Lysis

Brittany Bannish

bbannish@uco.edu> & Brad Paynter

 bpaynter

CONTENTS:

	lysis 1.1 lysis package	1
2	Indices and tables	19
Ру	ython Module Index	21
In	ndex	23

CHAPTER

ONE

LYSIS

1.1 lysis package

1.1.1 Subpackages

lysis.util package

Submodules

lysis.util.constants module

Bases: object

```
class lysis.util.constants.BoundaryCondition(value)
     Bases: Enum
     An enumeration.
    CONTINUING = 2
    PERIODIC = 1
    REFLECTING = 0
class lysis.util.constants.BoundaryDirection(value)
     Bases: IntEnum
     An enumeration.
    BACK = 5
    BOTTOM = 1
    FRONT = 4
    LEFT = 2
    RIGHT = 3
    TOP = 0
class lysis.util.constants.Const
```

```
class lysis.util.constants.ExpComponent(value)
     Bases: Flag
     An enumeration.
     ALL = 15
    MACRO = 4
    MACRO_POSTPROCESSING = 8
    MICRO = 1
    MICRO_POSTPROCESSING = 2
    NONE = 0
class lysis.util.constants.FiberDirection(value)
     Bases: Enum
     An enumeration.
    DOWN = -1
    IN = -3
    LEFT = 2
    OUT = 3
    RIGHT = -2
    UP = 1
class lysis.util.constants.Neighbors
     Bases: object
class lysis.util.constants.RandomDraw(value)
     Bases: IntEnum
     An enumeration.
    BINDING_TIME_WHEN_MOVING = 1
    BINDING_TIME_WHEN_UNBINDING = 0
    CONFLICT_RESOLUTION = 6
    LYSIS_TIME = 5
    MICRO\_UNBIND = 2
    MOVE = 3
    RESTRICTED_MOVE = 7
    UNBINDING\_TIME = 4
```

lysis.util.datastore module

```
class lysis.util.datastore.DataStatus(value)
     Bases: Flag
     An enumeration.
     INITIALIZED = 1
     LOADED = 2
     NONE = 0
     SAVED = 4
class lysis.util.datastore.DataStore(path: str | bytes | PathLike, filenames: Mapping | None = None)
     Bases: object
     append(key: AnyStr, value: Any, axis: int | None = None)
     delete(key: AnyStr)
     load_from_disk(key: AnyStr)
     new(key: AnyStr, filename: AnyStr)
     overwrite(key: AnyStr, value: Any)
     save_to_disk(key: AnyStr)
     status(key: AnyStr) \rightarrow DataStatus
     to\_dict() \rightarrow Mapping
lysis.util.datastore.identity(x: Any)
```

lysis.util.kiss module

Minimal skeleton for using the Marsaglia KISS generator

Wrapper for C module.

Example Usage:

Initialize

```
>>> kiss = KissRandomGenerator()
>>> kiss.seed(123)
```

• Generate random integer 0..(2^32)-1

```
>>> kiss.kiss32()
```

• Generate random U[0,1]

```
>>> kiss.random()
```

```
class lysis.util.kiss.KissRandomGenerator(seed: int | None = None)
```

Bases: object

A Pseudo-random Number Generator.

Wrapper class for kiss.so C module

```
getstate() \rightarrow Tuple[int, int, int, int]
```

Returns the current state of the Random Number Generator.

```
integers(bottom: int, top: int | None = None, size: int | None = None) \rightarrow int | ndarray
```

```
kiss32() \rightarrow int
```

Returns the next 32-bit unsigned integer from the pseudo-random number stream. This is distributed uniformly from 0 through $(2^32)-1$.

```
mscw() \rightarrow int
```

Returns a pseudo-random 32-bit unsigned integer (i.e., [0..(2^32)-1]). This is based on the system clock and NOT the current seed or state.

```
random(size: int \mid None = None) \rightarrow float | ndarray
```

Returns the next double-precision random number from the pseudo-random number stream. This is distributed uniformly from zero through one.

```
seed(seed: int | None = None)
```

Sets the seed for the Random Number Generator.

Parameters

seed – The seed.

