“Coeus”

Generated by Doxygen 1.8.6

Fri Feb 23 2018 16:43:12

Contents

Chapter 1

Main Page

Gnowee

Version

1.0

Gnowee is a general purpose hybrid metaheuristic optimization algorithm designed for rapid convergence to nearly globally optimum solutions for complex, constrained engineering problems with mixed-integer and combinatorial design vectors and high-cost, noisy, discontinuous, black box objective function evaluations. Gnowee's hybrid metaheuristic framework is based on a set of diverse, robust heuristics that appropriately balance diversification and intensification strategies across a wide range of optimization problems. Comparisons between Gnowee and several well-established metaheuristic algorithms are made for a set of eighteen continuous, mixed-integer, and combinatorial benchmarks. A summary of these benchmarks is `available`. These results demonstrate Gnoweee to have superior flexibility and convergence characteristics over a wide range of design spaces.

A paper, describing the Gnowee framework and benchmarks is `available`

Running Gnowee

For examples on how to run Gnowee, please refer to the `runGnowee notebook` included in the `src directory`. This contains multiple examples of how to modify and run Gnowee.

Building Documentation

To build the documentation, in the `docs/src directory` run the command:

>> doxygen Doxyfile

This will build the html and latex version of the documentation. The `symlink` in the `docs directory` for the html index should automatically update. If not the html index can be found `here`.

The up-to-date latex documentation is included in `pdf form`. If an update of the latex documentation is desired, go to the `docs/latex directory` and run the command:

>> make

This will build the latex documentation. The updated documentation file will be named `refman.pdf` and be placed in this directory.

Citation Information

To cite Gnowee, use the following:

Contact information

Bugs and suggestions for improvement can be submitted via the GitHub page: https://github.com/SlaybaughLab/Gnowee

Alternatively, questions or comments on Gnowee can be directed to:

James Bevins

james.e.bevins@gmail.com

Licensing Information

License:

GNU GPLv3.0+

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Acknowledements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. NSF 11-582.

This material is based upon work supported by the Department of Energy National Nuclear Security Administration through the Nuclear Science and Security Consortium under Award Numbers DE-NA0000979 and DE-NA0003180.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Packages

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

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

Module Documentation

7.1 Constraints

Defines a class to perform constraint calculations.

Classes

- class Constraints.Constraints

  The class creates a Constraints object that can be used in optimization algorithms.

7.1.1 Detailed Description

Defines a class to perform constraint calculations.

Author

James Bevins, Youdong Zhang

Date

22April

7.2 ObjectiveFunction

Defines a class to perform objective function calculations.

Defines a class to perform objective function calculations.

Author

James Bevins, Youdong Zhang

Date

22April

7.3 UserInputs

Defines a class to perform objective function calculations.

Classes

- class UserInputs.UserInputs

  The class creates a UserInputs object to store the user input file locations, read the user inputs, and set the appropriate classes required to run Coeus.

7.3.1 Detailed Description

Defines a class to perform objective function calculations. The keyword parameters discussed here are not case sensitive, and are separated by any number of spaces. Any other separator will result in the file not reading correctly. Only allowed key words will be read, and misspelled key words will be ignored. Depending on the input being specified, this may or may not result in a failed run. Warnings will be issued for unkown user input lines.

The Coeus user inputs will is constructed as follows:

There is one valid section name. The section name must be:

OBJECTIVE FUNCTION PARAMETERS

The allowed inputs for the OBJECTIVE FUCNTION PARAMETERS section are:

function Expects a function name corresponding to a function in the ObjectiveFunctions class.

tally Expects an integer number corresponding to an MCNP tally.

type This specifies the type of tally to be used for the objective function evaluation. Valid inputs are "total" or "spectrum". The actual tally can be have more components, but only the specified porion will be used for the objective fucntion evaluation.

objective If the type specified was "total", then this is a float or integer representing the desired objective. If "spectrum" was specified for the type, this is the number of bins for the objective spectrum followed by the form of the objective spectrum. The form responses are:

0 = "mcnp" 1 = "normalized" 2 = "differential" 3 = "normalized_differential" 4 = "lethargy" 5 = "normalized_lethargy"

Based on the response, the proper calculations will be performed to translate the mcnp tally to the correct form for objective function calculation. The spectrum is specified on the following line.

The spectrum is enetered in the form of energy amount seperated by a single (or multiple spaces. A couple of spectrum examples:

objective 4 1 0.25 2 0.5 5 0.2 10 0.05

or

objective 4 1 0.25 2 0.5 5 0.2 10 0.05

NOTE: The objective spectrum specified needs to be the same structure that is used used for the tally number specified.

Author

James Bevins

Date

23April

Chapter 8

Namespace Documentation

8.1 _init_ Namespace Reference

8.2 ADVANTG_Utilities Namespace Reference

Classes

- class ADVANTG_Settings

Functions

- def Print_ADVANTG_Input

  Print the generated MCNP input deck to file.

Variables

- tuple module_logger = logging.getLogger('Coesh.ADVANTG_Utilities')

8.2.1 Function Documentation

8.2.1.1 def ADVANTG_Utilities.Print_ADVANTG_Input (*eta*, *geom*, *S*, *num*, *cluster* = `False`)

Print the generated MCNP input deck to file.

Parameters

eta	[ETA parameters object] An object that contains all of the constraints required to initialize the geometry
geom	[MCNP_Geometry object] The geometry for running the MCNP radiation trasport code. Contains the surfaces, cells, and material information
S	[ADVANTG_Settings object] An object representing the settings for running the ADVANTG radiation trasport code.
num	integer The current cuckoo number being generated
cluster	boolean (optional) An indicator to change the file to run on a cluster using Run_Transport function and slurm job submission

8.2.2 Variable Documentation

8.2.2.1 tuple ADVANTG_Utilities.module_logger = logging.getLogger('Coesh.ADVANTG_Utilities')

8.3 Coeus Namespace Reference

Functions

- def print_MCNP_input_files

  Print MCNP input Files for each algorithm.
- def run_MCNP_on_algo

  Run MCNP for each algorithm.
- def main

  Entry point for the Coeus program.

8.3.1 Function Documentation

8.3.1.1 def Coeus.main ( )

Entry point for the Coeus program.

All inputs are optional. The program will load run inputs in the following order:

1) User specified path

2) Run directory defaults (Note: naming convention for files to be loaded from run directory must follow default naming convention shown in parameters)

3) Preset program defaults.

This order will be followed for each of the settings files so that some may be ommitted if desired.

Parameters

Parameters

input_path('inp')-:str	[default = ../Inputs/user_inputs.txt] The name and path for the user input file location. The format is a space delimited key word file as specified in the UserInputs class.
eta_constraints-_path('eta'):str	[default = ../Inputs/eta_constraints.csv] The name and path for the file containing the ETA design constraints. The format is a comma delimited key word input file. All keywords are optional. Non-specified keywords will default to preset program values.
gnowee_-settings_-path('gs'):str	[default = ../Inputs/gnowee_settings.csv] The name and path for the file containing the Gnowee search settings. The format is a comma delimited key word input file. All keywords are optional. Non-specified keywords will default to preset program values.
advantg_-settings_-path('adv'):str	[default = ../Inputs/advantg_settings.csv] The name and path for the file containing the advantg settings. The format is a comma delimited key word input file. All keywords are optional. Non-specified keywords will default to preset program values.
mcnp_settings_-path('mcnp'):str	[default = ../Inputs/mcnp_settings.csv] The name and path for the file containing the mcnp settings. The format is a comma delimited key word input file. All keywords are optional. Non-specified keywords will default to preset program values.
material_lib_-path('mat'):str	[default = ../Inputs/eta_materials_compendium.csv] The name and path for the file containing the materials to be included in the problem. The format is a comma delimited key word input file.
source_-path('src'):str	[default = ../Inputs/source.csv] The name and path for the file containing the starting neutron source distribution. The format is a comma delimited key word input file. All keywords are optional. Non-specified keywords will default to preset program values.

restart('r')-:boolean	Optional input to indicate the that search process will start with a preinitialized population. All members of the population must have full initialization inputs to work.

8.3.1.2 def Coeus.print_MCNP_input_files (*step*)

Print MCNP input Files for each algorithm.

Parameters

step	denotes which algorithm we are printing for

8.3.1.3 def Coeus.run_MCNP_on_algo (*args, algo, update_gen, update_feval, objFunc*)

Run MCNP for each algorithm.

Parameters

args	arguments for Run_transport
algo	denotes the algorithm we are on
update_gen	
update_feval	
objFunc	

8.4 Coeus_local Namespace Reference

Functions

- def main
- def print_MCNP_input_files

  Local Function definitions.
- def run_MCNP_on_algo

Variables

- tuple start_time = time.time()
- tuple rundir = os.path.abspath(os.path.join(os.path.abspath(os.getcwd()),os.pardir))
- tuple logger = logging.getLogger('Coeus')
- tuple hdlr = logging.FileHandler('{}/Results/logfile.log'.format(os.path.abspath(os.path.join(os.path.abspath(os.getcwd()), os.pardir))))
- tuple formatter = logging.Formatter('%(asctime)s %(levelname)s %(message)s')
- tuple restart = kwargs.get('restart')
- tuple obj_path = kwargs.get('obj_path')
- tuple eta_path = kwargs.get('eta_constraints_path')
- tuple gs_path = kwargs.get('gs_settings_path')
- tuple advantg_path = kwargs.get('advantg_settings_path')
- tuple mcnp_path = kwargs.get('mcnp_settings_path')
- tuple materials_library_path = kwargs.get('materials_library_path')
- tuple source_path = kwargs.get('source_path')
- tuple eta_params = ETA_Parameters()
- tuple g_set = Gnowee_Settings()
- tuple advantg_set = ADVANTG_Settings()
- tuple mcnp_set = MCNP_Settings(eta_params)

- tuple mat_lib = Build_Matlib(materials_library_path)
- list pop = [ ]
- tuple base_eta = MCNP_Geometry()
- tuple eta = MCNP_Geometry()
- tuple nps = eta.read_geom(rundir+str(i)+"/ETA.inp", mat_lib)
- list ids = [ ]

  33 if history.tline[-1].g>10: converge=True 33
- list particles = [ ]
- tuple stats = Meta_Stats()
- tuple history = Timeline()
- tuple mod_rat = Calc_Moderating_Ratio(mat_lib)
- tuple new_pop = Partial_Inversion(pop,mod_rat,mat_lib,g_set)

  Partial Inversion ########.
- converge = False

  Test Convergence ######## Test generational convergence.

8.4.1 Function Documentation

8.4.1.1 def Coeus_local.main (*kwargs*)

8.4.1.2 def Coeus_local.print_MCNP_input_files (*step*)

Local Function definitions.

8.4.1.3 def Coeus_local.run_MCNP_on_algo (*algo, update_gen, update_feval*)

8.4.2 Variable Documentation

8.4.2.1 tuple Coeus_local.advantg_path = kwargs.get('advantg_settings_path')

8.4.2.2 tuple Coeus_local.advantg_set = ADVANTG_Settings()

8.4.2.3 tuple Coeus_local.base_eta = MCNP_Geometry()

8.4.2.4 Coeus_local.converge = False

Test Convergence ######## Test generational convergence.

8.4.2.5 tuple Coeus_local.eta = MCNP_Geometry()

8.4.2.6 tuple Coeus_local.eta_params = ETA_Parameters()

8.4.2.7 tuple Coeus_local.eta_path = kwargs.get('eta_constraints_path')

8.4.2.8 tuple Coeus_local.formatter = logging.Formatter('%(asctime)s %(levelname)s %(message)s')

8.4.2.9 tuple Coeus_local.g_set = Gnowee_Settings()

8.4.2.10 tuple Coeus_local.gs_path = kwargs.get('gs_settings_path')

8.4.2.11 tuple Coeus_local.hdlr = logging.FileHandler('{}/Results/logfile.log'.format(os.path.abspath(os.path.join(os.path.abspath(os.getcwd()), os.pardir))))

8.4.2.12 tuple Coeus_local.history = Timeline()

8.4.2.13 list Coeus_local.ids = [ ]

33 if history.tline[-1].g$>$10: converge=True 33

Update weight window maps ######## Print MCNP input Files

8.4.2.14 tuple Coeus_local.logger = logging.getLogger('Coeus')

8.4.2.15 tuple Coeus_local.mat_lib = Build_Matlib(materials_library_path)

8.4.2.16 tuple Coeus_local.materials_library_path = kwargs.get('materials_library_path')

8.4.2.17 tuple Coeus_local.mcnp_path = kwargs.get('mcnp_settings_path')

8.4.2.18 tuple Coeus_local.mcnp_set = MCNP_Settings(eta_params)

8.4.2.19 tuple Coeus_local.mod_rat = Calc_Moderating_Ratio(mat_lib)

8.4.2.20 tuple Coeus_local.new_pop = Partial_Inversion(pop,mod_rat,mat_lib,g_set)

Partial Inversion ########.

Discard Cells ########.

