalAddToSim ( *self, sim* )

#### 7.3.3.3 def pySTDLM.Cells.CapsuleShellCell.internalModelCell ( *self, attr* )

### 7.3.4 Member Data Documentation

#### 7.3.4.1 pySTDLM.Cells.CapsuleShellCell.length

#### 7.3.4.2 pySTDLM.Cells.CapsuleShellCell.maxs

#### 7.3.4.3 pySTDLM.Cells.CapsuleShellCell.membraneThickness

#### 7.3.4.4 pySTDLM.Cells.CapsuleShellCell.mins

#### 7.3.4.5 pySTDLM.Cells.CapsuleShellCell.name

#### 7.3.4.6 pySTDLM.Cells.CapsuleShellCell.p1

#### 7.3.4.7 pySTDLM.Cells.CapsuleShellCell.p2

#### 7.3.4.8 pySTDLM.Cells.CapsuleShellCell.radius

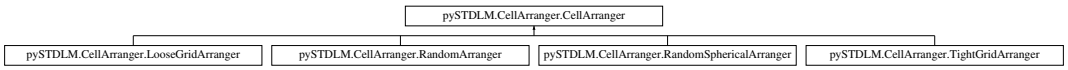
The documentation for this class was generated from the following file:

- Cells.py

## 7.4 pySTDLM.CellArranger.CellArranger Class Reference

A base class for laying out cells in a 3D space.

Inheritance diagram for pySTDLM.CellArranger.CellArranger:




### Public Member Functions

- def __init__

  *Create a CellArranger.*
- def cellFactory
- def packVolume

  *Pack a specified volume with cells.*
- def addToSimulation

  *Add packed volume to RDME simulation.*

### Public Attributes

- allowedCellTypes
- allowedForms
- cellType
- cellAttributes
- cells

- packerName
- arrangerAttributes

### 7.4.1 Detailed Description

A base class for laying out cells in a 3D space.

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 def pySTDLM.CellArranger.CellArranger.__init__ ( *self*, *cellType* = None, *cellAttributes* = None, *arrangerAttributes* = None, *name* = 'Unknown' )

Create a CellArranger.

**Parameters**

| | |
|---|---|
| *self* | |
| *cellType* | The type of cell this packer represents, can be one of: ["Sphere", "Capsule", "Box", "Capsule-Shell"] |
| *cellAttributes* | A dictionary with the required attributes specified for the cell and the form |
| *arranger-Attributes* | A dictionary with the required/optional requirements for the individiual arranger |

### 7.4.3 Member Function Documentation

#### 7.4.3.1 def pySTDLM.CellArranger.CellArranger.addToSimulation ( *self*, *sim* )

Add packed volume to RDME simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *sim* | An RDMESimulation object |

#### 7.4.3.2 def pySTDLM.CellArranger.CellArranger.cellFactory ( *self* )

#### 7.4.3.3 def pySTDLM.CellArranger.CellArranger.packVolume ( *self*, *sim*, *volume* = None )

Pack a specified volume with cells.

**Parameters**

| | |
|---|---|
| *self* | |
| *volume* | A volume specified as [[xmin,ymin,zmin],[xmax,ymax,zmax]] (TODO: Add angles) |
| *form* | The form of the packing, which can one of: ["Random", "RandomSpherical", "TightGrid", "LooseGrid", "SkewedGrid"] |

**Returns**

A tuple of the form [volumePacked, volumePercent, numberCells]

### 7.4.4 Member Data Documentation

#### 7.4.4.1 pySTDLM.CellArranger.CellArranger.allowedCellTypes

#### 7.4.4.2 pySTDLM.CellArranger.CellArranger.allowedForms

#### 7.4.4.3 pySTDLM.CellArranger.CellArranger.arrangerAttributes

#### 7.4.4.4 pySTDLM.CellArranger.CellArranger.cellAttributes

#### 7.4.4.5 pySTDLM.CellArranger.CellArranger.cells

#### 7.4.4.6 pySTDLM.CellArranger.CellArranger.cellType

#### 7.4.4.7 pySTDLM.CellArranger.CellArranger.packerName

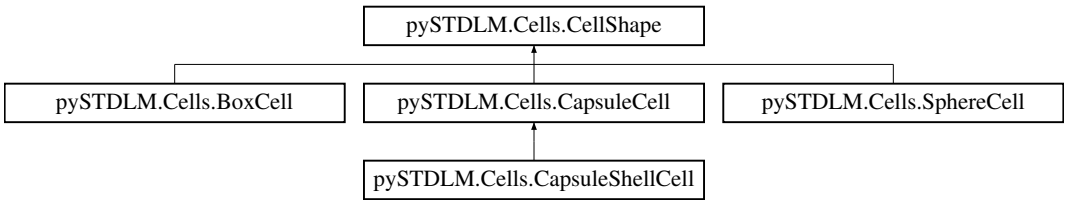
The documentation for this class was generated from the following file:

- CellArranger.py

## 7.5 pySTDLM.Cells.CellShape Class Reference

Base class for a particular cell type for use in CellArranger (and later more)

Inheritance diagram for pySTDLM.Cells.CellShape:




### Public Member Functions

- def __init__
- def getVolume

  *Volume occupied by cell NOTE: When overriding this class, you must specify a "computeVolume()" function of no arguments.*
- def boundingBox

  *Return a bounding box for the cell TODO: Make it return an orientation as well.*
- def addToSimulation

  *Add the cell to the simulation.*
- def translateCell

  *Shift a cell in space by the specified amount.*
- def createModelCell

  *Create model cell.*
- def setRegions

  *Set regions.*

### Public Attributes

- name
- volume
- mins
- maxs

- membraneThickness
- memtype
- cyttype

### 7.5.1 Detailed Description

Base class for a particular cell type for use in CellArranger (and later more)

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 def pySTDLM.Cells.CellShape.__init__ ( *self* )

### 7.5.3 Member Function Documentation

#### 7.5.3.1 def pySTDLM.Cells.CellShape.addToSimulation ( *self, sim* )

Add the cell to the simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *sim* | An RDMESimulation object |

#### 7.5.3.2 def pySTDLM.Cells.CellShape.boundingBox ( *self* )

Return a bounding box for the cell TODO: Make it return an orientation as well.

#### 7.5.3.3 def pySTDLM.Cells.CellShape.createModelCell ( *self, attr* )

Create model cell.

**Parameters**

| | |
|---|---|
| *self* | |
| *attrs* | A dictionary of attributes |

#### 7.5.3.4 def pySTDLM.Cells.CellShape.getVolume ( *self* )

Volume occupied by cell NOTE: When overriding this class, you must specify a "computeVolume()" function of no arguments.

**Returns**

volume In intrinsic units

#### 7.5.3.5 def pySTDLM.Cells.CellShape.setRegions ( *self, membrane, cytoplasm* )

Set regions.

**Parameters**

| | |
|---|---|
| *self* | |
| *membrane* | The name of the region in which the membrane should be considered |
| *cytoplasm* | The name of the region in which the cytoplasm should be considered |

#### 7.5.3.6 def pySTDLM.Cells.CellShape.translateCell ( *self, point* )

Shift a cell in space by the specified amount.

**Parameters**

| | |
|---|---|
| *self* | |
| *point* | [x,y,z] translation in space |

### 7.5.4 Member Data Documentation

#### 7.5.4.1 pySTDLM.Cells.CellShape.cyttype

#### 7.5.4.2 pySTDLM.Cells.CellShape.maxs

#### 7.5.4.3 pySTDLM.Cells.CellShape.membraneThickness

#### 7.5.4.4 pySTDLM.Cells.CellShape.memtype

#### 7.5.4.5 pySTDLM.Cells.CellShape.mins

#### 7.5.4.6 pySTDLM.Cells.CellShape.name

#### 7.5.4.7 pySTDLM.Cells.CellShape.volume

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

- Cells.py

## 7.6 pyLM.CME.CMESimulation Class Reference

A CME simulation that contains reactions and species.