Note: While Python will accept any 64-bit integer (-9,223,372,036,854,775,806 through 9,223,372,036,854,775,807), this will be converted to an unsigned 32-bit integer (0 through 4,294,967,295) when passed to the underlying C code with unpredictable results.

```
setstate(state: Tuple[int, int, int, int])
```

Sets the state of the Random Number Generator.

Parameters

state – A tuple of four integers.

Note: While Python will accept any 64-bit integer (-9,223,372,036,854,775,806 through 9,223,372,036,854,775,807), these will be converted to unsigned 32-bit integers (0 through 4,294,967,295) when passed to the underlying C code with unpredictable results.

lysis.util.parameters module

Code for holding, storing, and reading information about an experiment

This module gives a uniform way to handle the data and parameters of a given experiment. It contains classes to house these and make them accessible to the rest of the code. It also handles the storing and reading of parameters and data to/from disk.

Typical usage example:

```
>>> # Create a new experiment
>>> exp = Experiment('path/to/data')
>>> param = {'override_parameter': 2.54, 'another_new_parameter': 32}
>>> exp.initialize_macro_param(param)

(continues on next page)
```

(continued from previous page)

```
>>> exp.to_file()
>>> # Load an existing experiment
>>> exp = Experiment('path/to/data', '2022_12_27_1100')
>>> exp.read_file()
>>> # Access a parameter
>>> exp.macro_params.pore_size
>>> # Access data
>>> exp.data.lysis_time[4][18]
```

Bases: object

Houses all information about a given experimental run.

This object contains:

- · Data location
- Experiment parameters
- · Experiment data

It includes methods for

- Initializing with default parameters
- · Reading parameters from disk
- · Saving parameters to disk
- Reading input data from disk
- · Writing result data to disk

Parameters

- data_root The path of the folder containing datasets
- **experiment_code** The code number of the experiment. This will be the name of the folder containing the data specific to this experiment. This should be a date and time in 'YYYY-MM-DD-hhmm' format If no code is given, one will be generated from the current date and time.

experiment_code

The code number of the experiment.

```
Type str
```

os_path

The path to the folder containing this experiment's data

```
Type str
```

macro_params

A dictionary of

Type

DataClass

Raises

RuntimeError – An invalid data folder was given.

$initialize_macro_param(params: dict[str, Any] | None = None) \rightarrow None$

Creates the parameters for the Macroscale model.

Parameters are set to the default values unless new values are passed in the params dictionary.

This method is essentially a wrapper for the MacroParameters constructor.

Parameters

params -

A dictionary of parameters that differ from the default values.

For example,

```
>>> {'binding_rate': 10, 'pore_size': 3,}
```

$initialize_micro_param(params: Mapping[str, Any] \mid None = None) \rightarrow None$

Creates the parameters for the Microscale model.

Parameters are set to the default values unless new values are passed in the params dictionary.

This method is essentially a wrapper for the MicroParameters constructor.

Parameters

params -

A dictionary of parameters that differ from the default values.

For example,

```
>>> {'binding_rate': 10, 'pore_size': 3,}
```

$read_file() \rightarrow None$

Load the experiment parameters from disk.

Raises

RuntimeError – No parameter file is available for this experiment.

$to_dict() \rightarrow dict$

Returns the internally stored data as a dictionary.

Does not include system-specific information like paths.

$to_file() \rightarrow None$

Stores the experiment parameters to disk.

Creates or overwrites the params.json file in the experiment's data folder. This file will contain the current experiment parameters (including any micro- and macroscale parameters) in JSON format.

```
class lysis.util.parameters.MacroParameters(pore_size: ~pint.registry.Quantity = <Quantity(1.0135, 'micrometer')>, diffusion_coeff: ~pint.registry.Quantity = <Quantity(5e-07, 'centimeter ** 2 / second')>, forced_unbind: float = 0.0852, average_bound_time: ~pint.registry.Quantity = <Quantity(27.8, 'second')>, cols: int = 93, rows: int = 121, empty_rows: int = 28, total_molecules: int = 43074, moving_probability: float = 0.2, simulations: int = 10, total_time: ~pint.registry.Quantity = <Quantity(20, 'minute')>, seed: int = 0, save_interval: ~pint.registry.Quantity = <Quantity(10, 'second')>, macro_version: str = 'diffuse_into_and_along', log_lvl: int = 30, duplicate_fortran: bool = False, processing_library: str = 'numpy')
```

Bases: object

Contains parameters for the Macroscale model.