3-opt ########

2-opt ########

Crossover ########.

Mutate ########.

Elite_Crossover ########.

Levy flight permutation of cells ########.

Levy flight permutation of materials ########.

8.4.2.21 tuple Coeus_local.nps = eta.read_geom(rundir+str(i)+"/ETA.inp", mat_lib)

8.4.2.22 tuple Coeus_local.obj_path = kwargs.get('obj_path')

8.4.2.23 list Coeus_local.particles = [ ]

8.4.2.24 tuple Coeus_local.pop = [ ]

8.4.2.25 Coeus_local.restart = kwargs.get('restart')

8.4.2.26 tuple Coeus_local.rundir = os.path.abspath(os.path.join(os.path.abspath(os.getcwd()),os.pardir))

8.4.2.27 tuple Coeus_local.source_path = kwargs.get('source_path')

8.4.2.28 tuple Coeus_local.start_time = time.time()

8.4.2.29 tuple Coeus_local.stats = Meta_Stats()

8.5 Constraints Namespace Reference

Classes

- class Constraints

  The class creates a Constraints object that can be used in optimization algorithms.

Variables

- tuple module_logger = logging.getLogger('Coeus.Constraints')

8.5.1 Variable Documentation

8.5.1.1 tuple Constraints.module_logger = logging.getLogger('Coeus.Constraints')

8.6 ETA_Utilities Namespace Reference

Classes

- class ETA_Parameters

Variables

- tuple module_logger = logging.getLogger('Coeus.ETA_Utilities')

8.6.1 Variable Documentation

8.6.1.1 tuple ETA_Utilities.module_logger = logging.getLogger('Coeus.ETA_Utilities')

8.7 Gnowee_Utilities Namespace Reference

Classes

- class Gnowee_Settings
- class Parent
- class Timeline

Functions

- def Calc_Fitness

  Print the generated MCNP input deck to file.
- def Pop_Update

  Updates the population based on the assessed fitness values.
- def Rejection_Bounds

  Application of problem boundaries to generated solutions.
- def Simple_Bounds

  Application of problem boundaries to generated solutions.

Variables

- tuple module_logger = logging.getLogger('Coeus.Gnowee_Utilities')

8.7.1 Function Documentation

8.7.1.1 def Gnowee_Utilities.Calc_Fitness (*ids*, *pop*, *obj*, *min_fiss* = 0, *max_w* = 1000)

Print the generated MCNP input deck to file.

Parameters

ids	[list of integers] The parents that need to have fitness solutions calculated
pop	[list of parent objects] The population and their design features
obj	ObjectiveFunction Object An object containing all of the parameters required for evaluating the objective function.
min_fiss	float (optional) A constraint specifying the minimum number fo fissions. Implemented as a soft constraint. [Default = 0]
max_w	float (optional) A constraint specifying the maximum weight of the assembly. Implemented as a hard constraint.

8.7.1.2 def Gnowee_Utilities.Pop_Update (*old*, *new*, *nps*, *runArgs*, *eta* = None, *mats* = None, *run* = None, *rr* = False)

Updates the population based on the assessed fitness values.

Parameters

old	[list of parent objects] The current population and their design features
new	[list of parent objects] The proposed population and their design features
nps	float The baseline number of NPS particles specified
runArgs	list of arguments for run transport
eta	[ETA parameters object] (optional) An object that contains all of the constraints required to initialize the geometry
mats	[dict of material objects] (optional) A dictionary of the material objects from which ETA materials can be selected
run	function (optional) A function that runs the radiation transport calculations
rr	boolean (optional) Indicator if random replacement is used to update the list

Returns

changes Integer The number of accepted changes
feval Integer The number of function evaluations performed for increasing the particles

8.7.1.3 def Gnowee_Utilities.Rejection_Bounds (*parent*, *child*, *stepsize*, *lb*, *ub*, *S*, *change_count* = 0)

Application of problem boundaries to generated solutions.

If a solution is outside of the bounds, the step is rejected and that particular value reverts to the previous solution.

Parameters

parent	array The current system designs
child	array The proposed new system designs
stepsize	float The Levy flight stepsize
lb	array The lower bounds of the design variable(s)
ub	array The upper bounds of the design variable(s)

S	[Gnowee Settings Object] An object representing the settings for the Gnowee optimization algorithm
change_count	integer (optional) Counter to track the number of solutions that occur outside of problem boundaries. Can be used to diagnose too large or small of alpha (Default: 0)

Returns

new array The new system designs that are within problem boundaries

8.7.1.4 def Gnowee_Utilities.Simple_Bounds (*tmp,* *lb,* *ub,* *change_count* = 0)

Application of problem boundaries to generated solutions.

Parameters

tmp	array The proposed new system designs
lb	array The lower bounds of the design variable(s)
ub	array The upper bounds of the design variable(s)
change_count	integer (optional) Counter to track the number of solutions that occur outside of problem boundaries. Can be used to diagnose too large or small of alpha (Default: 0)

Returns

tmp array The new system designs that are within problem boundaries

8.7.2 Variable Documentation

8.7.2.1 tuple Gnowee_Utilities.module_logger = logging.getLogger('Coeus.Gnowee_Utilities')

8.8 mainpage Namespace Reference

8.9 MCNP_Utilities Namespace Reference

Classes

- class MCNP_Settings
- class MCNP_Geometry
- class MCNP_Surface
- class MCNP_Cell

Functions

- def Print_MCNP_Input

  Print the generated MCNP input deck to file.
- def Read_Tally_Output

  Read the generated MCNP output and return the tally results.
- def Read_MCNP_Output

  Read the generated MCNP output and return the tally results.

Variables

- tuple module_logger = logging.getLogger('Coeus.MCNP_Utilities')

8.9.1 Function Documentation

8.9.1.1 def MCNP_Utilities.Print_MCNP_Input (*eta*, *tallySpectrum*, *geom*, *settings*, *mats*, *num*, *adv_print* = True)

Print the generated MCNP input deck to file.

Parameters

eta	[ETA parameters object] An object that contains all of the constraints required to initialize the geometry
tallySpectrum	[Numpy array] Contains the energy structure to be used for the tally.
geom	[MCNP_Geometry object] The geometry for running the MCNP radiation trasport code. Contains the surfaces, cells, and material information
settings	[MCNP_Settings object] An object representing the settings for running the MCNP radiation trasport code. Contains the source, physics, and tally information.
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes.
num	int The current parent number being generated
adv_print	boolean (optional) An optional indicator to determine whether to print weight window and source bias information in the input file from ADVANTG outputs.

8.9.1.2 def MCNP_Utilities.Read_MCNP_Output (*path*, *tnum*, *rnum*)

Read the generated MCNP output and return the tally results.

Parameters

path	String The path, including filename, to the MCNP output file to be read
tnum	String The number of the tally to be read. Returns the entire binned tally.
rnum	String The number of the tally to be read for the total reactions only.

Returns

tally array Array of tally results for the tally specified by tnum [Ebins, tally, uncertainty]
rxs array Total number of reactions for the tally specified by rx_num [tally, uncertainty]
weight float The total weight of the system

8.9.1.3 def MCNP_Utilities.Read_Tally_Output (*path*, *tnum*)

Read the generated MCNP output and return the tally results.

Parameters

path	String The path, including filename, to the MCNP output file to be read
tnum	String The number of the tally to be read

Returns

tally array Array of tally results

8.9.2 Variable Documentation

8.9.2.1 tuple MCNP_Utilities.module_logger = logging.getLogger('Coeus.MCNP_Utilities')

8.10 Metaheuristics Namespace Reference

Functions

- def Mat_Levy_Flights

  Change cell materials based on Levy draw.
- def Cell_Levy_Flights

  Cell Levy Flight: Change all cell and foil starting locations and cell deltas based on Levy draw.
- def Elite_Crossover

  Change the materials between the top parent and an elite parent based on moderating ratio.
- def Partial_Inversion

  Invert materials based on moderating ratio gradient.
- def Two_opt

  Implement 2_opt by reordering layers for top parents.
- def Crossover

  For each parent in top S.fe parents, N1, randomly select a parent, N2.
- def Three_opt

  Perform for horizontal macrobodies if the number of cells is greater than 6.
- def Discard_Cells

  Discard a cell from fd parents.
- def Mutate

  Mutate parent population and build new ones.

Variables

- tuple module_logger = logging.getLogger('Coeus.Metaheuristics')

8.10.1 Function Documentation

8.10.1.1 def Metaheuristics.Cell_Levy_Flights (*x*, *eta*, *S*)

Cell Levy Flight: Change all cell and foil starting locations and cell deltas based on Levy draw.

The parameters modified are z_{foil}, $ z_{hc}$, r_{vc}, $ r_{vc}$, z_{vc} , and $ z_{vc}$.

Parameters

x	[list of parent objects]
eta	Object An object representing the constraints for the eta design
S	Object An object representing the settings for the optimization algorithm

Returns

tmp [list of parent objects] The proposed parents representing new system designs

8.10.1.2 def Metaheuristics.Crossover (*x*, *S*)

For each parent in top S.fe parents, N1, randomly select a parent, N2.

Randomly select an overall cell from N1 and copy into N2. Repeat s.pt times.

Parameters

x	[list of parent objects] The current parents representing system designs
S	Object An object representing the settings for the optimization algorithm

Returns

tmp [list of parent objects] The proposed parents representing new system designs

8.10.1.3 def Metaheuristics.Discard_Cells (*x*, *mats*, *S*)

Discard a cell from fd parents.

Bias discard towards better solutions. Only accept if the discard improves the fitness.

Parameters

x	[list of parent objects] The current parents representing system designs
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes.
S	Object An object representing the settings for the optimization algorithm

Returns

tmp [list of parent objects] The proposed parents representing new system design

8.10.1.4 def Metaheuristics.Elite_Crossover (*x*, *mr*, *eta*, *mats*, *S*, *exclude*)

Change the materials between the top parent and an elite parent based on moderating ratio.

The materials will be changed by moderating ratio (for both 1 and 14 MeV). The choice will be based on a random number draw and be 50/50.

Parameters

x	[list of parent objects] The current parent representing system designs
mr	[list of moderating ratio objects] Contains the moderating ratios for the materials library used to guide the Levy flight
eta	[bject] An object representing the constraints for the eta design
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes.
S	[Object] An object representing the settings for the optimization algorithm
exclude	[list] A list of materials to be excluded

Returns

tmp [list of parent objects] The proposed parent representing new system design

8.10.1.5 def Metaheuristics.Mat_Levy_Flights (*x*, *mats*, *mr*, *S*, *exclude*)

Change cell materials based on Levy draw.

The materials will be changed by either a) using material library key list index numbers or b) moderating ratio (for both 1 and 14 MeV). The choice will be based on a random number draw and be 33/33/33. This provides for a decoupling to the moderation power to add a layer of more randomness, while maintaining a physics based Levy flight process.

Parameters

x	[list of parent objects] The current parents representing system designs
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes.
mr	[list of modertaing raio objects] Contains the moderating ratios for the materials library used to guide the Levy flight
S	object An object representing the settings for the optimization algorithm
exclude	list A list of materials to be excluded

Returns

tmp [list of parent objects] The proposed parents representing new system designs

8.10.1.6 def Metaheuristics.Mutate (*x,* *eta,* *S*)

Mutate parent population and build new ones.

Bias discard to poor solutions.

Parameters

x	[list of parent objects] The current parents representing system designs
eta	Object n object representing the constraints for the eta design
S	Object An object representing the settings for the optimization algorithm

Returns

tmp [list of parent objects] The proposed parents representing new system designs

8.10.1.7 def Metaheuristics.Partial_Inversion (*x,* *mr,* *mats,* *S*)

Invert materials based on moderating ratio gradient.

I.e. Pick random layer l. If layer l+1 has the next highest (or lowest) moderating ratio do nothing. Otherwise invert the layer(s) between layer l and the layer with the next higher (or lower) ratio.

Parameters

x	[list of parent objects] The current parents representing system designs
mr	[list of moderating ratio objects] Contains the moderating ratios for the materials library used to guide the inversion
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes.
S	Object An object representing the settings for the optimization algorithm

Returns

tmp [list of parent objects] The proposed parents representing new system designs

8.10.1.8 def Metaheuristics.Three_opt (*x,* *S*)

Perform for horizontal macrobodies if the number of cells is greater than 6.

Reorganizes the cells is all of the possible combinations for each parent.

Parameters

x	[list of parent objects] The current parents representing system designs
S	Object An object representing the settings for the optimization algorithm

Returns

tmp [list of parent objects] The proposed parents representing new system designs

8.10.1.9 def Metaheuristics.Two_opt (*x*, *S*)

Implement 2_opt by reordering layers for top parents.

Parameters

x	[list of parent objects] The current parents representing system designs
S	Object An object representing the settings for the optimization algorithm

Returns

tmp [list of parent objects] The proposed parents representing new system designs

8.10.2 Variable Documentation

8.10.2.1 tuple Metaheuristics.module_logger = logging.getLogger('Coeus.Metaheuristics')

8.11 NuclearData Namespace Reference

Classes

- class Moderating_Ratio

  Creates a moderating ratio object.