### Public Member Functions

- def __init__

  *A constructor fo the CMESimulation.*
- def defineSpecies

  *Define a specie/s of particles that exist in the simulation.*
- def addParticles

  *Add a specified number of particles of the specified type to the specified region.*
- def addConcentration

  *Add a concentration of particles of the specified type to the simulation.*
- def addReaction

  *Adds a 0th, 1st or 2nd order reaction.*
- def buildReactionModel

*Return the Lattice Microbes ReactionModel object for fine-tuning.*

- def setRandomSeed
- def setWriteInterval

  *Set the simulation state write-to-disk interval.*
- def setSimulationTime

  *Set the total simulation time.*
- def setTimestep
- def save

  *Create an HDF5 version of the simulation amenable for later running or stand-alone running.*
- def run

  *Run the simulation using the specified solver the specified amount of time.*
- def runMPI

  *Run the simulation using a call to mpirun with the given options.*
- def runSolver

### Public Attributes

- particleMap
- species_id
- initial_counts
- reactions
- parameters
- volume
- name
- replicates
- filename

## 7.6.1 Detailed Description

A CME simulation that contains reactions and species.

## 7.6.2 Constructor & Destructor Documentation

### 7.6.2.1 def pyLM.CME.CMESimulation.__init__ ( *self*, *volume* = `None`, *name* = `"unnamed"` )

A constructor fo the CMESimulation.

**Parameters**

| | |
|---|---|
| *volume* | The reaction vessel volume (in Litres). Specifying 'None' signifies that the user has already accounted for volume in rate constants. |
| *name* | The name of the CME simulation; Default: "unnamed" |

**Returns**

self

## 7.6.3 Member Function Documentation

### 7.6.3.1 def pyLM.CME.CMESimulation.addConcentration ( *self*, *species* = `'unknown'`, *conc* = `0.0` )

Add a concentration of particles of the specified type to the simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The name of the specie to add |
| *concentration* | The concentration of the species (Molar). Particle count is rounded to nearest integer. |

**Exceptions**

| | |
|---|---|
| *An* | error if the volume is not specified |

#### 7.6.3.2 def pyLM.CME.CMESimulation.addParticles ( *self, species =* `'unknown'`, *count =* `1` )

Add a specified number of particles of the specified type to the specified region.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The name of the specie to add particles to |
| *count* | The number of that particle to start with (default 1) |

#### 7.6.3.3 def pyLM.CME.CMESimulation.addReaction ( *self, reactant, product, rate* )

Adds a 0th, 1st or 2nd order reaction.

**Parameters**

| | |
|---|---|
| *self* | |
| *reactant* | The list or reactants |
| *product* | The list of products |
| *rate* | The rate of reaction |

#### 7.6.3.4 def pyLM.CME.CMESimulation.buildReactionModel ( *self* )

Return the Lattice Microbes ReactionModel object for fine-tuning.

**Parameters**

| | |
|---|---|
| *self* | |

**Returns**

The reaction model for the simulation

#### 7.6.3.5 def pyLM.CME.CMESimulation.defineSpecies ( *self, species* )

Define a specie/s of particles that exist in the simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | A list of species to add to the simulation |

#### 7.6.3.6 def pyLM.CME.CMESimulation.run ( *self, filename, method, replicates =* `1`, *seed =* `None` )

Run the simulation using the specified solver the specified amount of time.

**Parameters**

| | |
|---|---|
| *self* | |
| *filename* | The HDF file to write to |
| *method* | The class name for the solver to use (e.g., lm::cme::GillespieDSolver") |
| *replicates* | The number of replicates to serially run |
| *seed* | A seed for the random number generator to use when running the simulation; None indicates default |

#### 7.6.3.7 def pyLM.CME.CMESimulation.runMPI ( *self, filename, method, replicates* = `1`, *driver* = `"mpirun"`, *ppe* = `1`, *seed* = `None` )

Run the simulation using a call to mpirun with the given options.

**Parameters**

| | |
|---|---|
| *self* | |
| *filename* | The HDF file to write to |
| *method* | The class name for the solver to use (e.g., lm::cme::GillespieDSolver") |
| *replicates* | The number of replicates to serially run |
| *driver* | The program to execute the parallel run, e.g. "mpirun", "aprun", "ibrun", etc. |
| *ppe* | The number of processing elements to use |
| *seed* | A seed for the random number generator to use when running the simulation; None indicates default |

#### 7.6.3.8 def pyLM.CME.CMESimulation.runSolver ( *self, filename, solver, replicates* = `1` )

#### 7.6.3.9 def pyLM.CME.CMESimulation.save ( *self, filename* )

Create an HDF5 version of the simulation amenable for later running or stand-alone running.

**Parameters**

| | |
|---|---|
| *self* | |
| *filename* | The filename to save the simulation setup in |

#### 7.6.3.10 def pyLM.CME.CMESimulation.setRandomSeed ( *self, seed* )

#### 7.6.3.11 def pyLM.CME.CMESimulation.setSimulationTime ( *self, time* )

Set the total simulation time.

**Parameters**

| | |
|---|---|
| *self* | |
| *time* | Time length of the simulation |

#### 7.6.3.12 def pyLM.CME.CMESimulation.setTimestep ( *self, time* )

#### 7.6.3.13 def pyLM.CME.CMESimulation.setWriteInterval ( *self, time* )

Set the simulation state write-to-disk interval.

**Parameters**

| | |
|---|---|
| *self* | |
| *time* | Time length between writes |

### 7.6.4 Member Data Documentation

#### 7.6.4.1 pyLM.CME.CMESimulation.filename

#### 7.6.4.2 pyLM.CME.CMESimulation.initial_counts

#### 7.6.4.3 pyLM.CME.CMESimulation.name

#### 7.6.4.4 pyLM.CME.CMESimulation.parameters

#### 7.6.4.5 pyLM.CME.CMESimulation.particleMap

#### 7.6.4.6 pyLM.CME.CMESimulation.reactions

#### 7.6.4.7 pyLM.CME.CMESimulation.replicates

#### 7.6.4.8 pyLM.CME.CMESimulation.species_id

#### 7.6.4.9 pyLM.CME.CMESimulation.volume

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

- CME.py

## 7.7 pyLM.RDME.CMESimulation Class Reference

A CME simulation that contains reactions and species.

### Public Member Functions

- def __init__

  *A constructor fo the CMESimulation.*
- def defineSpecies

  *Define a specie/s of particles that exist in the simulation.*
- def addParticles

  *Add a specified number of particles of the specified type to the specified region.*
- def addConcentration

  *Add a concentration of particles of the specified type to the simulation.*
- def addReaction

  *Adds a 0th, 1st or 2nd order reaction.*

### Public Attributes

- particleMap
- species_id
- initial_counts
- reactions

- parameters
- volume
- name
- replicates
- filename

### 7.7.1 Detailed Description

A CME simulation that contains reactions and species.

### 7.7.2 Constructor & Destructor Documentation

#### 7.7.2.1 def pyLM.RDME.CMESimulation.__init__ ( *self*, *volume* = `None`, *name* = `"unnamed"` )

A constructor fo the CMESimulation.

**Parameters**

| | |
|---|---|
| *volume* | The reaction vessel volume (in Litres). Specifying 'None' signifies that the user has already accounted for volume in rate constants. |
| *name* | The name of the CME simulation; Default: "unnamed" |

**Returns**

self

### 7.7.3 Member Function Documentation

#### 7.7.3.1 def pyLM.RDME.CMESimulation.addConcentration ( *self*, *species* = `'unknown'`, *conc* = `0.0` )

Add a concentration of particles of the specified type to the simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The name of the specie to add |
| *concentration* | The concentration of the species (Molar). Particle count is rounded to nearest integer. |

**Exceptions**

| | |
|---|---|
| *An* | error if the volume is not specified |

#### 7.7.3.2 def pyLM.RDME.CMESimulation.addParticles ( *self*, *species* = `'unknown'`, *count* = `1` )

Add a specified number of particles of the specified type to the specified region.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The name of the specie to add particles to |
| *count* | The number of that particle to start with (default 1) |

#### 7.7.3.3 def pyLM.RDME.CMESimulation.addReaction ( *self*, *reactant*, *product*, *rate* )

Adds a 0th, 1st or 2nd order reaction.