Parameters can be accessed as attributes. Independent parameters should only be set at initialization. Dependent parameters should never be set manually, but are automatically calculated by internal code.

Should only be used inside an Experiment object.

Example

```
>>> # Initialize using the default values
>>> macro_params_default = MacroParameters()
>>> # Get parameter value
>>> macro_params_default.pore_size
1.0135e-4
>>> # Initialize overriding some default values
>>> p = {'binding_rate': 10, 'pore_size': 3}
>>> macro_params_override = MacroParameters(**p)
```

```
average_bound_time: Quantity = <Quantity(27.8, 'second')>
```

This is the average time a tPA molecule stays bound to fibrin. For now I'm using 27.8 to be 1/0.036, the value in the absence of PLG.

```
Units
seconds

Fortran
avgwait = 1/koff

cols: int = 93
The number of lattice nodes in each (horizontal) row
Units
None
Fortran
N

diffusion_coeff: Quantity = <Quantity(5e-07, 'centimeter ** 2 / second')>
Diffusion coefficient
```

```
Units
```

cm²/s

Fortran

Diff

duplicate_fortran: bool = False

Whether the Python code should follow the Fortran code step-by-step. Theoretically, with this set to "True", both sets of code will produce the exact same output. This will impact performance negatively. This currently does nothing.

Units

None

Fortran

None

empty_edges: int

The number of edges without fibrin. Also the 1-D index of the last edge without fibrin when 1-indexing

This is probably unnecessary when using a 2-D data structure, but is kept for historical reasons.

Units

None

Fortran

enoFB

empty_rows: int = 28

The number of fibrin-free rows at the top of the grid.

Equivalent to 'first_fiber_row', which is the 1st node in vertical direction containing fibers. So if first_fiber_row = 10, then rows 0-9 have no fibers, there's one more row of fiber-free planar vertical edges, and then the row with index 'first_fiber_row' (e.g. 11th) is a full row of fibers.

Units

None

Fortran

Ffree-1

fiber_rows: int

The number of rows containing fibrin

Units

None

Fortran

Fhat

forced_unbind: float = 0.0852

Fraction of times tPA was forced to unbind in microscale model.

Units

None

Fortran

frac_forced

static fortran_names()

Returns a dictionary whose keys are the names of all parameters (both micro- and macroscale) that have equivalents in Fortran. The values in the dictionary are the names of the equivalent Fortran variable names. These values are parsed from the docstrings in this file.

```
full_row: int

Edges in a full row of nodes

Units

None

Fortran

None
```

input_data: List[str]

The data (from the Microscale model) required to run the Macroscale model.

```
log_lvl: int = 30
```

How much debugging information to write out to the console

Units

None

Fortran

None

```
macro_version: str = 'diffuse_into_and_along'
```

A string identifying which version of the Macroscale model is being run. This string was included in data filenames stored by the Fortran code.

```
moving_probability: float = 0.2
```

The probability of moving.

Make sure it is small enough that we've converged.

```
Units
```

None

Fortran

q

number_of_saves: int

The number of times data will be saved from the model.

Units

None

Fortran

nplt

output_data: List[str]

The data output by the Macroscale model.

pore_size: Quantity = <Quantity(1.0135, 'micrometer')>

Pore size (distance between fibers/nodes)

Units

centimeters

Fortran

delx

```
static print_default_values() → str
     Returns the default parameters for the Macroscale model.
processing_library: str = 'numpy'
     Which library the macroscale model should use for processing. Options include
       • 'numpy'
       · 'cupy'
         Units
            None
         Fortran
            None
rows:
        int = 121
     The number of lattice nodes in each (vertical) column
         Units
            None
         Fortran
            F
save_interval: Quantity = <Quantity(10, 'second')>
     How often to record data from the model.
         Units
             sec
         Fortran
            save_interval
seed: int = 0
     Seed for the random number generator
         Units
            None
         Fortran
            seed
simulations: int = 10
     The number of independent simulations to be run
         Units
            None
         Fortran
             stats
state: Tuple[int, int, int, int]
     State for the random number generator.
         Units
            None
         Fortran
            state
```

```
time_step: float
    The length of one timestep.
        Units
            seconds
        Fortran
            tstep
total_edges: int
    The total number of edges in the model
        Units
            None
        Fortran
            num
total_fibers: int
    The total number of fibers in the model
        Units
            None
        Fortran
            None
total_molecules: int = 43074
    The total number of tPA molecules:
       • 43074 is Colin's [tPA]=0.6 nM
       • 86148 is Colin's [tPA]=1.2 nM
        Units
            None
        Fortran
            M
total_time: Quantity = <Quantity(20, 'minute')>
    Total running time for model.
        Units
            seconds
        Fortran
            tf
total_time_steps: int
    The total number of timesteps.
        Units
            None
        Fortran
            num_t
static units()
```