Functions

- def Build_Matlib

  Builds and initializes a library of elements and materials provided by user using PyNE material library functions.

- def Set_Density

  Initialized the material density for the elemental library.

- def Strip_Undesireables

  Removes materials from library that don't work from an engineering, safety, or cost perspective.

- def Calc_Moderating_Ratio

  Calculated and returns the moderating ratio for each material in a materials library.

Variables

- tuple module_logger = logging.getLogger('Coeus.NuclearData')

8.11.1 Function Documentation

8.11.1.1 def NuclearData.Build_Matlib (*mat_path* = '/home/pyne-user/Dropbox/UCB/Research/ETAs/META-CODE/MCNP/pyne/eta_materials_compendium.csv', *remove_gases* = True, *remove_liquids* = True, *remove_expensive* = True)

Builds and initializes a library of elements and materials provided by user using PyNE material library functions.

Parameters

mat_path	str absolute path to the location of the user supplied materials compendium
remove_gases	boolean allows the user to selectively remove gases from the elements library
remove_liquids	boolean allows the user to selectively remove liquids from the elements library
remove_expensive	boolean allows the user to selectively remove expensive materials from the elements library. "Expensive" encompasses from a materials hazard and cost perspective

Returns

> mat_lib [dictionary of material objects] a materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

8.11.1.2 def NuclearData.Calc_Moderating_Ratio (*mats*)

Calculated and returns the moderating ratio for each material in a materials library.

Currently limited to 1 and 14 MeV and the EAS data source. More sophisticated approaches are possible but not implemented. The moderating ratio is calculated as: MR={1 - (A-1)^2/2A ∗ ln[(A+1)(A-1)]} ∗ Sima_s / Sigma_a The EAS data source does not have any cross-sections for Zn, Dy, or Er.

Parameters

mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

Returns

> mr [list of Moderating_Ratio objects] A list containing the 1 and 14 MeV moderating ratios for the input material library

8.11.1.3 def NuclearData.Set_Density (*mat_lib*)

Initialized the material density for the elemental library.

Parameters

mat_lib	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

Returns

> mat_lib [dictionary of material objects] An updated materials library containing all relevant nulcear data required to run radiation transport codes.

8.11.1.4 def NuclearData.Strip_Undesireables (*mat_lib, remove_gases, remove_liquids, remove_expensive*)

Removes materials from library that don't work from an engineering, safety, or cost perspective.

Parameters

mat_lib	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

remove_gases	boolean Allows the user to selectively remove gases from the materials library
remove_liquids	boolean Allows the user to selectively remove liquids from the elements library
remove_expensive	boolean Allows the user to selectively remove expensive materials from the materials library. "Expensive" encompasses from a materials hazard and cost perspective

Returns

mat_lib [dictionary of material objects] An updated materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

8.11.2 Variable Documentation

8.11.2.1 tuple NuclearData.module_logger = logging.getLogger('Coeus.NuclearData')

8.12 ObjectiveFunction Namespace Reference

Classes

- class ObjectiveFunction

  The class creates a ObjectiveFunction object that can be used in optimization algorithms.

Variables

- tuple module_logger = logging.getLogger('Coeus.ObjectiveFunction')

8.12.1 Variable Documentation

8.12.1.1 tuple ObjectiveFunction.module_logger = logging.getLogger('Coeus.ObjectiveFunction')

8.13 OptiPlot Namespace Reference

Functions

- def Plot_Vars
- def Plot_Hist
- def Plot_Feval_Hist
- def Plot_TLF
- def Plot_Meta_Optimization

8.13.1 Function Documentation

8.13.1.1 def OptiPlot.Plot_Feval_Hist (*data* = [], *listdata* = [], *label* = [], *debug* = False)

```
Plots the fitness vs function evaluation results of an optimization algorithm run.
Can plot a single run or multiple to compare results.

Parameters
==========
Optional
========
data : array
    Contain the function eval history
    Columns are: [function evals, fitness, number of datapoints]
    (Default: Null)
```

```
listdata : list
    Contains a list of arrays containing the function eval history
    Columns are: [function evals, fitness, number of datapoints]
    (Default: Null)
label : list
    List of names of the optimization types ran
    (Default: Null)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
None, generates plot of design variables vs generation
```

8.13.1.2 def OptiPlot.Plot_Hist (*data*, *title =* [], *xlabel =* '', *ylabel =* [], *debug =* False)

```
Perform a cuckoo search optimization (CS)

Parameters
==========
data : list of event objects
    Contain the optimization history in event objects within the data list
    Attributes are: generation (.g), function evaluations (.e), fitness (.f), and design (.d)

Optional
========
low_bounds : array
    The lower bounds of the design variable(s)
up_bounds : array
    The upper bounds of the design variable(s)
title : string
    Title for plot
    (Default: [])
xlabel : string
    Label for independent variable
    (Default: [])
ylabel : list
    List of names of design variables
    (Default: [])
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
None, generates plot of design variables vs generation
```

8.13.1.3 def OptiPlot.Plot_Meta_Optimization (*data*, *label*, *title =* '', *debug =* False)

```
Plots the results of meta-optimization process for a given algorithm and parameter.

Parameters
==========
data : array
    Contain the function eval history
    Columns are: [function evals, fitness, number of datapoints]
label : list
    List of names of the problem types ran
Optional
========
title : string
    Title for plot
    (Default: '')
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)
```

```
Returns
=======
None, generates plot of design variables vs generation
```

8.13.1.4 def OptiPlot.Plot_TLF (*alpha* = 1.5, *gamma* = 1., *num_samp* = 1E7, *cut_point* = 10., *debug* = False)

```
Plots the comparison of the TLF to the Levy distribution

Parameters
==========

Optional
========
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1.)
num_samp : integer
    Number of Levy flights to sample (Default: 1E7)
cut_point : scalar
    Point at which to cut sampled Levy values and resample
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
None, generates plot
```

8.13.1.5 def OptiPlot.Plot_Vars (*data*, *low_bounds* = [], *up_bounds* = [], *title* = [], *label* = [], *debug* = False)

```
Perform a cuckoo search optimization (CS)

Parameters
==========
data : list of event objects
    Contain the optimization history in event objects within the data list
    Attributes are: generation (.g), function evaluations (.e), fitness (.f), and design (.d)
Optional
========
low_bounds : array
    The lower bounds of the design variable(s)
up_bounds : array
    The upper bounds of the design variable(s)
title : string
    Title for plot
    (Default: [])
label : list
    List of names of design variables
    (Default: [])
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
None, generates plot of design variables vs generation
```

8.14 pyDOE Namespace Reference

Namespaces

- build_regression_matrix

- doe_box_behnken
- doe_composite
- doe_factorial
- doe_fold
- doe_lhs
- doe_plackett_burman
- doe_repeat_center
- doe_star
- doe_union
- var_regression_matrix

Variables

- string __author__ = 'Abraham Lee'
- string __version__ = '0.3.8'

8.14.1 Detailed Description

```
==================================================================================
pyDOE: Design of Experiments for Python
==================================================================================

This code was originally published by the following individuals for use with
Scilab:
Copyright (C) 2012 - 2013 - Michael Baudin
Copyright (C) 2012 - Maria Christopoulou
Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
Copyright (C) 2009 - Yann Collette
Copyright (C) 2009 - CEA - Jean-Marc Martinez

website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.14.2 Variable Documentation

8.14.2.1 string pyDOE.__author__ = 'Abraham Lee'

8.14.2.2 string pyDOE.__version__ = '0.3.8'

8.15 pyDOE.build_regression_matrix Namespace Reference

Functions

- def grep
- def build_regression_matrix

8.15.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
Copyright (C) 2012 - 2013 - Michael Baudin
Copyright (C) 2012 - Maria Christopoulou
Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
Copyright (C) 2009 - Yann Collette
Copyright (C) 2009 - CEA - Jean-Marc Martinez
```

```
website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.15.2 Function Documentation

8.15.2.1 def pyDOE.build_regression_matrix.build_regression_matrix (*H,* *model,* *build* = None)

```
Build a regression matrix using a DOE matrix and a list of monomials.

Parameters
----------
H : 2d-array
model : str
build : bool-array

Returns
-------
R : 2d-array
```

8.15.2.2 def pyDOE.build_regression_matrix.grep (*haystack,* *needle*)

8.16 pyDOE.doe_box_behnken Namespace Reference

Functions

- def bbdesign

Variables

- list __all__ = ['bbdesign']

8.16.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
    Copyright (C) 2012 - 2013 - Michael Baudin
    Copyright (C) 2012 - Maria Christopoulou
    Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
    Copyright (C) 2009 - Yann Collette
    Copyright (C) 2009 - CEA - Jean-Marc Martinez

    website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.16.2 Function Documentation

8.16.2.1 def pyDOE.doe_box_behnken.bbdesign (*n,* *center* = None)

```
Create a Box-Behnken design

Parameters
----------
n : int
    The number of factors in the design
```

```
Optional
--------
center : int
    The number of center points to include (default = 1).

Returns
-------
mat : 2d-array
    The design matrix

Example
-------
::

    >>> bbdesign(3)
    array([[-1., -1.,  0.],
           [ 1., -1.,  0.],
           [-1.,  1.,  0.],
           [ 1.,  1.,  0.],
           [-1.,  0., -1.],
           [ 1.,  0., -1.],
           [-1.,  0.,  1.],
           [ 1.,  0.,  1.],
           [ 0., -1., -1.],
           [ 0.,  1., -1.],
           [ 0., -1.,  1.],
           [ 0.,  1.,  1.],
           [ 0.,  0.,  0.],
           [ 0.,  0.,  0.],
           [ 0.,  0.,  0.]])
```

8.16.3 Variable Documentation

8.16.3.1 list pyDOE.doe_box_behnken.__all__ = ['bbdesign']

8.17 pyDOE.doe_composite Namespace Reference

Functions

- def ccdesign

Variables

- list __all__ = ['ccdesign']

8.17.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
    Copyright (C) 2012 - 2013 - Michael Baudin
    Copyright (C) 2012 - Maria Christopoulou
    Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
    Copyright (C) 2009 - Yann Collette
    Copyright (C) 2009 - CEA - Jean-Marc Martinez

    website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.17.2 Function Documentation

8.17.2.1 def pyDOE.doe_composite.ccdesign (*n*, *center* = (4, 4, *alpha* = 'orthogonal', *face* = 'circumscribed')

```
Central composite design

Parameters
----------
n : int
    The number of factors in the design.

Optional
--------
center : int array
    A 1-by-2 array of integers, the number of center points in each block
    of the design. (Default: (4, 4)).
alpha : str
    A string describing the effect of alpha has on the variance. ``alpha``
    can take on the following values:

    1. 'orthogonal' or 'o' (Default)

    2. 'rotatable' or 'r'

face : str
    The relation between the start points and the corner (factorial) points.
    There are three options for this input:

    1. 'circumscribed' or 'ccc': This is the original form of the central
       composite design. The star points are at some distance ``alpha``
       from the center, based on the properties desired for the design.
       The start points establish new extremes for the low and high
       settings for all factors. These designs have circular, spherical,
       or hyperspherical symmetry and require 5 levels for each factor.
       Augmenting an existing factorial or resolution V fractional
       factorial design with star points can produce this design.

    2. 'inscribed' or 'cci': For those situations in which the limits
       specified for factor settings are truly limits, the CCI design
       uses the factors settings as the star points and creates a factorial
       or fractional factorial design within those limits (in other words,
       a CCI design is a scaled down CCC design with each factor level of
       the CCC design divided by ``alpha`` to generate the CCI design).
       This design also requires 5 levels of each factor.

    3. 'faced' or 'ccf': In this design, the star points are at the center
       of each face of the factorial space, so ``alpha`` = 1. This
       variety requires 3 levels of each factor. Augmenting an existing
       factorial or resolution V design with appropriate star points can
       also produce this design.

Notes
-----
- Fractional factorial designs are not (yet) available here.
- 'ccc' and 'cci' can be rotatable design, but 'ccf' cannot.
- If ``face`` is specified, while ``alpha`` is not, then the default value
  of ``alpha`` is 'orthogonal'.