**Parameters**

| | |
|---|---|
| *self* | |
| *reactant* | The list or reactants |
| *product* | The list of products |
| *rate* | The rate of reaction |

#### 7.7.3.4 def pyLM.RDME.CMESimulation.defineSpecies ( *self, species* )

Define a specie/s of particles that exist in the simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | A list of species to add to the simulation |

### 7.7.4 Member Data Documentation

#### 7.7.4.1 pyLM.RDME.CMESimulation.filename

#### 7.7.4.2 pyLM.RDME.CMESimulation.initial_counts

#### 7.7.4.3 pyLM.RDME.CMESimulation.name

#### 7.7.4.4 pyLM.RDME.CMESimulation.parameters

#### 7.7.4.5 pyLM.RDME.CMESimulation.particleMap

#### 7.7.4.6 pyLM.RDME.CMESimulation.reactions

#### 7.7.4.7 pyLM.RDME.CMESimulation.replicates

#### 7.7.4.8 pyLM.RDME.CMESimulation.species_id

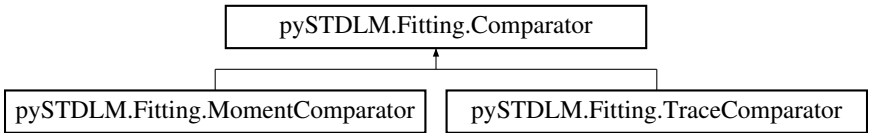
#### 7.7.4.9 pyLM.RDME.CMESimulation.volume

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

- RDME.py

## 7.8 pySTDLM.Fitting.Comparator Class Reference

Inheritance diagram for pySTDLM.Fitting.Comparator:




### Public Member Functions

- def __init__

- def computeMetric

### Public Attributes

- specie

### 7.8.1 Constructor & Destructor Documentation

#### 7.8.1.1 def pySTDLM.Fitting.Comparator.__init__ ( *self, specie* )

### 7.8.2 Member Function Documentation

#### 7.8.2.1 def pySTDLM.Fitting.Comparator.computeMetric ( *self, fileHandle, time* = None, *data* = None )

### 7.8.3 Member Data Documentation

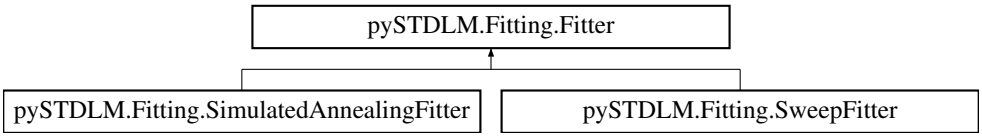
#### 7.8.3.1 pySTDLM.Fitting.Comparator.specie

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

- Fitting.py

## 7.9 pySTDLM.Fitting.Fitter Class Reference

Inheritance diagram for pySTDLM.Fitting.Fitter:




### Public Member Functions

- def __init__
- def addExperiment
- def addParameter
- def performFit
- def saveFit

### Public Attributes

- experiments
- parameters
- hasFit

### 7.9.1 Constructor & Destructor Documentation

#### 7.9.1.1 def pySTDLM.Fitting.Fitter.__init__ ( *self* )

### 7.9.2 Member Function Documentation

#### 7.9.2.1 def pySTDLM.Fitting.Fitter.addExperiment ( *self, simulation, comparisons, type, weight* = None, *name* = None )

#### 7.9.2.2 def pySTDLM.Fitting.Fitter.addParameter ( *self, parameter, bounds, initialValue* )

#### 7.9.2.3 def pySTDLM.Fitting.Fitter.performFit ( *self, numberProcElements* )

#### 7.9.2.4 def pySTDLM.Fitting.Fitter.saveFit ( *self, filebase* )

### 7.9.3 Member Data Documentation

#### 7.9.3.1 pySTDLM.Fitting.Fitter.experiments

#### 7.9.3.2 pySTDLM.Fitting.Fitter.hasFit

#### 7.9.3.3 pySTDLM.Fitting.Fitter.parameters

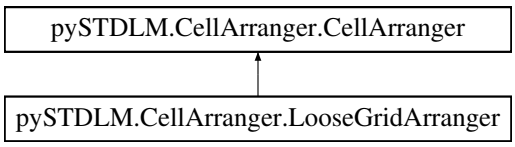
The documentation for this class was generated from the following file:

- Fitting.py

## 7.10 pySTDLM.CellArranger.LooseGridArranger Class Reference

An arranger that orients cells in square grid with padding in between.

Inheritance diagram for pySTDLM.CellArranger.LooseGridArranger:




### Public Member Functions

- def __init__
- def intPackVolume

### Additional Inherited Members

### 7.10.1 Detailed Description

An arranger that orients cells in square grid with padding in between.

The attribute dictionary requires "offset" which is a list of the form [dx, dy, dz].

### 7.10.2 Constructor & Destructor Documentation

#### 7.10.2.1 def pySTDLM.CellArranger.LooseGridArranger.__init__ ( *self, cellType, cellAttributes, arrangerAttributes* )

### 7.10.3 Member Function Documentation

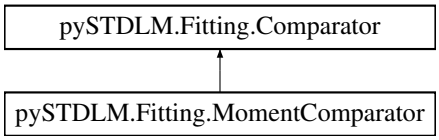
#### 7.10.3.1 def pySTDLM.CellArranger.LooseGridArranger.intPackVolume ( *self, sim, mins, maxs* )

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

- CellArranger.py

## 7.11 pySTDLM.Fitting.MomentComparator Class Reference

Inheritance diagram for pySTDLM.Fitting.MomentComparator:




### Public Member Functions

- def __init__
- def computeMetric

### Public Attributes

- moment
- value
- sem

### 7.11.1 Constructor & Destructor Documentation

#### 7.11.1.1 def pySTDLM.Fitting.MomentComparator.__init__ ( *self, specie, moment, value* = `None`, *sem* = `None` )

### 7.11.2 Member Function Documentation

#### 7.11.2.1 def pySTDLM.Fitting.MomentComparator.computeMetric ( *self, fileHandle, time* = `None`, *data* = `None` )

### 7.11.3 Member Data Documentation

#### 7.11.3.1 pySTDLM.Fitting.MomentComparator.moment

#### 7.11.3.2 pySTDLM.Fitting.MomentComparator.sem

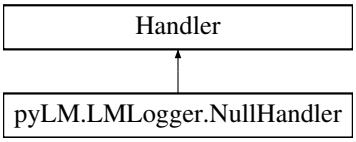
#### 7.11.3.3 pySTDLM.Fitting.MomentComparator.value

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

- Fitting.py

## 7.12 pyLM.LMLogger.NullHandler Class Reference

Inheritance diagram for pyLM.LMLogger.NullHandler:




### Public Member Functions

- def emit

### 7.12.1 Member Function Documentation

#### 7.12.1.1 def pyLM.LMLogger.NullHandler.emit ( *self, record* )

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

- LMLogger.py

## 7.13 pySTDLM.Distributions.ParticleGenerator Class Reference

A particle generator that adds particles to a particular region of a simulation.

### Public Member Functions

- def __init__
  *Constructor.*
- def addParticlesToRegion
  *Add a number of particles to a region based on Monte-Carlo sampling of the generator.*

### Public Attributes

- generator
- particlename
- totalAdded
- addedLast

### 7.13.1 Detailed Description

A particle generator that adds particles to a particular region of a simulation.

### 7.13.2 Constructor & Destructor Documentation

#### 7.13.2.1 def pySTDLM.Distributions.ParticleGenerator.__init__ ( *self, gen, ptype* )

Constructor.

**Parameters**

| | |
|---|---|
| *gen* | A PDF set for x,y,z that on the domain [-1,1] and range [0,1) where x is the range of interest |
| *ptype* | The name of the particle |

### 7.13.3 Member Function Documentation

#### 7.13.3.1 def pySTDLM.Distributions.ParticleGenerator.addParticlesToRegion ( *self, rdmesim, region, number, directions, spatial* = `None` )

Add a number of particles to a region based on Monte-Carlo sampling of the generator.