Returns a dictionary whose keys are the names of all parameters (both micro- and macroscale) that have units. The values in the dictionary are those units. These values are parsed from the docstrings in this file.

```
xz_row: int
    Number of all x- and z-edges in a row

Units
    None
    Fortran
    None

class lysis.util.parameters.MicroParameters(fibrinogen length: ~pint.registry.Quantity = <Quantity(45,</pre>
```

```
'nanometer')>, fibrinogen radius: ~pint.registry.Quantity
= < Quantity(1.2, 'nanometer')>, fiber_radius:
\simpint.registry.Quantity = <Quantity(36.35, 'nanometer')>,
diss_const_tPA_wPLG: ~pint.registry.Quantity =
< Quantity(0.02, 'micromolar')>, diss_const_tPA_woPLG:
\simpint.registry.Quantity = <Quantity(0.36, 'micromolar')>,
diss_const_PLG_intact: ~pint.registry.Quantity =
< Quantity(38, 'micromolar')>, diss_const_PLG_nicked:
\sim pint.registry.Quantity = < Quantity(2.2, 'micromolar')>,
bind_rate_tPA: \sim pint.registry.Quantity = < Quantity(0.1, '1)
/ micromolar / second')>, bind_rate_PLG:
\simpint.registry.Quantity = <Quantity(0.1, '1 / micromolar /
second')>, conc free PLG: ~pint.registry.Quantity =
< Quantity(2, 'micromolar')>, deg_rate_fibrin:
\simpint.registry.Quantity = <Quantity(5.0, '1 / second')>,
unbind_rate_PLi: ~pint.registry.Quantity =
<Quantity(57.6, '1 / second')>, activation rate PLG:
\simpint.registry.Quantity = <Quantity(0.1, '1 / second')>,
exposure_rate_binding_site: ~pint.registry.Quantity =
<Quantity(5.0, '1 / second')>, nodes_in_row: int = 7,
simulations: int = 50000, seed: int = 0, micro\_version: str
= 'micro_rates', log_lvl: int = 30)
```

Bases: object

This will contain the parameters for the Microscale model.

Parameters can be accessed as attributes. Independent parameters should only be set at initialization. Dependent parameters should never be set manually, but are automatically calculated by internal code.

Should only be used inside an Experiment object.

Example

```
>>> # Initialize using the default values
>>> micro_params_default = MicroParameters()
>>> # Get parameter value
>>> micro_params_default.fibrinogen_length
45 nanometers
>>> # Initialize overriding some default values
>>> p = {'fibrinogen_radius': "10 nm", 'fiber_radius': "3 um"}
>>> micro_params_override = MicroParameters(**p)
```