Returns
-------
mat : 2d-array
    The design matrix with coded levels -1 and 1

Example
-------
::

    >>> ccdesign(3)
    array([[-1.        , -1.        , -1.        ],
           [ 1.        , -1.        , -1.        ],
           [-1.        ,  1.        , -1.        ],
           [ 1.        ,  1.        , -1.        ],
           [-1.        , -1.        ,  1.        ],
           [ 1.        , -1.        ,  1.        ],
           [-1.        ,  1.        ,  1.        ],
           [ 1.        ,  1.        ,  1.        ],
           [ 0.        ,  0.        ,  0.        ],
           [ 0.        ,  0.        ,  0.        ],
           [ 0.        ,  0.        ,  0.        ],
           [ 0.        ,  0.        ,  0.        ],
           [-1.82574186,  0.        ,  0.        ],
           [ 1.82574186,  0.        ,  0.        ],
           [ 0.        , -1.82574186,  0.        ],
           [ 0.        ,  1.82574186,  0.        ],
           [ 0.        ,  0.        , -1.82574186],
           [ 0.        ,  0.        ,  1.82574186],
           [ 0.        ,  0.        ,  0.        ],
           [ 0.        ,  0.        ,  0.        ],
           [ 0.        ,  0.        ,  0.        ],
           [ 0.        ,  0.        ,  0.        ]])
```

8.17.3 Variable Documentation

8.17.3.1 list pyDOE.doe_composite.__all__ = ['ccdesign']

8.18 pyDOE.doe_factorial Namespace Reference

Functions

- def fullfact
- def ff2n
- def fracfact

Variables

- list __all__ = ['np', 'fullfact', 'ff2n', 'fracfact']

8.18.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
Copyright (C) 2012 - 2013 - Michael Baudin
Copyright (C) 2012 - Maria Christopoulou
Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
Copyright (C) 2009 - Yann Collette
Copyright (C) 2009 - CEA - Jean-Marc Martinez

website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.18.2 Function Documentation

8.18.2.1 def pyDOE.doe_factorial.ff2n (*n*)

```
Create a 2-Level full-factorial design

Parameters
----------
n : int
    The number of factors in the design.

Returns
-------
```

```
mat : 2d-array
    The design matrix with coded levels -1 and 1

Example
-------
::

    >>> ff2n(3)
    array([[-1., -1., -1.],
           [ 1., -1., -1.],
           [-1.,  1., -1.],
           [ 1.,  1., -1.],
           [-1., -1.,  1.],
           [ 1., -1.,  1.],
           [-1.,  1.,  1.],
           [ 1.,  1.,  1.]])
```

8.18.2.2 def pyDOE.doe_factorial.fracfact (*gen*)

```
Create a 2-level fractional-factorial design with a generator string.

Parameters
----------
gen : str
    A string, consisting of lowercase, uppercase letters or operators "-"
    and "+", indicating the factors of the experiment

Returns
-------
H : 2d-array
    A m-by-n matrix, the fractional factorial design. m is 2^k, where k
    is the number of letters in ``gen``, and n is the total number of
    entries in ``gen``.

Notes
-----
In ``gen`` we define the main factors of the experiment and the factors
whose levels are the products of the main factors. For example, if

    gen = "a b ab"

then "a" and "b" are the main factors, while the 3rd factor is the product
of the first two. If we input uppercase letters in ``gen``, we get the same
result. We can also use the operators "+" and "-" in ``gen``.

For example, if

    gen = "a b -ab"

then the 3rd factor is the opposite of the product of "a" and "b".

The output matrix includes the two level full factorial design, built by
the main factors of ``gen``, and the products of the main factors. The
columns of ``H`` follow the sequence of ``gen``.

For example, if

    gen = "a b ab c"

then columns H[:, 0], H[:, 1], and H[:, 3] include the two level full
factorial design and H[:, 2] includes the products of the main factors.

Examples
--------
::

    >>> fracfact("a b ab")
    array([[-1., -1.,  1.],
           [ 1., -1., -1.],
           [-1.,  1., -1.],
           [ 1.,  1.,  1.]])
    >>> fracfact("A B AB")
    array([[-1., -1.,  1.],
           [ 1., -1., -1.],
           [-1.,  1., -1.],
           [ 1.,  1.,  1.]])

    >>> fracfact("a b -ab c +abc")
    array([[-1., -1., -1., -1., -1.],
           [ 1., -1.,  1., -1.,  1.],
           [-1.,  1.,  1., -1.,  1.],
           [ 1.,  1., -1., -1., -1.],
           [-1., -1., -1.,  1.,  1.],
           [ 1., -1.,  1.,  1., -1.],
           [-1.,  1.,  1.,  1., -1.],
           [ 1.,  1., -1.,  1.,  1.]])
```

8.18.2.3 def pyDOE.doe_factorial.fullfact (*levels*)

```
Create a general full-factorial design

Parameters
----------
levels : array-like
    An array of integers that indicate the number of levels of each input
    design factor.

Returns
-------
mat : 2d-array
    The design matrix with coded levels 0 to k-1 for a k-level factor

Example
-------
::

    >>> fullfact([2, 4, 3])
    array([[ 0.,  0.,  0.],
           [ 1.,  0.,  0.],
           [ 0.,  1.,  0.],
           [ 1.,  1.,  0.],
           [ 0.,  2.,  0.],
           [ 1.,  2.,  0.],
           [ 0.,  3.,  0.],
           [ 1.,  3.,  0.],
           [ 0.,  0.,  1.],
           [ 1.,  0.,  1.],
           [ 0.,  1.,  1.],
           [ 1.,  1.,  1.],
           [ 0.,  2.,  1.],
           [ 1.,  2.,  1.],
           [ 0.,  3.,  1.],
           [ 1.,  3.,  1.],
           [ 0.,  0.,  2.],
           [ 1.,  0.,  2.],
           [ 0.,  1.,  2.],
           [ 1.,  1.,  2.],
           [ 0.,  2.,  2.],
           [ 1.,  2.,  2.],
           [ 0.,  3.,  2.],
           [ 1.,  3.,  2.]])
```

8.18.3 Variable Documentation

8.18.3.1 list pyDOE.doe_factorial.__all__ = ['np', 'fullfact', 'ff2n', 'fracfact']

8.19 pyDOE.doe_fold Namespace Reference

Functions

- def fold

Variables

- list __all__ = ['fold']

8.19.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
Copyright (C) 2012 - 2013 - Michael Baudin
Copyright (C) 2012 - Maria Christopoulou
Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
Copyright (C) 2009 - Yann Collette
Copyright (C) 2009 - CEA - Jean-Marc Martinez

website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.19.2 Function Documentation

8.19.2.1 def pyDOE.doe_fold.fold (*H*, *columns* = None)

```
Fold a design to reduce confounding effects.

Parameters
----------
H : 2d-array
    The design matrix to be folded.
columns : array
    Indices of of columns to fold (Default: None). If ``columns=None`` is
    used, then all columns will be folded.

Returns
-------
Hf : 2d-array
    The folded design matrix.

Examples
--------
::
```

8.19.3 Variable Documentation

8.19.3.1 list pyDOE.doe_fold.__all__ = ['fold']

8.20 pyDOE.doe_lhs Namespace Reference

Functions

- def lhs

Variables

- list __all__ = ['lhs']

8.20.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
    Copyright (C) 2012 - 2013 - Michael Baudin
    Copyright (C) 2012 - Maria Christopoulou
    Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
    Copyright (C) 2009 - Yann Collette
    Copyright (C) 2009 - CEA - Jean-Marc Martinez

    website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.20.2 Function Documentation

8.20.2.1 def pyDOE.doe_lhs.lhs (*n*, *samples* = None, *criterion* = None, *iterations* = None)

```
Generate a latin-hypercube design

Parameters
----------
n : int
    The number of factors to generate samples for

Optional
--------
samples : int
    The number of samples to generate for each factor (Default: n)
criterion : str
    Allowable values are "center" or "c", "maximin" or "m",
    "centermaximin" or "cm", and "correlation" or "corr". If no value
    given, the design is simply randomized.
iterations : int
    The number of iterations in the maximin and correlations algorithms
    (Default: 5).

Returns
-------
H : 2d-array
    An n-by-samples design matrix that has been normalized so factor values
    are uniformly spaced between zero and one.

Example
-------
A 3-factor design (defaults to 3 samples)::

    >>> lhs(3)
    array([[ 0.40069325,  0.08118402,  0.69763298],
           [ 0.19524568,  0.41383587,  0.29947106],
           [ 0.85341601,  0.75460699,  0.360024  ]])

A 4-factor design with 6 samples::

    >>> lhs(4, samples=6)
    array([[ 0.27226812,  0.02811327,  0.62792445,  0.91988196],
           [ 0.76945538,  0.43501682,  0.01107457,  0.09583358],
           [ 0.45702981,  0.76073773,  0.90245401,  0.18773015],
           [ 0.99342115,  0.85814198,  0.16996665,  0.65069309],
           [ 0.63092013,  0.22148567,  0.33616859,  0.36332478],
           [ 0.05276917,  0.5819198 ,  0.67194243,  0.78703262]])

A 2-factor design with 5 centered samples::
    >>> lhs(2, samples=5, criterion='center')
    array([[ 0.3,  0.5],
           [ 0.7,  0.9],
           [ 0.1,  0.3],
           [ 0.9,  0.1],
           [ 0.5,  0.7]])

A 3-factor design with 4 samples where the minimum distance between
all samples has been maximized::

    >>> lhs(3, samples=4, criterion='maximin')
    array([[ 0.02642564,  0.55576963,  0.50261649],
           [ 0.51606589,  0.88933259,  0.34040838],
           [ 0.98431735,  0.0380364 ,  0.01621717],
           [ 0.40414671,  0.33339132,  0.84845707]])

A 4-factor design with 5 samples where the samples are as uncorrelated
as possible (within 10 iterations)::

    >>> lhs(4, samples=5, criterion='correlate', iterations=10)
```

8.20.3 Variable Documentation

8.20.3.1 list pyDOE.doe_lhs.__all__ = ['lhs']

8.21 pyDOE.doe_plackett_burman Namespace Reference

Functions

- def pbdesign

Variables

- list __all__ = ['pbdesign']

8.21.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
    Copyright (C) 2012 - 2013 - Michael Baudin
    Copyright (C) 2012 - Maria Christopoulou
    Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
    Copyright (C) 2009 - Yann Collette
    Copyright (C) 2009 - CEA - Jean-Marc Martinez

    website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.21.2 Function Documentation

8.21.2.1 def pyDOE.doe_plackett_burman.pbdesign (*n*)

```
Generate a Plackett-Burman design

Parameter
---------
n : int
    The number of factors to create a matrix for.

Returns
```

```
-------
H : 2d-array
    An orthogonal design matrix with n columns, one for each factor, and
    the number of rows being the next multiple of 4 higher than n (e.g.,
    for 1-3 factors there are 4 rows, for 4-7 factors there are 8 rows,
    etc.)

Example
-------

A 3-factor design::

    >>> pbdesign(3)
    array([[-1., -1.,  1.],
           [ 1., -1., -1.],
           [-1.,  1., -1.],
           [ 1.,  1.,  1.]])

A 5-factor design::

    >>> pbdesign(5)
    array([[-1., -1.,  1., -1.,  1.],
           [ 1., -1., -1., -1., -1.],
           [-1.,  1., -1., -1.,  1.],
           [ 1.,  1.,  1., -1., -1.],
           [-1., -1.,  1.,  1., -1.],
           [ 1., -1., -1.,  1.,  1.],
           [-1.,  1., -1.,  1., -1.],
           [ 1.,  1.,  1.,  1.,  1.]])
```

8.21.3 Variable Documentation

8.21.3.1 list pyDOE.doe_plackett_burman.__all__ = ['pbdesign']

8.22 pyDOE.doe_repeat_center Namespace Reference

Functions

- def repeat_center

8.22.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
Copyright (C) 2012 - 2013 - Michael Baudin
Copyright (C) 2012 - Maria Christopoulou
Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
Copyright (C) 2009 - Yann Collette
Copyright (C) 2009 - CEA - Jean-Marc Martinez

website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.22.2 Function Documentation

8.22.2.1 def pyDOE.doe_repeat_center.repeat_center (*n*, *repeat*)

```
Create the center-point portion of a design matrix

Parameters
----------
n : int
```

```
    The number of factors in the original design
repeat : int
    The number of center points to repeat

Returns
-------
mat : 2d-array
    The center-point portion of a design matrix (elements all zero).

Example
-------
::

    >>> repeat_center(3, 2)
    array([[ 0.,  0.,  0.],
           [ 0.,  0.,  0.]])
```

8.23 pyDOE.doe_star Namespace Reference

Functions

- def star

8.23.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
Copyright (C) 2012 - 2013 - Michael Baudin
Copyright (C) 2012 - Maria Christopoulou
Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
Copyright (C) 2009 - Yann Collette
Copyright (C) 2009 - CEA - Jean-Marc Martinez

website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.23.2 Function Documentation

8.23.2.1 def pyDOE.doe_star.star (*n*, *alpha* = ′faced′, *center* = (1, 1)

```
Create the star points of various design matrices

Parameters
----------
n : int
    The number of variables in the design

Optional
--------
alpha : str
    Available values are 'faced' (default), 'orthogonal', or 'rotatable'
center : array
    A 1-by-2 array of integers indicating the number of center points
    assigned in each block of the response surface design. Default is
    (1, 1).