**Parameters**

| | |
|---|---|
| *self* | |
| *rdmesim* | An RDME simulation object that has already been "discretized" |
| *region* | A region name of the simulation to add particles to |
| *number* | The number of the particle to add |
| *directions* | A list of the form (x,y,z) where a 0 indicates a uniform distribution in that direction and a 1 indicates the particles are sampled from the generator in that direction |
| *spatial* | (OPTIONAL) A definition of a subsection of the domain that this generation should occur in as a list of lists [[xl,yl,zl],[xh,yh,zh]], otherwise the generator is scaled from (0,0,0) to domain extends (w,h,d) |

### 7.13.4 Member Data Documentation

#### 7.13.4.1 pySTDLM.Distributions.ParticleGenerator.addedLast

#### 7.13.4.2 pySTDLM.Distributions.ParticleGenerator.generator

#### 7.13.4.3 pySTDLM.Distributions.ParticleGenerator.particlename

#### 7.13.4.4 pySTDLM.Distributions.ParticleGenerator.totalAdded

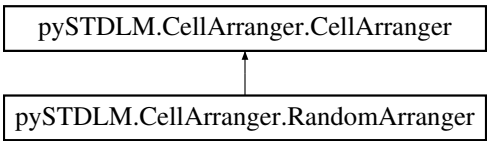
The documentation for this class was generated from the following file:

- Distributions.py

## 7.14 pySTDLM.CellArranger.RandomArranger Class Reference

An arranger that randomly orients a number of cells in a regin until either a count or a volume fraction target is met.

Inheritance diagram for pySTDLM.CellArranger.RandomArranger:




### Public Member Functions

- def __init__
- def newBBox

- def doesIntersect
- def outsideDomain
- def intPackVolume

### Additional Inherited Members

### 7.14.1 Detailed Description

An arranger that randomly orients a number of cells in a regin until either a count or a volume fraction target is met.

This class requires either "fraction" or "number" attributes specified in the arrangerAttributes. It also has an optional parameter "padding" which maintains that much padding between the cells and their neighbors.

### 7.14.2 Constructor & Destructor Documentation

#### 7.14.2.1 def pySTDLM.CellArranger.RandomArranger.__init__ ( *self, cellType, cellAttributes, arrangerAttributes* )

### 7.14.3 Member Function Documentation

#### 7.14.3.1 def pySTDLM.CellArranger.RandomArranger.doesIntersect ( *self, b1, b2* )

#### 7.14.3.2 def pySTDLM.CellArranger.RandomArranger.intPackVolume ( *self, sim, mins, maxs* )

#### 7.14.3.3 def pySTDLM.CellArranger.RandomArranger.newBBox ( *self, bbox, padding* )

#### 7.14.3.4 def pySTDLM.CellArranger.RandomArranger.outsideDomain ( *self, b1, mins, maxs* )

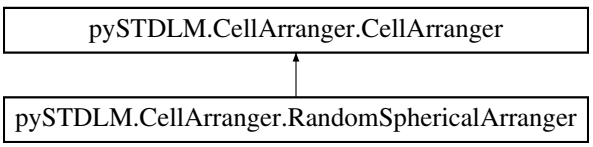
The documentation for this class was generated from the following file:

- CellArranger.py

## 7.15 pySTDLM.CellArranger.RandomSphericalArranger Class Reference

An arranger that randomly orients a number of cells in a sphere inside a regin until either a count or a volume fraction target is met.

Inheritance diagram for pySTDLM.CellArranger.RandomSphericalArranger:




### Public Member Functions

- def __init__
- def intPackVolume

### Additional Inherited Members

### 7.15.1 Detailed Description

An arranger that randomly orients a number of cells in a sphere inside a regin until either a count or a volume fraction target is met.

This class requires either "fraction" or "number" attributes specified in the arrangerAttributes. It also has an optional parameter "padding" which maintains that much padding between the cells and their neighbors.

### 7.15.2 Constructor & Destructor Documentation

#### 7.15.2.1 def pySTDLM.CellArranger.RandomSphericalArranger.__init__ ( *self, cellType, cellAttributes, arrangerAttributes* )

### 7.15.3 Member Function Documentation

#### 7.15.3.1 def pySTDLM.CellArranger.RandomSphericalArranger.intPackVolume ( *self, sim, mins, maxs* )

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

- CellArranger.py

## 7.16 pyLM.RDME.RDMERegion Class Reference

A class that represents a type region of an RDME simulation.

### Public Member Functions

- def __init__

  *Creates a new RDMERegion.*
- def addReaction

  *Adds a 0th, 1st or 2nd order reaction that can occur in the region.*
- def setDefaultDiffusionRate

  *Specifies the default diffusion rate of all particles in the region.*
- def setDiffusionRate

  *Specify the diffusion rate for a particular particle type.*
- def getReactionCount

  *Return the number of reactions defined in this region.*

### Public Attributes

- name
- reactions
- defaultDiffusionRate
- diffusionRate

### 7.16.1 Detailed Description

A class that represents a type region of an RDME simulation.

Reactions may be specified that live within a region. In addition a specie's diffusion constant is region dependent and diffusion between regions must be specified. For example: cytosol, membrane, extracellular, nucleoid, etc.

### 7.16.2 Constructor & Destructor Documentation

#### 7.16.2.1 def pyLM.RDME.RDMERegion.__init__ ( *self, name* )

Creates a new RDMERegion.

**Parameters**

| | |
|---|---|
| *self* | |
| *name* | The name of the region (e.g. cytosol) |

### 7.16.3 Member Function Documentation

#### 7.16.3.1 def pyLM.RDME.RDMERegion.addReaction ( *self, reactant, product, rate* )

Adds a 0th, 1st or 2nd order reaction that can occur in the region.

**Parameters**

| | |
|---|---|
| *self* | |
| *reactant* | A set of reactants either as a singleton or a list |
| *product* | A set of products either as a singeton or a list |

**Returns**

self

#### 7.16.3.2 def pyLM.RDME.RDMERegion.getReactionCount ( *self* )

Return the number of reactions defined in this region.

**Parameters**

| | |
|---|---|
| *self* | |

**Returns**

Get the number of reactions in the region

#### 7.16.3.3 def pyLM.RDME.RDMERegion.setDefaultDiffusionRate ( *self, rate* )

Specifies the default diffusion rate of all particles in the region.

**Parameters**

| | |
|---|---|
| *self* | |
| *rate* | The rate of diffusion in um^2/s or um/s for 3D or 2D diffusion |

**Returns**

self

#### 7.16.3.4 def pyLM.RDME.RDMERegion.setDiffusionRate ( *self, species, rate* )

Specify the diffusion rate for a particular particle type.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The particle type |
| *rate* | The rate of diffusion in um^2/s or um/s for 3D or 2D diffusion |

**Returns**

self

### 7.16.4 Member Data Documentation

#### 7.16.4.1 pyLM.RDME.RDMERegion.defaultDiffusionRate

#### 7.16.4.2 pyLM.RDME.RDMERegion.diffusionRate

#### 7.16.4.3 pyLM.RDME.RDMERegion.name

#### 7.16.4.4 pyLM.RDME.RDMERegion.reactions

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

- RDME.py

## 7.17 pyLM.RDME.RDMESimulation Class Reference

A class that contains all regions, reactions, diffusions and rules for a RDME simulation.