activation_rate_PLG: Quantity = <Quantity(0.1, '1 / second')>

The catalytic rate constant, $k_{\rm cat}^{\rm ap}$, for activation of PLG into PLI.

```
Units
             sec^-1
         Fortran
             kapcat
bind_rate_PLG: Quantity = <Quantity(0.1, '1 / micromolar / second')>
     The binding rate of PLG, k_{\rm PLG}^{\rm on}, to fibrin.
         Units
             (micromolar*sec)^-1
         Fortran
             kPLGon
bind_rate_tPA: Quantity = <Quantity(0.1, '1 / micromolar / second')>
     The binding rate of tPA, k_{\text{tPA}}^{\text{on}}, to fibrin.
         Units
             (micromolar*sec)^-1
         Fortran
             ktPAon
binding_sites: Quantity
     Concentration of binding sites.
             micromolar
         Fortran
conc_free_PLG: Quantity = <Quantity(2, 'micromolar')>
     The concentration of free plasminogen.
         Units
             micromolar
         Fortran
             freeplg
deg_rate_fibrin: Quantity = <Quantity(5.0, '1 / second')>
     The plasmin-mediated rate of fibrin degradation.
         Units
             sec^-1
         Fortran
             kdeg
diss_const_PLG_intact: Quantity = <Quantity(38, 'micromolar')>
     The dissociation constant of PLG, k_{\rm PLG}^{\cal D}, to intact fibrin.
         Units
             micromolar
         Fortran
             KdPLGintact
```

```
diss_const_PLG_nicked: Quantity = <Quantity(2.2, 'micromolar')>
     The dissociation constant of PLG, k_{\rm PLG}^D, to nicked fibrin.
         Units
             micromolar
         Fortran
             KdPLGnicked
diss_const_tPA_wPLG: Quantity = <Quantity(0.02, 'micromolar')>
     The dissociation constant of tPA, k_{\mathrm{tPA}}^{D}, to fibrin in the presence of PLG.
         Units
             micromolar
         Fortran
             KdtPAyesplg
diss_const_tPA_woPLG: Quantity = <Quantity(0.36, 'micromolar')>
     The dissociation constant of tPA, k_{tPA}^{D}, to fibrin in the absence of PLG.
         Units
             micromolar
         Fortran
             KdtPAnoplg
exposure_rate_binding_site: Quantity = <Quantity(5.0, '1 / second')>
     The catalytic rate constant, k_{\text{cat}}^{\text{n}}, for the PLi-mediated rate of exposure of new binding sites.
         Units
             sec^-1
         Fortran
             kncat
fiber_radius: Quantity = <Quantity(36.35, 'nanometer')>
     The radius of each fiber in the model.
         Units
             microns
         Fortran
             radius
fibrin_conc_per_fiber: Quantity
     The concentration of fibrin in each fiber
         Units
             micromolar
         Fortran
             None
fibrinogen_length: Quantity = <Quantity(45, 'nanometer')>
     The length of a fibronogen molecule.
         Units
             microns
         Fortran
             None
```

```
fibrinogen_radius: Quantity = <Quantity(1.2, 'nanometer')>
     The radius of a fibronogen molecule.
         Units
             microns
         Fortran
             None
static fortran_names()
     Returns a dictionary whose keys are the names of all parameters (both micro- and macroscale) that have
     equivalents in Fortran. The values in the dictionary are the names of the equivalent Fortran variable names.
     These values are parsed from the docstrings in this file.
log_lvl: int = 30
     How much debugging information to write out to the console
         Units
             None
         Fortran
             None
micro_version: str = 'micro_rates'
     A string identifying which version of the Microscale model is being run.
nodes_in_row: int = 7
     The number of protofibrils in one row of the lattice inside one fiber.
         Units
             None
         Fortran
             nodes
output_data: List[str]
     The data output by the Microscale model.
static print_default_values() → str
     Returns the default parameters for the Microscale model.
protein_per_fiber: Quantity
     The fraction of protein in each fiber (by volume?)
         Units
             %
         Fortran
             None
protofibril_radius: Quantity
     The radius of a protofibril.
         Units
             microns
         Fortran
             None
```

```
seed: int = 0
     Seed for the random number generator
         Units
             None
         Fortran
             seed
simulations: int = 50000
     The number of independent trials run in the microscale model.
             None
         Fortran
             runs
The proportion of doublets that need to be degraded before the fiber snaps.
         Units
             None
         Fortran
             snap\_proportion
unbind_rate_PLG_intact: Quantity
     The unbinding rate of PLG, k_{\rm PLG}^{\rm off}, from intact fibrin.
         Units
             sec^-1
         Fortran
             kplgoff
unbind_rate_PLG_nicked: Quantity
     The unbinding rate of PLG, k_{\rm PLG}^{\rm off}, from nicked fibrin.
         Units
             sec^-1
         Fortran
             kplgoffnick
unbind_rate_PLi: Quantity = <Quantity(57.6, '1 / second')>
     The unbinding rate of PLi, k_{\mathrm{PLi}}^{\mathrm{off}}, from fibrin.
         Units
             sec^-1
         Fortran
             kplioff
unbind_rate_tPA_wPLG: Quantity
     The unbinding rate of tPA, k_{\rm tPA}^{\rm off}, from fibrin in the presence of PLG.
         Units
             sec^-1
         Fortran
             kaoff12
```

unbind_rate_tPA_woPLG: Quantity

The unbinding rate of tPA, $k_{\text{tPA}}^{\text{off}}$, from fibrin in the absence of PLG.

```
Units
sec^-1
Fortran
kaoff10
```

static units()