Returns
-------
H : 2d-array
    The star-point portion of the design matrix (i.e. at +/- alpha)
a : scalar
    The alpha value to scale the star points with.
```

```
Example
-------
::

    >>> star(3)
    array([[-1.,  0.,  0.],
           [ 1.,  0.,  0.],
           [ 0., -1.,  0.],
           [ 0.,  1.,  0.],
           [ 0.,  0., -1.],
           [ 0.,  0.,  1.]])
```

8.24 pyDOE.doe_union Namespace Reference

Functions

- def union

8.24.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
    Copyright (C) 2012 - 2013 - Michael Baudin
    Copyright (C) 2012 - Maria Christopoulou
    Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
    Copyright (C) 2009 - Yann Collette
    Copyright (C) 2009 - CEA - Jean-Marc Martinez

    website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.24.2 Function Documentation

8.24.2.1 def pyDOE.doe_union.union (*H1*, *H2*)

```
Join two matrices by stacking them on top of each other.

Parameters
----------
H1 : 2d-array
    The matrix that goes on top of the new matrix
H2 : 2d-array
    The matrix that goes on bottom of the new matrix

Returns
-------
mat : 2d-array
    The new matrix that contains the rows of ``H1`` on top of the rows of
    ``H2``.

Example
-------
::

    >>> union(np.eye(2), -np.eye(2))
    array([[ 1.,  0.],
           [ 0.,  1.],
           [-1.,  0.],
           [ 0., -1.]])
```

8.25 pyDOE.var_regression_matrix Namespace Reference

Functions

- def var_regression_matrix

8.25.1 Detailed Description

```
This code was originally published by the following individuals for use with
Scilab:
Copyright (C) 2012 - 2013 - Michael Baudin
Copyright (C) 2012 - Maria Christopoulou
Copyright (C) 2010 - 2011 - INRIA - Michael Baudin
Copyright (C) 2009 - Yann Collette
Copyright (C) 2009 - CEA - Jean-Marc Martinez

website: forge.scilab.org/index.php/p/scidoe/sourcetree/master/macros

Much thanks goes to these individuals. It has been converted to Python by
Abraham Lee.
```

8.25.2 Function Documentation

8.25.2.1 def pyDOE.var_regression_matrix.var_regression_matrix (*H*, *x*, *model*, *sigma* = 1)

```
Compute the variance of the ’regression error’.

Parameters
----------
H : 2d-array
    The regression matrix
x : 2d-array
    The coordinates to calculate the regression error variance at.
model : str
    A string of tokens that define the regression model (e.g.
    ’1 x1 x2 x1*x2’)
sigma : scalar
    An estimate of the variance (default: 1).

Returns
-------
var : scalar
    The variance of the regression error, evaluated at ``x``.
```

8.26 SamplingMethods Namespace Reference

Functions

- def Initial_Samples
- def integrand
- def Levy_Function
- def Levy
- def TLF
- def NOLH
- def params
- def Get_CDR_Permutations

Variables

- tuple module_logger = logging.getLogger('METACODE.SamplingMethods')
- tuple parser
- string help = "The configuration vector given as a list N1 N2 ... Nm"
- tuple args = parser.parse_args()

8.26.1 Function Documentation

8.26.1.1 def SamplingMethods.Get_CDR_Permutations (*dim*, *debug* = False)

```
Generate a set of samples in a given phase space

Parameters
==========
dim : integer
    The dimension of the phase space

Optional
========
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
conf : array
    Configuration vector
remove : array
    Array containing the indexes of the colummns to be removed from conf vetor
```

8.26.1.2 def SamplingMethods.Initial_Samples (*lb*, *ub*, *method*, *n* = 25, *debug* = False)

```
Generate a set of samples in a given phase space

Parameters
==========
lb : array
    The lower bounds of the design variable(s)
ub : array
    The upper bounds of the design variable(s)
method : string
    String representing the chosen sampling method

Optional
========
n : int
    The number of samples to be generated.  Ignored for nolh algorithms. (Default=25)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
s : array
    The list of coordinates for the sampled phase space
```

8.26.1.3 def SamplingMethods.integrand (*x*, *a*, *b*, *g*)

8.26.1.4 def SamplingMethods.Levy (*nc*, *nr* = 0, *alpha* = 1.5, *gamma* = 1, *n* = 1, *debug* = False)

```
Sample the Levy distribution to generate Levy flights steps
```

```
Parameters
==========
nc : int
    The number of columns of Levy values for the return array

Optional
========
nr : int
    The number of rows of Levy values for the return array
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1)
n : integer
    Number of independent variables - can be used to reduce Levy flight variance (Default: 1)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
z : array
    Array representing the levy flights for each nest
```

8.26.1.5 def SamplingMethods.Levy_Function (*bins*, *alpha* = 1.5, *gamma* = 1, *debug* = False)

```
Generate the levy function.

Parameters
==========
bins : array
    The bin values used to generate the disribution

Optional
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
l : array
    Array representing the levy flights for each nest
```

8.26.1.6 def SamplingMethods.NOLH (*conf*, *remove* = None)

```
This library allows to generate Nearly Orthogonal Latin Hypercubes (NOLH) according to
Cioppa (2007) and De Rainville et al. (2012) and reference therein.

https://pypi.python.org/pypi/pynolh

Constructs a Nearly Orthogonal Latin Hypercube (NOLH) of order *m* from a configuration vector
*conf*. The configuration vector may contain either the numbers in $[0 q-1]$ or $[1 q]$ where
$q = 2^{m-1}$. The columns to be *removed* are also in $[0 d-1]$ or $[1 d]$ where $d = m +
\binom{m-1}{2}$ is the NOLH dimensionality.

The whole library is incorporated here with minimal modification for commonality and
consolidation of methods.

Parameters
==========
conf : array
    Configuration vector
```

```
Optional
========
remove : array
    Array containing the indexes of the colummns to be removed from conf vetor
    (Default: NONE)

Returns
=======
nolh : array
    Array containing nearly orthogonal latin hypercube sampling.
```

8.26.1.7 def SamplingMethods.params (*dim*)

```
Returns the NOLH order $m$, the required configuration length $q$
and the number of columns to remove to obtain the desired dimensionality.
```

8.26.1.8 def SamplingMethods.TLF (*alpha* = 1.5, *gamma* = 1., *num_samp* = 1, *cut_point* = 20., *debug* = False)

```
Generates and samples from a truncated levy flight distribution. Produces a Levy Equivalent
distribution on the interval (0,1)

Parameters
==========

Optional
========
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1.)
num_samp : integer
    Number of Levy flights to sample (Default: 1)
cut_point : scalar
    Point at which to cut sampled Levy values and resample
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
levy : array
    Array representing the levy flights on the interval (0,1)
```

8.26.2 Variable Documentation

8.26.2.1 tuple SamplingMethods.args = parser.parse_args()

8.26.2.2 string SamplingMethods.help = "The configuration vector given as a list N1 N2 ... Nm"

8.26.2.3 tuple SamplingMethods.module_logger = logging.getLogger('METACODE.SamplingMethods')

8.26.2.4 tuple SamplingMethods.parser

Initial value:

```
= argparse.ArgumentParser(description=("Compute a Nearly "
        "Orthogonal Latin hypercube from a configuration vector."))
```

8.27 SamplingMethods_Full Namespace Reference

Functions

- def Initial_Samples
- def Plot_Samples
- def integrand
- def Levy_Function
- def Levy
- def TLF
- def Plot_Levy
- def NOLH
- def params
- def Get_CDR_Permutations

Variables

- tuple parser
- string help = "The configuration vector given as a list N1 N2 ... Nm"
- tuple args = parser.parse_args()

8.27.1 Function Documentation

8.27.1.1 def SamplingMethods_Full.Get_CDR_Permutations (*dim*, *debug* = `False`)

```
Generate a set of samples in a given phase space

Parameters
==========
dim : integer
    The dimension of the phase space

Optional
========
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
conf : array
    Configuration vector
remove : array
    Array containing the indexes of the colummns to be removed from conf vetor
```

8.27.1.2 def SamplingMethods_Full.Initial_Samples (*lb*, *ub*, *method*, *n* = `25`, *debug* = `False`)

```
Generate a set of samples in a given phase space

Parameters
==========
lb : array
    The lower bounds of the design variable(s)
ub : array
    The upper bounds of the design variable(s)
method : string
    String representing the chosen sampling method

Optional
========
n : int
```

```
    The number of samples to be generated.  Ignored for nolh algorithms. (Default=25)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
s : array
    The list of coordinates for the sampled phase space
```

8.27.1.3 def SamplingMethods_Full.integrand (*x*, *a*, *b*, *g*)

8.27.1.4 def SamplingMethods_Full.Levy (*nc*, *nr =* `0`, *alpha =* `1.5`, *gamma =* `1`, *n =* `1`, *debug =* `False`)

```
Sample the Levy distribution to generate Levy flights steps

Parameters
==========
nc : int
    The number of columns of Levy values for the return array

Optional
========
nr : int
    The number of rows of Levy values for the return array
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1)
n : integer
    Number of independent variables - can be used to reduce Levy flight variance (Default: 1)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
z : array
    Array representing the levy flights for each nest
```

8.27.1.5 def SamplingMethods_Full.Levy_Function (*bins*, *alpha =* `1.5`, *gamma =* `1`, *debug =* `False`)

```
Generate the levy function.

Parameters
==========
bins : array
    The bin values used to generate the disribution

Optional
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
l : array
    Array representing the levy flights for each nest
```

8.27.1.6 def SamplingMethods_Full.NOLH (*conf*, *remove* = None)

```
This library allows to generate Nearly Orthogonal Latin Hypercubes (NOLH) according to
Cioppa (2007) and De Rainville et al. (2012) and reference therein.

https://pypi.python.org/pypi/pynolh

Constructs a Nearly Orthogonal Latin Hypercube (NOLH) of order *m* from a configuration vector
*conf*. The configuration vector may contain either the numbers in $[0 q-1]$ or $[1 q]$ where
$q = 2^{m-1}$. The columns to be *removed* are also in $[0 d-1]$ or $[1 d]$ where $d = m +
\binom{m-1}{2}$ is the NOLH dimensionality.

The whole library is incorporated here with minimal modification for commonality and
consolidation of methods.

Parameters
==========
conf : array
    Configuration vector

Optional
========
remove : array
    Array containing the indexes of the colummns to be removed from conf vetor
    (Default: NONE)

Returns
=======
nolh : array
    Array containing nearly orthogonal latin hypercube sampling.
```

8.27.1.7 def SamplingMethods_Full.params (*dim*)

```
Returns the NOLH order $m$, the required configuration length $q$
and the number of columns to remove to obtain the desired dimensionality.
```

8.27.1.8 def SamplingMethods_Full.Plot_Levy (*first*, *xbins*, *second* = [], *sec_bins* = [], *alpha* = 1.5, *gamma* = 1., *log* = False, *debug* = False)

```
Plots the levy distribution

Parameters
==========
first : array
    The array of the Levy distribution
xbins : array
    The bin values used to plot the disribution

Optional
========
second : array
    An array of a second data set to plot for comparison
sec_bins : array
    The bin values used for the second data set
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1.)
log : boolean
    If True, plot on a log scale
    (Default: False)
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
```

```
=======
None
```

8.27.1.9 def SamplingMethods_Full.Plot_Samples (*s*, *debug* = False)

```
Plot the first 2 and 3 dimensions on the sample distribution.