### Public Member Functions

- def __init__

  *Specify a cuboid region that represents the extents to the reaction region as well as the lattice spacing.*
- def addRegion

  *Add a region to the simulation.*
- def addCuboidRegion

  *Add a cuboid to the builder.*
- def addShape
- def modifyRegion

  *Return a pointer to a region so that it may be modified.*
- def defineSpecies

  *Define a specie/s of particles that exist in the simulation.*
- def siteVolume

  *Get the actual volume of a specific site in L.*
- def buildCapsidCell

  *Build a capsule based shell in this RDMESimulation centered within the simulation domain that includes a membrane and cytoplasm.*
- def buildSphericalCell

  *Build a spherical based shell in this RDMESimulation centered within the simulation domain that includes a membrane and cytoplasm.*
- def addParticles

  *Add a specified number of particles of the specified type to the specified region.*
- def packRegion

  *Add nonmoving obstacles to a particular region.*

- def setTransitionRate

  *Specify the diffusion rate between species; this is a one directional rate e.g.*
- def setTwoWayTransitionRate

  *Specify the diffusion rate between species; this is a two directional rate.*
- def buildDiffusionModel

  *Return the Lattice Microbes DiffusionModel object for fine-tuning.*
- def buildReactionModel

  *Return the Lattice Microbes ReactionModel object for fine-tuning.*
- def buildSpatialModel

  *Return the Lattice Microbes SpatialModel object for fine-tuning.*
- def setWriteInterval
- def setLatticeWriteInterval
- def setSimulationTime
- def setTimestep
- def setRandomSeed
- def getLattice

  *Get a discretized version of the simulation domain.*
- def setLatticeSite

  *Set a particular lattice site type.*
- def getLatticeSite

  *Get a particular lattice site.*
- def addParticleAt

  *Add a particle/ to a particular site.*
- def addParticleAtIdx

  *Create an HDF5 version of the simulation amenable for later running or stand-alone running.*
- def save

  *Create an HDF5 version of the simulation amenable for later running or stand-alone running.*
- def run
- def runMPI

  *Run the simulation using a call to mpirun with the given options.*
- def runSolver
- def buildVector
- def buildPoint
- def buildSphere
- def buildEllipse
- def buildDifference

## Public Attributes

- siteTypes
- particleMap
- customAddedParticleList
- species_id
- regions
- transitionRates
- initial_counts
- parameters
- continousDimensions
- latticeSpacing
- lm_builder
- lattice
- hasBeenDiscretized
- name
- replicates
- filename

### 7.17.1 Detailed Description

A class that contains all regions, reactions, diffusions and rules for a RDME simulation.

### 7.17.2 Constructor & Destructor Documentation

#### 7.17.2.1 def pyLM.RDME.RDMESimulation.__init__ ( *self, dimensions, spacing, name* = `"unnamed"`, *defaultRegion* = `"default"` )

Specify a cuboid region that represents the extents to the reaction region as well as the lattice spacing.

**Parameters**

| | |
|---|---|
| *self* | |
| *dimensions* | A list of [x,y,z] |
| *spacing* | Lattice spacing |
| *name* | The name of the RDME simulation; default: "unnamed" |
| *defaultRegion* | The name of the region that is associated with the lattice sites before any other regions are added; default:"default" |

**Returns**

self

### 7.17.3 Member Function Documentation

#### 7.17.3.1 def pyLM.RDME.RDMESimulation.addCuboidRegion ( *self, name, a, b* )

Add a cuboid to the builder.

**Parameters**

| | |
|---|---|
| *self* | |
| *name* | Name of the site type for this region |
| *a* | tuple for the first corner in continous space |
| *b* | tuple for the second corner in continous space |

#### 7.17.3.2 def pyLM.RDME.RDMESimulation.addParticleAt ( *self, index, particleType* )

Add a particle/ to a particular site.

**Parameters**

| | |
|---|---|
| *index* | (x,y,z) a list of spatial location |
| *specie* | The specie type to add |

#### 7.17.3.3 def pyLM.RDME.RDMESimulation.addParticleAtIdx ( *self, index, particleType* )

Create an HDF5 version of the simulation amenable for later running or stand-alone running.

**Parameters**

| | |
|---|---|
| *self* | |

| | |
|---|---|
| *filename* | A file to write the simulation to Add a particle/ to a particular site |
| *index* | (x,y,z) a list of lattice site indices |
| *specie* | The specie type to add |

#### 7.17.3.4 def pyLM.RDME.RDMESimulation.addParticles ( *self, species* = 'unknown', *region* = 'default', *count* = 1 )

Add a specified number of particles of the specified type to the specified region.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The species to add to the region |
| *region* | The region to add particles to |
| *count* | Number of particles to add (default: 1) |

**Returns**

The simulation object

#### 7.17.3.5 def pyLM.RDME.RDMESimulation.addRegion ( *self, region* )

Add a region to the simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *region* | The region to add to the simulation |

**Returns**

The region just added

#### 7.17.3.6 def pyLM.RDME.RDMESimulation.addShape ( *self, shape* )

#### 7.17.3.7 def pyLM.RDME.RDMESimulation.buildCapsidCell ( *self, length, diameter, membraneThickness, points* = False )

Build a capsule based shell in this RDMESimulation centered within the simulation domain that includes a membrane and cytoplasm.

**Parameters**

| | |
|---|---|
| *self* | |
| *length* | The length of the capsule from one sphere origin to the other |
| *diameter* | The diameter of the cell |
| *membrane-Thickness* | The thickness of the membrane |
| *points* | OPTIONAL: List of lists containing the coordinates of the centers of the sphereoids that cap the capsid cell, e.g. [[x1, y1, z1], [x2, y2, z2]]. If not given, cell is centered in the volume and aligned in the z-direction |

**Returns**

The simulation object

#### 7.17.3.8 def pyLM.RDME.RDMESimulation.buildDifference ( *self, arg1, arg2, arg3* )

#### 7.17.3.9 def pyLM.RDME.RDMESimulation.buildDiffusionModel ( *self* )

Return the Lattice Microbes DiffusionModel object for fine-tuning.

**Parameters**

| | |
|---|---|
| *self* | The diffusion modle for the simulation |

#### 7.17.3.10 def pyLM.RDME.RDMESimulation.buildEllipse ( *self, arg1, arg2, arg3, arg4, arg5, arg6, arg7* )

#### 7.17.3.11 def pyLM.RDME.RDMESimulation.buildPoint ( *self, arg1, arg2, arg3* )

#### 7.17.3.12 def pyLM.RDME.RDMESimulation.buildReactionModel ( *self* )

Return the Lattice Microbes ReactionModel object for fine-tuning.

**Parameters**

| | |
|---|---|
| *self* | |

**Returns**

The reaction model for this simulation

#### 7.17.3.13 def pyLM.RDME.RDMESimulation.buildSpatialModel ( *self* )

Return the Lattice Microbes SpatialModel object for fine-tuning.

**Parameters**

| | |
|---|---|
| *self* | |

**Returns**

The spatial model (i.e. obstacles, sites, etc.) for this simulation

#### 7.17.3.14 def pyLM.RDME.RDMESimulation.buildSphere ( *self, arg1, arg2, arg3* )

#### 7.17.3.15 def pyLM.RDME.RDMESimulation.buildSphericalCell ( *self, diameter, membraneThickness, point* = `False` )

Build a spherical based shell in this RDMESimulation centered within the simulation domain that includes a membrane and cytoplasm.

**Parameters**

| | |
|---|---|
| *self* | |
| *diameter* | The diameter of the cell |
| *membrane-Thickness* | The thickness of the membrane |

| | |
|---|---|
| *point* | The center of the spherical cell |

**Returns**

The simulation object

#### 7.17.3.16 def pyLM.RDME.RDMESimulation.buildVector ( *self, arg1, arg2, arg3* )

#### 7.17.3.17 def pyLM.RDME.RDMESimulation.defineSpecies ( *self, species* )

Define a specie/s of particles that exist in the simulation.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | A list of species to add to the simulation |

**Returns**

The simulation object

#### 7.17.3.18 def pyLM.RDME.RDMESimulation.getLattice ( *self* )

Get a discretized version of the simulation domain.

Call this after building all spherical and capsule cells

**Parameters**

| | |
|---|---|
| *self* | |

**Returns**

A lattice object. This function should only be called once.

#### 7.17.3.19 def pyLM.RDME.RDMESimulation.getLatticeSite ( *self, index* )

Get a particular lattice site.

**Parameters**

| | |
|---|---|
| *index* | (x,y,z) a list of coordinates |

**Returns**

The type of the lattice site (string)

#### 7.17.3.20 def pyLM.RDME.RDMESimulation.modifyRegion ( *self, region* )

Return a pointer to a region so that it may be modified.