Returns a dictionary whose keys are the names of all parameters (both micro- and macroscale) that have units. The values in the dictionary are those units. These values are parsed from the docstrings in this file.

lysis.util.util module

```
lysis.util.util.dict_to_formatted_str(d: Mapping[AnyStr, Any]) → str
```

Converts a dictionary into a formatted, JSON-like string.

Align keys and values, including the alignment of sub-dicts.

e.g.,

Parameters

d (*dict*) – A dictionary with string-like keys.

Module contents

1.1.2 Submodules

1.1.3 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

```
lysis, 17
lysis.util, 17
lysis.util.constants, 1
lysis.util.datastore, 3
lysis.util.kiss, 3
lysis.util.parameters, 4
lysis.util.util, 17
```

22 Python Module Index

INDEX

A	DataStore (class in lysis.util.datastore), 3
activation_rate_PLG (ly-	deg_rate_fibrin (ly-
sis.util.parameters.MicroParameters attribute),	sis.util.parameters.MicroParameters attribute), 13
ALL (lysis.util.constants.ExpComponent attribute), 2	<pre>delete() (lysis.util.datastore.DataStore method), 3</pre>
append() (lysis.util.datastore.DataStore method), 3	<pre>dict_to_formatted_str() (in module lysis.util.util),</pre>
average_bound_time (ly-	17
sis.util.parameters.MacroParameters attribute),	diffusion_coeff (ly-
7	sis.util.parameters.MacroParameters attribute),
В	diss_const_PLG_intact (ly-
	diss_const_PLG_intact (ly- sis.util.parameters.MicroParameters attribute),
BACK (lysis.util.constants.BoundaryDirection attribute), 1	13
bind_rate_PLG (lysis.util.parameters.MicroParameters attribute), 13	diss_const_PLG_nicked (ly-
bind_rate_tPA (lysis.util.parameters.MicroParameters	sis.util.parameters.MicroParameters attribute),
attribute), 13	13
binding_sites (lysis.util.parameters.MicroParameters	${\tt diss_const_tPA_woPLG} \hspace{1.5cm} (\mathit{ly-}$
attribute), 13	sis.util.parameters.MicroParameters attribute),
BINDING_TIME_WHEN_MOVING (ly-	14
sis.util.constants.RandomDraw $attribute),$	diss_const_tPA_wPLG (ly-
2	sis.util.parameters.MicroParameters attribute),
BINDING_TIME_WHEN_UNBINDING (ly-	DOWN (lysis.util.constants.FiberDirection attribute), 2
sis.util.constants.RandomDraw attribute),	duplicate_fortran (ly-
BOTTOM (lvsis.util.constants.BoundaryDirection at-	sis.util.parameters.MacroParameters attribute),
BOTTOM (lysis.util.constants.BoundaryDirection attribute), 1	8
BoundaryCondition (class in lysis.util.constants), 1	E
BoundaryDirection (class in lysis.util.constants), 1	_
	empty_edges (lysis.util.parameters.MacroParameters
C	attribute), 8 empty_rows (lysis.util.parameters.MacroParameters at-
cols (lysis.util.parameters.MacroParameters attribute), 7	tribute), 8
${\tt conc_free_PLG}\ ({\it lysis.util.parameters.MicroParameters}$	ExpComponent (class in lysis.util.constants), 1
attribute), 13	Experiment (class in lysis.util.parameters), 5
CONFLICT_RESOLUTION (ly-	<pre>experiment_code (lysis.util.parameters.Experiment at-</pre>
sis.util.constants.RandomDraw attribute),	tribute), 5
Const (class in lysis util constants) 1	exposure_rate_binding_site (ly-
Const (class in lysis.util.constants), 1 CONTINUING (lysis.util.constants.BoundaryCondition at-	sis.util.parameters.MicroParameters attribute),
tribute), 1	14
_	F
D	fiber_radius (lysis.util.parameters.MicroParameters
DataStatus (class in lysis.util.datastore), 3	attribute), 14