Parameters
==========
s : array
    The list of coordinates for the sampled phase space

Optional
========
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
None
```

8.27.1.10 def SamplingMethods_Full.TLF (*alpha* = 1.5, *gamma* = 1., *num_samp* = 1, *cut_point* = 20., *debug* = False)

```
Generates and samples from a truncated levy flight distribution. Produces a Levy Equivalent
distribution on the interval (0,1)

Parameters
==========

Optional
========
alpha : scalar
    Levy exponent - defines the index of the distribution and controls scale properties of the stochastic proc
    (Default: 1.5)
gamma : scalar
    Gamma - Scale unit of process for Levy flights (Default: 1.)
num_samp : integer
    Number of Levy flights to sample (Default: 1)
cut_point : scalar
    Point at which to cut sampled Levy values and resample
debug : boolean
    If True, progress statements will be displayed every iteration
    (Default: False)

Returns
=======
levy : array
    Array representing the levy flights on the interval (0,1)
```

8.27.2 Variable Documentation

8.27.2.1 tuple SamplingMethods_Full.args = parser.parse_args()

8.27.2.2 string SamplingMethods_Full.help = "The configuration vector given as a list N1 N2 ... Nm"

8.27.2.3 tuple SamplingMethods_Full.parser

Initial value:

```
= argparse.ArgumentParser(description=("Compute a Nearly "
        "Orthogonal Latin hypercube from a configuration vector."))
```

8.28 setup Namespace Reference

Functions

- def read

8.28.1 Function Documentation

8.28.1.1 def setup.read (*fname*)

8.29 test_ADVANTG_Utilities Namespace Reference

Functions

- def test_ADVANTG_setings
- def test_ADVANTG_settings_repr
- def test_ADVANTG_settings_str
- def test_ADVANTG_settings_read_settings
- def test_Print_ADVANTG_Input

Variables

- tuple constraint_path = os.getcwd()
- tuple mat_path = os.getcwd()
- tuple set_path = os.getcwd()
- string test_settings_repr = ’ADVANTG Settings(dplus, cadis, mcnp silo, 24, True, 0.01, 1, 0.5, 0.5, 0.5, 0.25, 0.25, 0.05, 1.0)’
- string test_settings_str

8.29.1 Function Documentation

8.29.1.1 def test_ADVANTG_Utilities.test_ADVANTG_setings ( )

8.29.1.2 def test_ADVANTG_Utilities.test_ADVANTG_settings_read_settings ( )

8.29.1.3 def test_ADVANTG_Utilities.test_ADVANTG_settings_repr ( )

8.29.1.4 def test_ADVANTG_Utilities.test_ADVANTG_settings_str ( )

8.29.1.5 def test_ADVANTG_Utilities.test_Print_ADVANTG_Input ( )

8.29.2 Variable Documentation

8.29.2.1 tuple test_ADVANTG_Utilities.constraint_path = os.getcwd()

8.29.2.2 tuple test_ADVANTG_Utilities.mat_path = os.getcwd()

8.29.2.3 tuple test_ADVANTG_Utilities.set_path = os.getcwd()

8.29.2.4 string test_ADVANTG_Utilities.test_settings_repr = ’ADVANTG Settings(dplus, cadis, mcnp silo, 24, True, 0.01, 1, 0.5, 0.5, 0.5, 0.25, 0.25, 0.05, 1.0)’

8.29.2.5 string test_ADVANTG_Utilities.test_settings_str

Initial value:

```
= "\nADVANTG Program Settings:\n\
Multi-Group Library = dplus\n\
Solution Method = cadis\n\
Outputs = mcnp silo\n\
Adjoint Tally Number = 24\n\
Force Point Source = True\n\
Material Mix Tolerance = 0.01\n\
Scattering Order = 1\n\
ETA X Spacing Interval = 0.5\n\
ETA Y Spacing Interval = 0.5\n\
ETA Z Spacing Interval = 0.5\n\
Foil X Spacing Interval = 0.25\n\
Foil Y Spacing Interval = 0.25\n\
Foil Z Spacing Interval = 0.05\n\
External Spacing Interval = 1.0\n"
```

8.30 test_ETA_Utilities Namespace Reference

Functions

- def test_eta_parameters
- def test_eta_parameters_repr
- def test_eta__parameters_str
- def test_eta_parameters_read_obj
- def test_eta_parameters_read_constraints

Variables

- tuple eta_path = os.getcwd()
- tuple obj_path = os.getcwd()
- tuple spectrum
- string test_parameters_repr = "ETA_Params(normalized differential, [], 1.59896054911e-06, 125.0, 5e+15, 15.24, 0.3, 0.5, 52.14, 1.0, 2.4, 5.48, 9.39, 70.22, Al, Al, Air (dry near sea level), Pb, 0.014, 2.69, Al, [0.1, 0.1, 0.1, 0.1, 0.01], 2.5, ['Zr', 'Zn', 'In', 'Al', 'Ta'], In, Al, [0.0254, 0.0127], 1.252, ['Au', 'Pb'], Al, Fe, 2.0, 3, 7)"
- string test_parameters_str

8.30.1 Function Documentation

8.30.1.1 def test_ETA_Utilities.test_eta__parameters_str ( )

8.30.1.2 def test_ETA_Utilities.test_eta_parameters ( )

8.30.1.3 def test_ETA_Utilities.test_eta_parameters_read_constraints ( )

8.30.1.4 def test_ETA_Utilities.test_eta_parameters_read_obj ( )

8.30.1.5 def test_ETA_Utilities.test_eta_parameters_repr ( )

8.30.2 Variable Documentation

8.30.2.1 tuple test_ETA_Utilities.eta_path = os.getcwd()

8.30.2.2 tuple test_ETA_Utilities.obj_path = os.getcwd()

8.30.2.3 tuple test_ETA_Utilities.spectrum

Initial value:

```
= np.array([[1.0E-8,1.0E-5], [1.0E-7,2.0E-5], [1.0E-6,1.0E-1], [1.24E-2,1.2546568188E+2],
             [2.345E+1,2.53E+12]])
```

8.30.2.4 string test_ETA_Utilities.test_parameters_repr = "ETA_Params(normalized differential, [ ], 1.59896054911e-06, 125.0, 5e+15, 15.24, 0.3, 0.5, 52.14, 1.0, 2.4, 5.48, 9.39, 70.22, Al, Al, Air (dry near sea level), Pb, 0.014, 2.69, Al, [0.1, 0.1, 0.1, 0.1, 0.01], 2.5, [’Zr’, ’Zn’, ’In’, ’Al’, ’Ta’], In, Al, [0.0254, 0.0127], 1.252, [’Au’, ’Pb’], Al, Fe, 2.0, 3, 7)"

8.30.2.5 string test_ETA_Utilities.test_parameters_str

8.31 test_Gnowee_Utilities Namespace Reference

Functions

- def test_gnowee_settings
- def test_gnowee_settings_read_settings
- def test_parent
- def test_timeline
- def test_pop_update
- def test_Rejection_Bounds

Variables

- string test_repr = ’Gnowee Settings(25, lhc, 0.25, 0.2, 0.4, 10000, 100000, 1e-06, 200, 0.01, 0.01, 1.5, 1.0, 1, 10.0)’
- string test_str = ’\nGnowee Optimization Settings:\nPopulation Size = 25\nInitial sampling method = lhc\n-Discovery Fraction = 0.2\nElite fraction = 0.2\nLevy fraction = 0.4\nMaximum number of genrations = 10000\n-Maximum number of function evaluations = 100000\nStall convergence tolerance = 1e-06\nStall iteration limit = 200\nOptimal fitness = 0.01\nOptimal convergence tolerance = 0.01\nLevy exponent = 1.5\nLevy scale unit = 1.0\nLevy independent variables = 1\nStep size scaling factor = 10.0\n’
- tuple set_path = os.getcwd()
- tuple mcnp_path = os.getcwd()
- tuple source_path = os.getcwd()
- tuple eta_path = os.getcwd()
- tuple mat_path = os.getcwd()
- tuple obj_path = os.getcwd()

8.31.1 Function Documentation

8.31.1.1 def test_Gnowee_Utilities.test_gnowee_settings ( )

8.31.1.2 def test_Gnowee_Utilities.test_gnowee_settings_read_settings ( )

8.31.1.3 def test_Gnowee_Utilities.test_parent ( )

8.31.1.4 def test_Gnowee_Utilities.test_pop_update ( )

8.31.1.5 def test_Gnowee_Utilities.test_Rejection_Bounds ( )

8.31.1.6 def test_Gnowee_Utilities.test_timeline ( )

8.31.2 Variable Documentation

8.31.2.1 tuple test_Gnowee_Utilities.eta_path = os.getcwd()

8.31.2.2 tuple test_Gnowee_Utilities.mat_path = os.getcwd()

8.31.2.3 tuple test_Gnowee_Utilities.mcnp_path = os.getcwd()

8.31.2.4 tuple test_Gnowee_Utilities.obj_path = os.getcwd()

8.31.2.5 tuple test_Gnowee_Utilities.set_path = os.getcwd()

8.31.2.6 tuple test_Gnowee_Utilities.source_path = os.getcwd()

8.31.2.7 string test_Gnowee_Utilities.test_repr = 'Gnowee Settings(25, lhc, 0.25, 0.2, 0.4, 10000, 100000, 1e-06, 200, 0.01, 0.01, 1.5, 1.0, 1, 10.0)'

8.31.2.8 string test_Gnowee_Utilities.test_str = '\nGnowee Optimization Settings:\nPopulation Size = 25\nInitial sampling method = lhc\nDiscovery Fraction = 0.2\nElite fraction = 0.2\nLevy fraction = 0.4\nMaximum number of genrations = 10000\nMaximum number of function evaluations = 100000\nStall convergence tolerance = 1e-06\nStall iteration limit = 200\nOptimal fitness = 0.01\nOptimal convergence tolerance = 0.01\nLevy exponent = 1.5\nLevy scale unit = 1.0\nLevy independent variables = 1\nStep size scaling factor = 10.0\n'

8.32 test_MCNP_Utilities Namespace Reference

Functions

- def test_mcnp_surface1
- def test_mcnp_surface2
- def test_mcnp_surface3
- def test_mcnp_surface4
- def test_mcnp_surface5
- def test_mcnp_surface6
- def test_mcnp_surface7
- def test_mcnp_surface8
- def test_mcnp_surface9
- def test_mcnp_surface10
- def test_mcnp_surface_repr
- def test_mcnp_surface_str
- def test_mcnp_cell1
- def test_mcnp_cell_repr
- def test_mcnp_cell_str1
- def test_mcnp_cell_str2
- def test_mcnp_cell_str3
- def test_mcnp_cell_str4
- def test_mcnp_addMat
- def test_mcnp_addSurf
- def test_mcnp_addCell
- def test_mcnp_geometry
- def test_init_geometry
- def test_mcnp_settings_read_source
- def test_Print_MCNP_Input
- def test_mcnp_settings_read_settings
- def test_Read_Tally_Output
- def test_Read_MCNP_Output
- def test_Read_MCNP_Output2

Variables

- list source
- tuple set_path = os.getcwd()
- tuple src_path = os.getcwd()
- tuple constraint_path = os.getcwd()
- tuple materials_library_path = os.getcwd()
- string test_surf_repr = 'MCNP Surface(600, TRC, vx=1.0, vy=2.0, vz=3.0, hx=2.5, hy=23.6, hz=23.56, r1=3.4, r2=1.0, c=test)'
- string test_surf_str = '700 px 2.00000 $test\n'
- string test_cell_repr = 'MCNP Cell:(1, mat=10, units=atom, density=0.0422, booleam geom=500 -501, n imp=1, p_imp=0, comment=)'
- string test_cell_str1 = '1 10 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n'
- string test_cell_str2 = '1 10 -4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n'
- string test_cell_str3 = '1 10 500 -501 imp:n=1 imp:p=0 $\n'
- string test_cell_str4 = '1 10 -4.22300e-02 (500 -501):(502 -503):(504 -505):(506 -507):(508\n -509):(509 -510) imp:n=1 imp:p=0 $\n'
- string mat_card = "C name: Air (dry near sea level)\nC density = 0.0\nm?\n 6012 -1.2256e-04\n 6013 -1.4365e-06\n 7014 -7.5527e-01\n 8016 -2.3178e-01\n 18036 -3.8527e-05\n 18038 -7.6673e-06\n 18040 -1.2781e-02\n"
- string test_geom_str1 = 'MCNP geometry instance properties:\nMCNP Surfaces:\n509 TRC 1.00000 2.00000 3.00000 2.50000 23.60000 23.56000 \n 3.40000 1.00000 $one\n\n504 px 2.00000 $two\n\n505 Py -2.00000 $three\n\nMCNP Cells:\n1 11 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\n2 12 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\n3 13 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\nMCNP Materials:\nAir (dry near sea level)\nAl\n'
- string base_geom = 'MCNP geometry instance properties:\nMCNP Surfaces:\n500 TRC 0.00000 0.00000 16.12650 0.00000 0.00000 14.35147 \n 0.00001 5.16119 $inner debris cover\n\n501 TRC 0.00000 0.00000 15.24000 0.00000 0.00000 15.23797 \n 0.00001 5.48000 $outer debris cover\n\n502 TRC 0.00000 0.00000 30.77797 0.00000 0.00000 10.57235 \n 5.26908 9.07119 $inner cone\n\n503 TRC 0.00000 0.00000 30.47797 0.00000 0.00000 10.87235 \n 5.48000 9.39000 $outer cone\n\n504 RCC 0.00000 0.00000 41.35032 0.00000 0.00000 9.78968 \n 8.89000 $inner cylinder\n\n505 RCC 0.00000 0.00000 41.35032 0.00000 0.00000 9.78968 \n 9.39000 $outer cylinder\n\n506 RCC 0.00000 0.00000 51.14000 0.00000 0.00000 1.00000 \n 9.39000 $cover\n\n507 RCC 0.00000 0.00000 52.14000 0.00000 0.00000 2.40000 \n 5.63400 $adapter\n\nMCNP Cells:\n1 1 -2.70000e+00 500 -501 imp:n=1 imp:p=0 $\n\n2 1 -2.70000e+00 502 -503 imp:n=1 imp:p=0 $\n\n3 1 -2.70000e+00 504 -505 imp:n=1 imp:p=0 $\n\n4 1 -2.70000e+00 -506 imp:n=1 imp:p=0 $\n\n5 1 -2.70000e+00 -507 imp:n=1 imp:p=0 $\n\nMCNP Materials:\nAl\nZr\nZn\nIn\nTa\nAu\nPb\nFe\n'

8.32.1 Function Documentation

8.32.1.1 def test_MCNP_Utilities.test_init_geometry ( )

8.32.1.2 def test_MCNP_Utilities.test_mcnp_addCell ( )

8.32.1.3 def test_MCNP_Utilities.test_mcnp_addMat ( )

8.32.1.4 def test_MCNP_Utilities.test_mcnp_addSurf ( )

8.32.1.5 def test_MCNP_Utilities.test_mcnp_cell1 ( )

8.32.1.6 def test_MCNP_Utilities.test_mcnp_cell_repr ( )

8.32.1.7 def test_MCNP_Utilities.test_mcnp_cell_str1 ( )

8.32.1.8 def test_MCNP_Utilities.test_mcnp_cell_str2 ( )

8.32.1.9 def test_MCNP_Utilities.test_mcnp_cell_str3 ( )

8.32.1.10 def test_MCNP_Utilities.test_mcnp_cell_str4 ( )

8.32.1.11 def test_MCNP_Utilities.test_mcnp_geometry ( )

8.32.1.12 def test_MCNP_Utilities.test_mcnp_settings_read_settings ( )

8.32.1.13 def test_MCNP_Utilities.test_mcnp_settings_read_source ( )

8.32.1.14 def test_MCNP_Utilities.test_mcnp_surface1 ( )

8.32.1.15 def test_MCNP_Utilities.test_mcnp_surface10 ( )

8.32.1.16 def test_MCNP_Utilities.test_mcnp_surface2 ( )

8.32.1.17 def test_MCNP_Utilities.test_mcnp_surface3 ( )

8.32.1.18 def test_MCNP_Utilities.test_mcnp_surface4 ( )

8.32.1.19 def test_MCNP_Utilities.test_mcnp_surface5 ( )

8.32.1.20 def test_MCNP_Utilities.test_mcnp_surface6 ( )

8.32.1.21 def test_MCNP_Utilities.test_mcnp_surface7 ( )

8.32.1.22 def test_MCNP_Utilities.test_mcnp_surface8 ( )

8.32.1.23 def test_MCNP_Utilities.test_mcnp_surface9 ( )

8.32.1.24 def test_MCNP_Utilities.test_mcnp_surface_repr ( )

8.32.1.25 def test_MCNP_Utilities.test_mcnp_surface_str ( )

8.32.1.26 def test_MCNP_Utilities.test_Print_MCNP_Input ( )

8.32.1.27 def test_MCNP_Utilities.test_Read_MCNP_Output ( )

8.32.1.28 def test_MCNP_Utilities.test_Read_MCNP_Output2 ( )

8.32.1.29 def test_MCNP_Utilities.test_Read_Tally_Output ( )

8.32.2 Variable Documentation

8.32.2.1 string test_MCNP_Utilities.base_geom = 'MCNP geometry instance properties:\nMCNP Surfaces:\n500 TRC 0.00000 0.00000 16.12650 0.00000 0.00000 14.35147 \n 0.00001 5.16119 $inner debris cover\n\n501 TRC 0.00000 0.00000 15.24000 0.00000 0.00000 15.23797 \n 0.00001 5.48000 $outer debris cover\n\n502 TRC 0.00000 0.00000 30.77797 0.00000 0.00000 10.57235 \n 5.26908 9.07119 $inner cone\n\n503 TRC 0.00000 0.00000 30.47797 0.00000 0.00000 10.87235 \n 5.48000 9.39000 $outer cone\n\n504 RCC 0.00000 0.00000 41.35032 0.00000 0.00000 9.78968 \n 8.89000 $inner cylinder\n\n505 RCC 0.00000 0.00000 41.35032 0.00000 0.00000 9.78968 \n 9.39000 $outer cylinder\n\n506 RCC 0.00000 0.00000 51.14000 0.00000 0.00000 1.00000 \n 9.39000 $cover\n\n507 RCC 0.00000 0.00000 52.14000 0.00000 0.00000 2.40000 \n 5.63400 $adapter\n\nMCNP Cells:\n1 1 -2.70000e+00 500 -501 imp:n=1 imp:p=0 $\n\n2 1 -2.70000e+00 502 -503 imp:n=1 imp:p=0 $\n\n3 1 -2.70000e+00 504 -505 imp:n=1 imp:p=0 $\n\n4 1 -2.70000e+00 -506 imp:n=1 imp:p=0 $\n\n5 1 -2.70000e+00 -507 imp:n=1 imp:p=0 $\n\nMCNP Materials:\nAl\nZr\nZn\nIn\nTa\nAu\nPb\nFe\n'

8.32.2.2 tuple test_MCNP_Utilities.constraint_path = os.getcwd()

8.32.2.3 string test_MCNP_Utilities.mat_card = "C name: Air (dry near sea level)\nC density = 0.0\nm?\n 6012 -1.2256e-04\n 6013 -1.4365e-06\n 7014 -7.5527e-01\n 8016 -2.3178e-01\n 18036 -3.8527e-05\n 18038 -7.6673e-06\n 18040 -1.2781e-02\n"

8.32.2.4 tuple test_MCNP_Utilities.materials_library_path = os.getcwd()

8.32.2.5 tuple test_MCNP_Utilities.set_path = os.getcwd()

8.32.2.6 list test_MCNP_Utilities.source

Initial value:

```
= [[1.0E-8,0.9], [1.0E-7,0.5], [1.0E-6,1.0E-1], [1.24E-2,10],
             [2.345E+1,26.26]]
```

8.32.2.7 tuple test_MCNP_Utilities.src_path = os.getcwd()

8.32.2.8 string test_MCNP_Utilities.test_cell_repr = 'MCNP Cell:(1, mat=10, units=atom, density=0.0422, booleam geom=500 -501, n imp=1, p_imp=0, comment=)'

8.32.2.9 string test_MCNP_Utilities.test_cell_str1 = '1 10 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n'

8.32.2.10 string test_MCNP_Utilities.test_cell_str2 = '1 10 -4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n'

8.32.2.11 string test_MCNP_Utilities.test_cell_str3 = '1 10 500 -501 imp:n=1 imp:p=0 $\n'

8.32.2.12 string test_MCNP_Utilities.test_cell_str4 = '1 10 -4.22300e-02 (500 -501):(502 -503):(504 -505):(506 -507):(508\n -509):(509 -510) imp:n=1 imp:p=0 $\n'

8.32.2.13 string test_MCNP_Utilities.test_geom_str1 = 'MCNP geometry instance properties:\nMCNP Surfaces:\n509 TRC 1.00000 2.00000 3.00000 2.50000 23.60000 23.56000 \n 3.40000 1.00000 $one\n\n504 px 2.00000 $two\n\n505 Py -2.00000 $three\n\nMCNP Cells:\n1 11 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\n2 12 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\n3 13 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\nMCNP Materials:\nAir (dry near sea level)\nAl\n'

8.32.2.14 string test_MCNP_Utilities.test_surf_repr = 'MCNP Surface(600, TRC, vx=1.0, vy=2.0, vz=3.0, hx=2.5, hy=23.6, hz=23.56, r1=3.4, r2=1.0, c=test)'

8.32.2.15 string test_MCNP_Utilities.test_surf_str = '700 px 2.00000 $test\n'

8.33 test_Metaheuristics Namespace Reference

Functions

- def test_Mat_Levy_Flights

Variables

- tuple set_path = os.getcwd()
- tuple mat_path = os.getcwd()

8.33.1 Function Documentation

8.33.1.1 def test_Metaheuristics.test_Mat_Levy_Flights ( )

8.33.2 Variable Documentation

8.33.2.1 tuple test_Metaheuristics.mat_path = os.getcwd()

8.33.2.2 tuple test_Metaheuristics.set_path = os.getcwd()

8.34 test_NuclearData Namespace Reference

Functions

- def test_set_density1
- def test_set_density
- def test_strip_undesirables1
- def test_strip_undesirables2
- def test_build_matlib
- def test_Moderating_Ratio
- def test_Calc_Moderating_Ratio

Variables

- tuple mat_path = os.getcwd()

8.34.1 Function Documentation

8.34.1.1 def test_NuclearData.test_build_matlib ( )

8.34.1.2 def test_NuclearData.test_Calc_Moderating_Ratio ( )

8.34.1.3 def test_NuclearData.test_Moderating_Ratio ( )

8.34.1.4 def test_NuclearData.test_set_density ( )

8.34.1.5 def test_NuclearData.test_set_density1 ( )

8.34.1.6 def test_NuclearData.test_strip_undesirables1 ( )

8.34.1.7 def test_NuclearData.test_strip_undesirables2 ( )

8.34.2 Variable Documentation

8.34.2.1 tuple test_NuclearData.mat_path = os.getcwd()

8.35 test_Utilities Namespace Reference

Functions

- def test_cmd_thread
- def test_run_transport_pp
- def test_run_transport_threads
- def test_to_normdiff
- def test_Uopt
- def test_LeastSquares
- def test_RelativeLeastSquares

- def test_functhreadwithreturn
- def test_functhread
- def test_Event
- def test_Meta_Stats

Variables

- string test_mcnp = 'mcnp6 ../NSA_Proposal_ETA.inp NSA_Proposal_ETA.out'

8.35.1 Function Documentation

8.35.1.1 def test_Utilities.test_cmd_thread ( )

8.35.1.2 def test_Utilities.test_Event ( )

8.35.1.3 def test_Utilities.test_functhread ( )

8.35.1.4 def test_Utilities.test_functhreadwithreturn ( )

8.35.1.5 def test_Utilities.test_LeastSquares ( )

8.35.1.6 def test_Utilities.test_Meta_Stats ( )

8.35.1.7 def test_Utilities.test_RelativeLeastSquares ( )

8.35.1.8 def test_Utilities.test_run_transport_pp ( )

8.35.1.9 def test_Utilities.test_run_transport_threads ( )

8.35.1.10 def test_Utilities.test_to_normdiff ( )

8.35.1.11 def test_Utilities.test_Uopt ( )

8.35.2 Variable Documentation

8.35.2.1 string test_Utilities.test_mcnp = 'mcnp6 ../NSA_Proposal_ETA.inp NSA_Proposal_ETA.out'

8.36 UserInputs Namespace Reference

Classes

- class UserInputs

  The class creates a UserInputs object to store the user input file locations, read the user inputs, and set the appropriate classes required to run Coeus.

Variables

- tuple module_logger = logging.getLogger('Coeus.UserInputs')

8.36.1 Variable Documentation

8.36.1.1 tuple UserInputs.module_logger = logging.getLogger('Coeus.UserInputs')

8.37 Utilities Namespace Reference

Classes

- class Switch

  Creates a switch class object to switch between cases.
- class Cmd_Thread

  Creates a Thread class object to run command line programs in parallel.
- class FuncThread

  Creates a Thread class object to run functions without returns in parallel.
- class FuncThreadWithReturn

  Creates a Thread class object to run functions containing returns in parallel.
- class Event

  an event object representing a snapshot in the optimization process
- class WeightedRandomGenerator

  Defines a class of weights to be used to select number of instances in array randomly with linear weighting.
- class Meta_Stats

  Stores and prints effectiveness stats for each metaheuristic search method.

Functions

- def Run_Transport_Threads

  Runs a multi-threaded transport calculation.
- def Run_CmdLine

  A callable function to execute a command line program.
- def Run_Transport_PP

  Runs a multi-threaded transport calculation.
- def Run_Transport

  Build a Slurm Batch script using the Jobs Array feature to run transport calculations.
- def Build_Batch

  Build a Slurm Batch script using the Jobs Array feature to run transport calculations.
- def to_Norm

  Normalizes a MCNP tallied flux.
- def to_NormDiff

  Converts a MCNP tallied flux to a Normalized Differential flux.

Variables

- tuple module_logger = logging.getLogger('Coeus.Utilities')

8.37.1 Function Documentation

8.37.1.1 def Utilities.Build_Batch (*lst,* *tasks,* *code,* *qos,* *account,* *partition,* *timeout,* *suf* = " ")

Build a Slurm Batch script using the Jobs Array feature to run transport calculations.

Parameters

lst	list of parent identifier numbers to be ran
tasks	Number of tasks to run per code thread instance
code	[Default = 'mcnp6'] An indicator for which code to run (options = 'mcnp6', 'mcnp6.mpi', 'advantg')
suf	Optional string identifier suffix to be added at end of file

Returns

Filename for the batchfile created

8.37.1.2 def Utilities.Run_CmdLine (*cmd*, *cwdir*)

A callable function to execute a command line program.

Parameters

cwdir	Current working directory path
cmd	The command line input to be executed

8.37.1.3 def Utilities.Run_Transport (*lst*, *qos*, *account*, *partition*, *timeout*, *nps* = [], *code* = 'mcnp6')

Build a Slurm Batch script using the Jobs Array feature to run transport calculations.

Parameters

lst	list of parent identifier numbers to be ran
nps	list of number of particles to run per code thread instance. If left blank, calculation will be performed to assign all availiable cpus evenly
code	[Default = 'mcnp6'] An indicator for which code to run (options = 'mcnp6', 'mcnp6.mpi', 'advantg')

8.37.1.4 def Utilities.Run_Transport_PP (*lst*, *tasks* = 0, *code* = 'mcnp6')

Runs a multi-threaded transport calculation.

Doesn't work for clusters.

Parameters

lst	list of parent identifier numbers to be ran
tasks	Number of tasks to run per code thread instance. If left blank, calculation will be performed to assign all availiable cpus evenly
code	[Default = 'mcnp6'] An indicator for which code to run (options = 'mcnp6', 'mcnp6.mpi', 'advantg')

8.37.1.5 def Utilities.Run_Transport_Threads (*lst*, *tasks* = 0, *code* = 'mcnp6')

Runs a multi-threaded transport calculation.

Doesn't work for clusters.

Parameters

lst	list of parent identifier number to be ran
tasks	Number of tasks to run per code thread instance. If left blank, calculation will be performed to assign all availiable cpus evenly
code	[Default = 'mcnp6'] An indicator for which code to run (options = 'mcnp6', 'mcnp6.mpi', 'advantg')

8.37.1.6 def Utilities.to_Norm (*spectrum*)

Normalizes a MCNP tallied flux.

Parameters

spectrum	array Teh input flux spectrum

Returns

result array the output normalized differential flux spectrum

8.37.1.7 def Utilities.to_NormDiff (*spectrum*)

Converts a MCNP tallied flux to a Normalized Differential flux.

Parameters

spectrum	The input flux spectrum

Returns

The output normalized differential flux spectrum

8.37.2 Variable Documentation

8.37.2.1 tuple Utilities.module_logger = logging.getLogger('Coeus.Utilities')

Chapter 9

Class Documentation

9.1 ADVANTG_Utilities.ADVANTG_Settings Class Reference

Public Member Functions

- def __init__

  Creates a object representing the settings for running the ADVANTG deterministic radiation trasport code calculations.
- def __repr__
- def __str__
- def read_settings

  Parses a ADVANTG settings csv input file.

Public Attributes

- lib

  string The multi-group library used [Default: "dplus"]
- method

  string The solution method for ADVANTG (CADIS or DX) [Default: "cadis"]
- outputs

  string The output files to be produced [Default: "mcnp"]
- tnum

  int The tally number for calculating the adjoint flux [Default: 24]
- pt_src

  string Whether or not the source should be treated as a point source in deterministic transport [Default: False]
- mix_tol

  string The material mix tolerance fraction.
- pn_order

  integer The scattering order [Default: 1]
- eta_x

  float The spacing of the mesh intervals in the x (radial) axis in cm in the ETA.
- eta_y

  float The spacing of the mesh intervals in the y (radial) axis in cm in the ETA.
- eta_z

  float The spacing of the mesh intervals in the z (axial) axis in cm in the ETA.
- foil_x

  float The spacing of the mesh intervals in the x (radial) axis in cm near the foil.
- foil_y

float The spacing of the mesh intervals in the y (radial) axis in cm near the foil.