**Parameters**

| | |
|---|---|
| *self* | |
| *region* | Get a region that is attached to the simulation for modification |

**Returns**

The region to modify

#### 7.17.3.21 def pyLM.RDME.RDMESimulation.packRegion ( *self, region, radius, percentage, obstacleID* )

Add nonmoving obstacles to a particular region.

**Parameters**

| | |
|---|---|
| *self* | |
| *region* | The name of the region in which to add particles to |
| *radius* | The radius of the particles |
| *percentage* | The percentage of the total region volume that should be packed |
| *obstacleID* | an identifier for the obstacle |

**Returns**

The simulation object

#### 7.17.3.22 def pyLM.RDME.RDMESimulation.run ( *self, filename, method, replicates* = `1`, *seed* = `None` )

#### 7.17.3.23 def pyLM.RDME.RDMESimulation.runMPI ( *self, filename, method, replicates* = `1`, *driver* = `"mpirun"`, *ppe* = `1`, *seed* = `None` )

Run the simulation using a call to mpirun with the given options.

**Parameters**

| | |
|---|---|
| *self* | |
| *filename* | The HDF file to write to |
| *method* | The class name for the solver to use (e.g., lm::cme::GillespieDSolver") |
| *replicates* | The number of replicates to serially run |
| *driver* | The program to execute the parallel run, e.g. "mpirun", "aprun", "ibrun", etc. |
| *ppe* | The number of processing elements to use |
| *seed* | A seed for the random number generator to use when running the simulation; None indicates default |

#### 7.17.3.24 def pyLM.RDME.RDMESimulation.runSolver ( *self, filename, solver, replicates* = `1` )

#### 7.17.3.25 def pyLM.RDME.RDMESimulation.save ( *self, filename* )

Create an HDF5 version of the simulation amenable for later running or stand-alone running.

**Parameters**

| | |
|---|---|
| *self* | |

| | |
|---|---|
| *filename* | A file to write the simulation to |

#### 7.17.3.26 def pyLM.RDME.RDMESimulation.setLatticeSite ( *self, index, siteType* )

Set a particular lattice site type.

**Parameters**

| | |
|---|---|
| *index* | (x,y,z) a list of coordinates |
| *siteType* | The type to set the lattice point to. This would be the name of a region that has previously been performed" |

#### 7.17.3.27 def pyLM.RDME.RDMESimulation.setLatticeWriteInterval ( *self, time* )

#### 7.17.3.28 def pyLM.RDME.RDMESimulation.setRandomSeed ( *self, seed* )

#### 7.17.3.29 def pyLM.RDME.RDMESimulation.setSimulationTime ( *self, time* )

#### 7.17.3.30 def pyLM.RDME.RDMESimulation.setTimestep ( *self, time* )

#### 7.17.3.31 def pyLM.RDME.RDMESimulation.setTransitionRate ( *self, species, via, to, rate* )

Specify the diffusion rate between species; this is a one directional rate e.g.

membrane->cytosol or extracellular->membrane

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The specie that can transition between regions |
| *via* | From this region |
| *to* | To this region |
| *rate* | Diffusion rate between regions |

**Returns**

The simulation object

#### 7.17.3.32 def pyLM.RDME.RDMESimulation.setTwoWayTransitionRate ( *self, species, one, two, rate* )

Specify the diffusion rate between species; this is a two directional rate.

**Parameters**

| | |
|---|---|
| *self* | |
| *species* | The specie that can transition between regions |
| *one* | A region |
| *two* | The other region |
| *rate* | Diffusion rate between regions |

**Returns**

The simulation object

#### 7.17.3.33 def pyLM.RDME.RDMESimulation.setWriteInterval ( *self, time* )

#### 7.17.3.34 def pyLM.RDME.RDMESimulation.siteVolume ( *self* )

Get the actual volume of a specific site in L.

**Returns**

the volume of the site in L

### 7.17.4 Member Data Documentation

#### 7.17.4.1 pyLM.RDME.RDMESimulation.continousDimensions

#### 7.17.4.2 pyLM.RDME.RDMESimulation.customAddedParticleList

#### 7.17.4.3 pyLM.RDME.RDMESimulation.filename

#### 7.17.4.4 pyLM.RDME.RDMESimulation.hasBeenDiscretized

#### 7.17.4.5 pyLM.RDME.RDMESimulation.initial_counts

#### 7.17.4.6 pyLM.RDME.RDMESimulation.lattice

#### 7.17.4.7 pyLM.RDME.RDMESimulation.latticeSpacing

#### 7.17.4.8 pyLM.RDME.RDMESimulation.lm_builder

#### 7.17.4.9 pyLM.RDME.RDMESimulation.name

#### 7.17.4.10 pyLM.RDME.RDMESimulation.parameters

#### 7.17.4.11 pyLM.RDME.RDMESimulation.particleMap

#### 7.17.4.12 pyLM.RDME.RDMESimulation.regions

#### 7.17.4.13 pyLM.RDME.RDMESimulation.replicates

#### 7.17.4.14 pyLM.RDME.RDMESimulation.siteTypes

#### 7.17.4.15 pyLM.RDME.RDMESimulation.species_id

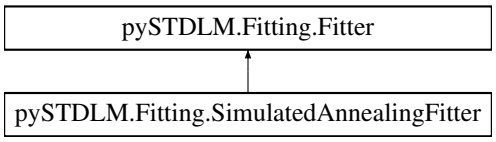
#### 7.17.4.16 pyLM.RDME.RDMESimulation.transitionRates

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

- RDME.py

## 7.18 pySTDLM.Fitting.SimulatedAnnealingFitter Class Reference

Inheritance diagram for pySTDLM.Fitting.SimulatedAnnealingFitter:




### Public Member Functions

- def __init__
- def performFit
- def saveFit

### Additional Inherited Members

#### 7.18.1 Constructor & Destructor Documentation

##### 7.18.1.1 def pySTDLM.Fitting.SimulatedAnnealingFitter.__init__ ( *self* )

#### 7.18.2 Member Function Documentation

##### 7.18.2.1 def pySTDLM.Fitting.SimulatedAnnealingFitter.performFit ( *self, numberProcElements* )

##### 7.18.2.2 def pySTDLM.Fitting.SimulatedAnnealingFitter.saveFit ( *self, filebase* )

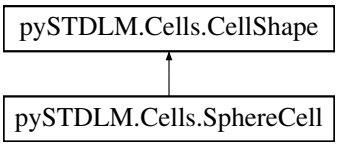
The documentation for this class was generated from the following file:

- Fitting.py

## 7.19 pySTDLM.Cells.SphereCell Class Reference

A representation for a spherical cell.

Inheritance diagram for pySTDLM.Cells.SphereCell:




### Public Member Functions

- def __init__
- def internalModelCell
- def computeVolume
- def internalAddToSim
- def internalTranslateCell

### Public Attributes

- name
- origin
- radius
- mins
- maxs
- membraneThickness

### 7.19.1 Detailed Description

A representation for a spherical cell.