$\label{linear_fiber_rows} \emph{fiber_rows} \emph{ (lysis.util.parameters.MacroParameters attribute)}, 8$	<pre>load_from_disk()</pre>
FiberDirection (class in lysis.util.constants), 2 fibrin_conc_per_fiber (ly- sis.util.parameters.MicroParameters attribute),	LOADED (lysis.util.datastore.DataStatus attribute), 3 log_lv1 (lysis.util.parameters.MacroParameters attribute), 9
14	log_lvl (lysis.util.parameters.MicroParameters at-
fibrinogen_length (ly-	tribute), 15
sis.util.parameters.MicroParameters attribute), 14	lysis module, 17
fibrinogen_radius (ly-	lysis.util
sis.util.parameters.MicroParameters attribute),	module, 17
14	lysis.util.constants
forced_unbind (lysis.util.parameters.MacroParameters	module, 1
attribute), 8	lysis.util.datastore
fortran_names() (ly-	module, 3
sis.util.parameters.MacroParameters static	lysis.util.kiss
method), 8	module, 3
fortran_names() (ly-	lysis.util.parameters
sis.util.parameters.MicroParameters static	module, 4
method), 15	lysis.util.util
FRONT (lysis.util.constants.BoundaryDirection attribute),	module, 17
1	LYSIS_TIME (lysis.util.constants.RandomDraw at-
full_row (lysis.util.parameters.MacroParameters	tribute), 2
attribute), 9	
	M
G	MACRO (lysis.util.constants.ExpComponent attribute), 2
<pre>getstate() (lysis.util.kiss.KissRandomGenerator</pre>	macro_params (lysis.util.parameters.Experiment at- tribute), 5
	MACRO_POSTPROCESSING (ly-
	sis.util.constants.ExpComponent attribute),
<pre>identity() (in module lysis.util.datastore), 3</pre>	2
IN (lysis.util.constants.FiberDirection attribute), 2	macro_version (lysis.util.parameters.MacroParameters
<pre>initialize_macro_param() (ly-</pre>	attribute), 9
sis.util.parameters.Experiment method),	MacroParameters (class in lysis.util.parameters), 6
6	MICRO (lysis.util.constants.ExpComponent attribute), 2
initialize_micro_param() (ly-	MICRO_POSTPROCESSING (ly-
sis.util.parameters.Experiment method), 6	sis.util.constants.ExpComponent attribute), 2
INITIALIZED (lysis.util.datastore.DataStatus attribute), 3	MICRO_UNBIND (lysis.util.constants.RandomDraw attribute), 2
<pre>input_data (lysis.util.parameters.MacroParameters at- tribute), 9</pre>	micro_version (lysis.util.parameters.MicroParameters attribute), 15
<pre>integers() (lysis.util.kiss.KissRandomGenerator method), 4</pre>	MicroParameters (class in lysis.util.parameters), 12 module
	lysis, 17
K	lysis.util,17
kiss32() (lysis.util.kiss.KissRandomGenerator method),	lysis.util.constants, 1
4	lysis.util.datastore, 3
KissRandomGenerator (class in lysis.util.kiss), 3	lysis.util.kiss,3
_	lysis.util.parameters, 4
L	lysis.util.util, 17
LEFT (lysis.util.constants.BoundaryDirection attribute), 1	MOVE (lysis.util.constants.RandomDraw attribute), 2
LEFT (lysis.util.constants.FiberDirection attribute), 2	moving_probability (ly-
· · · · · · · · · · · · · · · · · · ·	sis.util.parameters.MacroParameters attribute).