This cell requires an attribute dictionary that includes "radius" and "membraneThickness"

### 7.19.2 Constructor & Destructor Documentation

#### 7.19.2.1 def pySTDLM.Cells.SphereCell.__init__ ( *self, origin* = `3*[0.]`, *radius* = `0.` )

### 7.19.3 Member Function Documentation

#### 7.19.3.1 def pySTDLM.Cells.SphereCell.computeVolume ( *self* )

#### 7.19.3.2 def pySTDLM.Cells.SphereCell.internalAddToSim ( *self, sim* )

#### 7.19.3.3 def pySTDLM.Cells.SphereCell.internalModelCell ( *self, attr* )

#### 7.19.3.4 def pySTDLM.Cells.SphereCell.internalTranslateCell ( *self, point* )

### 7.19.4 Member Data Documentation

#### 7.19.4.1 pySTDLM.Cells.SphereCell.maxs

#### 7.19.4.2 pySTDLM.Cells.SphereCell.membraneThickness

#### 7.19.4.3 pySTDLM.Cells.SphereCell.mins

#### 7.19.4.4 pySTDLM.Cells.SphereCell.name

#### 7.19.4.5 pySTDLM.Cells.SphereCell.origin

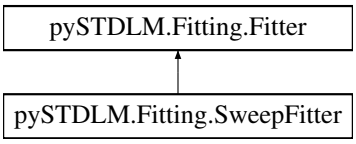
#### 7.19.4.6 pySTDLM.Cells.SphereCell.radius

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

- Cells.py

## 7.20 pySTDLM.Fitting.SweepFitter Class Reference

Inheritance diagram for pySTDLM.Fitting.SweepFitter:




### Public Member Functions

- def __init__
- def setParameterDivisions
- def performFit
- def saveFit

### Public Attributes

- divisions
- hasFit

### 7.20.1 Constructor & Destructor Documentation

#### 7.20.1.1 def pySTDLM.Fitting.SweepFitter.__init__ ( *self* )

### 7.20.2 Member Function Documentation

#### 7.20.2.1 def pySTDLM.Fitting.SweepFitter.performFit ( *self, numberProcElements* )

#### 7.20.2.2 def pySTDLM.Fitting.SweepFitter.saveFit ( *self, filebase, plot* )

#### 7.20.2.3 def pySTDLM.Fitting.SweepFitter.setParameterDivisions ( *self, parameter, divisions, scale* )

### 7.20.3 Member Data Documentation

#### 7.20.3.1 pySTDLM.Fitting.SweepFitter.divisions

#### 7.20.3.2 pySTDLM.Fitting.SweepFitter.hasFit

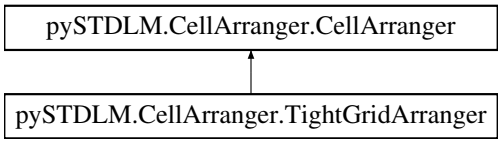
The documentation for this class was generated from the following file:

- Fitting.py

## 7.21 pySTDLM.CellArranger.TightGridArranger Class Reference

An arranger that orients cells in square grid.

Inheritance diagram for pySTDLM.CellArranger.TightGridArranger:



### Public Member Functions

- def __init__
- def intPackVolume

### Additional Inherited Members

#### 7.21.1 Detailed Description

An arranger that orients cells in square grid.

#### 7.21.2 Constructor & Destructor Documentation

##### 7.21.2.1 def pySTDLM.CellArranger.TightGridArranger.__init__ ( *self, cellType, cellAttributes, arrangerAttributes* )

#### 7.21.3 Member Function Documentation

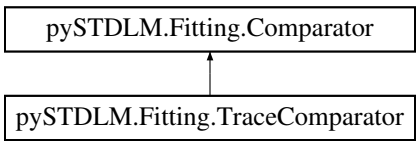
##### 7.21.3.1 def pySTDLM.CellArranger.TightGridArranger.intPackVolume ( *self, sim, mins, maxs* )

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

- CellArranger.py

## 7.22 pySTDLM.Fitting.TraceComparator Class Reference

Inheritance diagram for pySTDLM.Fitting.TraceComparator:




### Public Member Functions

- def __init__
- def computeMetric

### Public Attributes

- comparator
- times
- data

#### 7.22.1 Constructor & Destructor Documentation

##### 7.22.1.1 def pySTDLM.Fitting.TraceComparator.__init__ ( *self, comp, times, data* )

#### 7.22.2 Member Function Documentation

#### 7.22.2.1 def pySTDLM.Fitting.TraceComparator.computeMetric ( *self, fileHandle, time* = None, *data* = None )

### 7.22.3 Member Data Documentation

#### 7.22.3.1 pySTDLM.Fitting.TraceComparator.comparator

#### 7.22.3.2 pySTDLM.Fitting.TraceComparator.data

#### 7.22.3.3 pySTDLM.Fitting.TraceComparator.times

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

- Fitting.py

# Chapter 8

# File Documentation

## 8.1 __init__.py File Reference

### Namespaces

- pyLM

  *The problem solving environement which interfaces to Lattice Microbes simulation software.*

### Variables

- list pyLM.__all__ = ['LMLogger', 'CME', 'RDME', 'units', 'ipyInterface']

## 8.2 __init__.py File Reference

### Namespaces

- pySTDLM

  *A standard library of functionality that is commonly used in biological simulations.*

### Variables

- list pySTDLM.__all__ = ['StandardReactions', 'StandardReactionSystems', 'StandardCells', 'Cells', 'CellArranger', 'Distributions', 'PostProcessing', 'NetworkVisualization']

## 8.3 CellArranger.py File Reference

### Classes

- class pySTDLM.CellArranger.CellArranger

  *A base class for laying out cells in a 3D space.*
- class pySTDLM.CellArranger.RandomArranger

  *An arranger that randomly orients a number of cells in a regin until either a count or a volume fraction target is met.*
- class pySTDLM.CellArranger.RandomSphericalArranger

  *An arranger that randomly orients a number of cells in a sphere inside a regin until either a count or a volume fraction target is met.*
- class pySTDLM.CellArranger.TightGridArranger

*An arranger that orients cells in square grid.*

- class pySTDLM.CellArranger.LooseGridArranger

  *An arranger that orients cells in square grid with padding in between.*

### Namespaces

- pySTDLM.CellArranger

## 8.4 Cells.py File Reference

### Classes

- class pySTDLM.Cells.CellShape

  *Base class for a particular cell type for use in CellArranger (and later more)*
- class pySTDLM.Cells.SphereCell

  *A representation for a spherical cell.*
- class pySTDLM.Cells.BoxCell

  *A representation for a spherical cell.*
- class pySTDLM.Cells.CapsuleCell

  *A representation for a spherical cell.*
- class pySTDLM.Cells.CapsuleShellCell

  *A representation for a spherical cell.*

### Namespaces

- pySTDLM.Cells

## 8.5 CME.py File Reference

### Classes

- class pyLM.CME.CMESimulation

  *A CME simulation that contains reactions and species.*

### Namespaces

- pyLM.CME

### Functions

- def pyLM.CME.tqdm

## 8.6 Distributions.py File Reference

### Classes

- class pySTDLM.Distributions.ParticleGenerator

  *A particle generator that adds particles to a particular region of a simulation.*

### Namespaces

- pySTDLM.Distributions

## 8.7 Fitting.py File Reference

### Classes

- class pySTDLM.Fitting.Comparator
- class pySTDLM.Fitting.MomentComparator
- class pySTDLM.Fitting.TraceComparator
- class pySTDLM.Fitting.Fitter
- class pySTDLM.Fitting.SweepFitter
- class pySTDLM.Fitting.SimulatedAnnealingFitter

### Namespaces

- pySTDLM.Fitting

### Functions

- def pySTDLM.Fitting.DistributionComparator

### Variables

- pySTDLM.Fitting.counts
- pySTDLM.Fitting.binwidth
- pySTDLM.Fitting.measure

## 8.8 ipyInterface.py File Reference

### Namespaces

- pyLM.ipyInterface

### Functions

- def pyLM.ipyInterface.getReactionString

  *Write a reaction to a string with the correct units.*
- def pyLM.ipyInterface.writeTable

  *Write an HTML formatted table to a string.*
- def pyLM.ipyInterface.displayRDMESetup

  *Display an RDME initial conditions in a Jupyter Notebook using ipywidgets.*

## 8.9 LMLogger.py File Reference

### Classes

- class pyLM.LMLogger.NullHandler

### Namespaces

- pyLM.LMLogger

### Functions

- def pyLM.LMLogger.setLMLoggerLevel

  *Set the level of the logger for the application.*
- def pyLM.LMLogger.setLMLogFile

  *Set up file handler to print log to file.*
- def pyLM.LMLogger.setLMLogConsole

  *Set the logger to write to the console as the code is working.*

### Variables

- tuple pyLM.LMLogger.LMLogger = logging.getLogger('LMLogger')
- tuple pyLM.LMLogger.nullHandlerLM = logging.NullHandler()
- tuple pyLM.LMLogger.LMformatter = logging.Formatter('%(asctime)s: %(levelname)s: %(message)s')

## 8.10 NetworkVisualization.py File Reference

### Namespaces

- pySTDLM.NetworkVisualization

### Functions

- def pySTDLM.NetworkVisualization.plotReactionNetwork

  *Plot the reaction scheme as a network.*
- def pySTDLM.NetworkVisualization.plotCMEReactionNetwork

  *Plot the reaction scheme as a network.*
- def pySTDLM.NetworkVisualization.plotRDMEReactionNetwork

  *Plot the reaction scheme as a network.*
- def pySTDLM.NetworkVisualization.plotCMEDynamicReactionNetwork

  *Plot the dynamics of the species on the network into a dynamic graph file (extension: .gexf)*

### Variables

- pySTDLM.NetworkVisualization.hasGexf = True

## 8.11 OnTheFly.py File Reference

### Namespaces

- pySTDLM.OnTheFly

## 8.12 peekFile.py File Reference

### Namespaces

- peekFile

### Variables

- tuple peekFile.parser = argparse.ArgumentParser(description="Print info on simulation file")
- tuple peekFile.args = parser.parse_args()
- tuple peekFile.tf = float(hdf['Parameters'].attrs['maxTime'])
- list peekFile.t = sim['LatticeTimes']
- tuple peekFile.n = len(sim['Lattice'])

## 8.13 PostProcessing.py File Reference

### Namespaces

- pySTDLM.PostProcessing

### Functions

- def pySTDLM.PostProcessing.openLMFile

  *Open a Lattice Microbes File for reading.*
- def pySTDLM.PostProcessing.closeLMFile

  *Close a Lattice Microbes File.*
- def pySTDLM.PostProcessing.showTraceFromFile

  *Show species trace from a particular replicate.*
- def pySTDLM.PostProcessing.plotTraceFromFile

  *Plot species from an output file.*
- def pySTDLM.PostProcessing.showTrace

  *Show a specific species trace.*
- def pySTDLM.PostProcessing.plotTrace

  *Plot a specific species trace.*
- def pySTDLM.PostProcessing.showAvgVarFromFile

  *Show species from an output file.*
- def pySTDLM.PostProcessing.plotAvgVarFromFile

  *Plot species from an output file.*
- def pySTDLM.PostProcessing.showAvgVar

  *Show a specific species average over time and variance.*
- def pySTDLM.PostProcessing.plotAvgVar

  *Plot a specific species average over time and variance.*
- def pySTDLM.PostProcessing.getOccupancyKymograph

  *Compute the specie density(occupancy) among a slice of the simulation domain as a function over time for the given direciton.*
- def pySTDLM.PostProcessing.plotOccupancyKymograph

  *Plot a specific specie density(occupancy) among a slice of the simulation domain as a function of a direction over time.*
- def pySTDLM.PostProcessing.showOccupancyKymograph

  *Show a specific specie density(occupancy) among a slice of the simulation domain as a function of a direction over time.*

- def pySTDLM.PostProcessing.plotPhaseSpace

  *Plot the 2D or 3D phase space associated with the given species over the replicates indicated.*
- def pySTDLM.PostProcessing.getTimesteps

  *Extract the timestep times.*
- def pySTDLM.PostProcessing.getSpecieTrace

  *Extract data for a particular specie for the specified replicate.*
- def pySTDLM.PostProcessing.getAvgVarTrace

  *Get the average and variance of the specie trace over time.*
- def pySTDLM.PostProcessing.getHistogram

  *Get the histogram for a specie.*
- def pySTDLM.PostProcessing.getPhaseSpace

  *Get the nD phase space associated with the traces of species.*

## 8.14 RDME.py File Reference

### Classes

- class pyLM.RDME.CMESimulation

  *A CME simulation that contains reactions and species.*
- class pyLM.RDME.RDMERegion

  *A class that represents a type region of an RDME simulation.*
- class pyLM.RDME.RDMESimulation

  *A class that contains all regions, reactions, diffusions and rules for a RDME simulation.*

### Namespaces

- pyLM.RDME

### Functions

- def pyLM.RDME.tqdm

## 8.15 SBMLReader.py File Reference

### Namespaces

- pySTDLM.SBMLReader

### Functions

- def pySTDLM.SBMLReader.importSBMLModel

  *Helper Functions ##.*
- def pySTDLM.SBMLReader.importSBMLModelL3V1
- def pySTDLM.SBMLReader.getRate
- def pySTDLM.SBMLReader.readSBMLtoCME

  *Read an SBML file for the reaction model for a CME simulation.*
- def pySTDLM.SBMLReader.readSBMLtoRDME

  *Read an SBML file for the reaction model for a RDME simulation.*
- def pySTDLM.SBMLReader.readSBMLtoRegion

  *Read an SBML file for the reaction model into a specific region of the RDME simulation.*

## 8.16 StandardCells.py File Reference

### Namespaces

- pySTDLM.StandardCells

### Functions

- def pySTDLM.StandardCells.buildEColiCell

  *Build a standard E coli cell 0.5 microns across and 2 microns long.*
- def pySTDLM.StandardCells.buildDividingEColiCell

  *Build a standard E coli cell 0.5 microns across and 4 microns long.*
- def pySTDLM.StandardCells.buildSphericalEColiCell

  *Build a Spherical E coli cell, one that lacks the gene for elongation.*
- def pySTDLM.StandardCells.buildFilamentousEColiCell

  *Build a long filamentous E coli cell where the user can specify the length.*
- def pySTDLM.StandardCells.packFastGrowingEcoli

  *Pack a cell with the protein distribution for fast growing E coli.*
- def pySTDLM.StandardCells.packSlowGrowingEcoli

  *Pack a cell with the protein distribution for fast growing E coli.*

## 8.17 StandardReactions.py File Reference

### Namespaces

- pySTDLM.StandardReactions

### Functions

- def pySTDLM.StandardReactions.addMembraneTransporter

  *Adds a membrane transport system to an RDME system.*
- def pySTDLM.StandardReactions.addPassiveTransport

  *Adds a passive transport system to an RDME system.*
- def pySTDLM.StandardReactions.addMichaelisMenten

  *Adds a Michaelis Menten Reaction.*
- def pySTDLM.StandardReactions.addReversibleMichaelisMenten

  *Add a Reversible Michaelis Menten Reaction.*
- def pySTDLM.StandardReactions.createExpressionModel

  *Create a set of gene/mRNA/protein reactions based on a genebank file and a set of rates assuming constitutive expression.*

### Variables

- pySTDLM.StandardReactions.hasBio = True

## 8.18 StandardReactionSystems.py File Reference

### Namespaces

- pySTDLM.StandardReactionSystems

### Functions

- def pySTDLM.StandardReactionSystems.addMinDESystem

  *Adds the MinDE system in Ecoli as a standard reaction network.*
- def pySTDLM.StandardReactionSystems.addLacTwoStateSystem

  *Adds the Lac switch reaction with two states of the DNA.*
- def pySTDLM.StandardReactionSystems.addPTSPathway

  *Adds the Phosphoenolpyruvate-depenedent phosphotransferase reaction system.*

## 8.19 units.py File Reference

### Namespaces

- pyLM.units

### Functions

- def pyLM.units.angstrom

  *Length Wrapper Functions #.*
- def pyLM.units.nm

  *Returns a representation of a number in nanometers.*
- def pyLM.units.micron

  *Returns a representation of a number in micrometers.*
- def pyLM.units.mm

  *Returns a representation of a number in millimeter.*
- def pyLM.units.cm

  *Returns a representation of a number in centimeter.*
- def pyLM.units.ns

  *Time Wrapper Functions #.*
- def pyLM.units.microsecond

  *Returns a representation of a number in microsecond.*
- def pyLM.units.ms

  *Returns a representation of a number in millisecond.*
- def pyLM.units.second

  *Returns seconds in seconds.*
- def pyLM.units.minute

  *Returns a representation of a number in minutes.*
- def pyLM.units.hr

  *Returns a representation of a number in hours.*
- def pyLM.units.day

  *Returns a representation of a number in days.*

# Index