24 Index

9	RIGHT (lysis.util.constants.BoundaryDirection attribute),	
mscw() (lysis.util.kiss.KissRandomGenerator method), 4	RIGHT (lysis.util.constants.FiberDirection attribute), 2	
N	rows (lysis.util.parameters.MacroParameters attribute),	
Neighbors (class in lysis.util.constants), 2 new() (lysis.util.datastore.DataStore method), 3 nodes_in_row (lysis.util.parameters.MicroParameters	S save_interval (lysis.util.parameters.MacroParameters attribute), 10 save_to_disk() (lysis.util.datastore.DataStore method), 3 SAVED (lysis.util.datastore.DataStatus attribute), 3 seed (lysis.util.parameters.MacroParameters attribute),	
0	seed (lysis.util.parameters.MicroParameters attribute),	
os_path (lysis.util.parameters.Experiment attribute), 5 OUT (lysis.util.constants.FiberDirection attribute), 2 output_data (lysis.util.parameters.MacroParameters attribute), 9	seed() (lysis.util.kiss.KissRandomGenerator method), 4 setstate() (lysis.util.kiss.KissRandomGenerator method), 4	
output_data (lysis.util.parameters.MicroParameters attribute), 15	simulations (lysis.util.parameters.MacroParameters attribute), 10	
overwrite() (lysis.util.datastore.DataStore method), 3	simulations (lysis.util.parameters.MicroParameters at- tribute), 16	
P	snap_proportion (ly-	
PERIODIC (lysis.util.constants.BoundaryCondition attribute), 1	sis.util.parameters.MicroParameters attribute), 16	
pore_size (lysis.util.parameters.MacroParameters at- tribute), 9	state (lysis.util.parameters.MacroParameters attribute), 10	
print_default_values() (ly-	${\tt status()} \ ({\it lysis.util.datastore.DataStore} \ {\it method}), 3$	
sis.util.parameters.MacroParameters static method), 9	Т	
print_default_values() (ly- sis.util.parameters.MicroParameters static method), 15	time_step (lysis.util.parameters.MacroParameters at- tribute), 10 to_dict() (lysis.util.datastore.DataStore method), 3	
processing_library (ly- sis.util.parameters.MacroParameters attribute), 10	to_dict() (lysis.util.parameters.Experiment method), 6 to_file() (lysis.util.parameters.Experiment method), 6 TOP (lysis.util.constants.BoundaryDirection attribute), 1	
protein_per_fiber (ly- sis.util.parameters.MicroParameters attribute), 15	total_edges (lysis.util.parameters.MacroParameters attribute), 11 total_fibers (lysis.util.parameters.MacroParameters	
protofibril_radius (ly- sis.util.parameters.MicroParameters attribute),	attribute), 11 total_molecules (ly-	
15	sis.util.parameters.MacroParameters attribute),	
R	total_time (lysis.util.parameters.MacroParameters at-	
random() (lysis.util.kiss.KissRandomGenerator method), 4	<pre>tribute), 11 total_time_steps (ly-</pre>	
RandomDraw (class in lysis.util.constants), 2 read_file() (lysis.util.parameters.Experiment method),	sis.util.parameters.MacroParameters attribute), 11	
6 REFLECTING (lysis.util.constants.BoundaryCondition at-	U	
tribute), 1	unbind_rate_PLG_intact (ly-	
RESTRICTED_MOVE (lysis.util.constants.RandomDraw attribute), 2	sis.util.parameters.MicroParameters attribute), 16	

Index 25

unbina_i	rate_PLG_nicked	(iy)
	sis.util.parameters.MicroParameters attribu	ute).
	16	
unbind_	rate_PLi	(ly-
	sis.util.parameters.MicroParameters attribu	ute)
	16	
unbind_	rate_tPA_woPLG	(ly-
	sis.util.parameters.MicroParameters attribu	ute).
	16	
unbind_1	rate_tPA_wPLG	(ly-
	sis.util.parameters.MicroParameters attribu	ıte)
	16	
UNBINDI	${\tt NG_TIME}$ (lysis.util.constants.RandomDraw	at-
	tribute), 2	
units()	(lysis.util.parameters.MacroParameters st	tatio
	method), 11	
units()	(lysis.util.parameters.MicroParameters st	tatio
	method), 17	
UP (lysis.ı	util.constants.FiberDirection attribute), 2	
V		
^		
xz_row	(lysis.util.parameters.MacroParameters	at-
	tribute), 11	
X	util.constants.FiberDirection attribute), 2 (lysis.util.parameters.MacroParameters	ai

26 Index