- foil_z

  float The spacing of the mesh intervals in the z (axial) axis in cm near the foil.
- ext

  float The spacing of the mesh intervals in x,y, and z axis in cm outside the ETA.
- f

9.1.1 Constructor & Destructor Documentation

9.1.1.1 def ADVANTG_Utilities.ADVANTG_Settings.__init__ (*self*, *lib* = "dplus", *method* = "cadis", *outputs* = "mcnp silo", *tnum* = 24, *pt_src* = "True", *mix_tol* = 0.01, *pn_order* = 1, *eta_x* = 0.5, *eta_y* = 0.5, *eta_z* = 0.5, *foil_x* = 0.25, *foil_y* = 0.25, *foil_z* = 0.05, *ext_spacing* = 1.0)

Creates a object representing the settings for running the ADVANTG deterministic radiation trasport code calculations.

9.1.2 Member Function Documentation

9.1.2.1 def ADVANTG_Utilities.ADVANTG_Settings.__repr__ (*self*)

9.1.2.2 def ADVANTG_Utilities.ADVANTG_Settings.__str__ (*self*)

9.1.2.3 def ADVANTG_Utilities.ADVANTG_Settings.read_settings (*self*, *filename*)

Parses a ADVANTG settings csv input file.

The key word options are: Library Method Outputs Tally Number Point Source Material Mix Tolerance Scattering Order ETA X Spacing Interval ETA Y Spacing Interval ETA Z Spacing Interval Foil X Spacing Interval Foil Y Spacing Interval Foil Z Spacing Interval External Spacing Interval

9.1.3 Member Data Documentation

9.1.3.1 ADVANTG_Utilities.ADVANTG_Settings.eta_x

float The spacing of the mesh intervals in the x (radial) axis in cm in the ETA.

[Default: 0.5 cm]

9.1.3.2 ADVANTG_Utilities.ADVANTG_Settings.eta_y

float The spacing of the mesh intervals in the y (radial) axis in cm in the ETA.

[Default: 0.5 cm]

9.1.3.3 ADVANTG_Utilities.ADVANTG_Settings.eta_z

float The spacing of the mesh intervals in the z (axial) axis in cm in the ETA.

[Default: 0.5 cm]

9.1.3.4 ADVANTG_Utilities.ADVANTG_Settings.ext

float The spacing of the mesh intervals in x,y, and z axis in cm outside the ETA.

[Default: 1 cm]

9.1.3.5 ADVANTG_Utilities.ADVANTG_Settings.f

9.1.3.6 ADVANTG_Utilities.ADVANTG_Settings.foil_x

float The spacing of the mesh intervals in the x (radial) axis in cm near the foil.

[Default: 0.25 cm]

9.1.3.7 ADVANTG_Utilities.ADVANTG_Settings.foil_y

float The spacing of the mesh intervals in the y (radial) axis in cm near the foil.

[Default: 0.25 cm]

9.1.3.8 ADVANTG_Utilities.ADVANTG_Settings.foil_z

float The spacing of the mesh intervals in the z (axial) axis in cm near the foil.

[Default: 0.05 cm]

9.1.3.9 ADVANTG_Utilities.ADVANTG_Settings.lib

string The multi-group library used [Default: "dplus"]

9.1.3.10 ADVANTG_Utilities.ADVANTG_Settings.method

string The solution method for ADVANTG (CADIS or DX) [Default: "cadis"]

9.1.3.11 ADVANTG_Utilities.ADVANTG_Settings.mix_tol

string The material mix tolerance fraction.

Controls the precision of mixed cells. [Default: 0.01]

9.1.3.12 ADVANTG_Utilities.ADVANTG_Settings.outputs

string The output files to be produced [Default: "mcnp"]

9.1.3.13 ADVANTG_Utilities.ADVANTG_Settings.pn_order

integer The scattering order [Default: 1]

9.1.3.14 ADVANTG_Utilities.ADVANTG_Settings.pt_src

string Whether or not the source should be treated as a point source in deterministic transport [Default: False]

9.1.3.15 ADVANTG_Utilities.ADVANTG_Settings.tnum

int The tally number for calculating the adjoint flux [Default: 24]

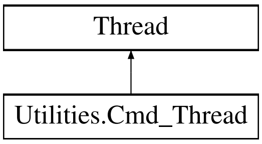
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/ADVANTG_Utilities.py

9.2 Utilities.Cmd_Thread Class Reference

Creates a Thread class object to run command line programs in parallel.

Inheritance diagram for Utilities.Cmd_Thread:



Public Member Functions

- def __init__

  The constructor.
- def __repr__
- def __str__
- def run

  Run Thread in local working directory.

Public Attributes

- cwdir

  Current working directory path.
- cmd

  The command line input to be executed.

9.2.1 Detailed Description

Creates a Thread class object to run command line programs in parallel.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 def Utilities.Cmd_Thread.__init__ (*self, cwdir, cmd*)

The constructor.

Parameters

cwdir	Current working directory path
cmd	The command line input to be executed

9.2.3 Member Function Documentation

9.2.3.1 def Utilities.Cmd_Thread.__repr__ (*self*)

9.2.3.2 def Utilities.Cmd_Thread.__str__ (*self*)

9.2.3.3 def Utilities.Cmd_Thread.run (*self*)

Run Thread in local working directory.

9.2.4 Member Data Documentation

9.2.4.1 Utilities.Cmd_Thread.cmd

The command line input to be executed.

9.2.4.2 Utilities.Cmd_Thread.cwdir

Current working directory path.

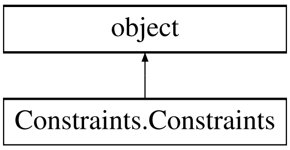
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py

9.3 Constraints.Constraints Class Reference

The class creates a Constraints object that can be used in optimization algorithms.

Inheritance diagram for Constraints.Constraints:



Public Member Functions

- def __init__

  Constructor to build the ObjectiveFunction class.
- def __repr__

  Constraint class param print function.
- def __str__

  Human readable Constraint print function.
- def set_constraint_func

  Converts an input string name for a function to a function handle.
- def get_penalty

  Calculate the constraint violation penalty, if any.
- def less_or_equal

  Compares a previously calculated value to a user specifed maximum.
- def greater_than

  Compares the calculated value to the minimum specified by the user.

Public Attributes

- func
- constraint

  is violated
- tallyNum
- penalty

9.3.1 Detailed Description

The class creates a Constraints object that can be used in optimization algorithms.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 def Constraints.Constraints.__init__ (*self*, *method* = `None`, *constraint* = `None`, *tallyNum* = `None`, *penalty* = `1E15`)

Constructor to build the ObjectiveFunction class.

Parameters

self	*object pointer* The object pointer.
_FUNCT_DICT	*dictionary* A mapping from string function names to function handles.
method	*string* The name of the constraint function to evaluate.
constraint	*float* The constraint to be compared against.
tallyNum	*integer* The tally number associated with the constraint.
penalty	*float* The penalty to be applied if a constraint is violated. 1E15 is recommended.

9.3.3 Member Function Documentation

9.3.3.1 def Constraints.Constraints.__repr__ (*self*)

Constraint class param print function.

Parameters

self	*pointer* The Constraint pointer.

9.3.3.2 def Constraints.Constraints.__str__ (*self*)

Human readable Constraint print function.

Parameters

self	*pointer* The Constraint pointer.

9.3.3.3 def Constraints.Constraints.get_penalty (*self, violation*)

Calculate the constraint violation penalty, if any.

Parameters

self	*pointer* The Constraint pointer.
violation	*float* The magnitude of the constraint violation used for scaling the penalty.

Returns

float: The scaled penalty.

9.3.3.4 def Constraints.Constraints.greater_than (*self, candidate*)

Compares the calculated value to the minimum specified by the user.

Parameters

self	*pointer* The Constraint pointer.
candidate	*float* The calculated value corresponding to a candidate design.

Returns

float: The penalty associated with the candidate design.

9.3.3.5 def Constraints.Constraints.less_or_equal (*self, candidate*)

Compares a previously calculated value to a user specifed maximum.

Parameters

self	*pointer* The Constraint pointer.
candidate	*float* The calculated value corresponding to a candidate design.

Returns

float: The penalty associated with the candidate design.

9.3.3.6 def Constraints.Constraints.set_constraint_func (*self, funcName*)

Converts an input string name for a function to a function handle.

Parameters

self	*pointer* The Constraint pointer.
funcName	*string* A string identifying the constraint function to be used.

9.3.4 Member Data Documentation

9.3.4.1 Constraints.Constraints.constraint

is violated

9.3.4.2 Constraints.Constraints.func

9.3.4.3 Constraints.Constraints.penalty

9.3.4.4 Constraints.Constraints.tallyNum

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Constraints.py

9.4 ETA_Utilities.ETA_Parameters Class Reference

Public Member Functions

- def __init__

  Creates an ETA object that stores the ETA design parameters, constraints, and objective function.
- def __repr__
- def __str__
- def read_constraints

  Parses an ETA constraints csv file.

Public Attributes

- max_weight

  float The maximum weight for the ETA assembly entered in kg [default=125.0 [kg]]
- src

  float NIF source neutron in 4pi [default=5E15]
- tcc_dist

  float The distance from the front face of the ETA assembly from target chamber center (TCC) entered in cm [default=15.24 [cm]]
- t_ds

  float The thickness of the debris cover and conical section of the ETA assembly entered in cm [default=0.3 [cm]]
- t_w

  float The thickness of the walls of the ETA assembly entered in cm [default=0.25 [cm]]
- snout_dist

  float The distance to where the snout mounts in cm.
- t_c

  float The thickness of the back cover for the ETA in cm [default=1.0 [cm]]

- t_m

  float The thickness of the mount connecting the ETA Nose Cone Assembly to the snout in cm [default=2.4 [cm]]
- r_f

  float The opening radius of the ETA assembly entered in cm.
- r_o

  float The maximum outer radius of the ETA assembly structure entered in cm.
- theta

  float The opening angle of the cone measured in degrees.
- ds_mat

  string The material used for the conical debris cover.
- struct_mat

  string The material used for the ETA structure.
- fill_mat

  string The material used to fill all nonspecified volume in the in the ETA.
- fissile_mat

  string The fissile material used in the ETA.
- t_nas

  float The thickness of the neutron activation spectrometer in cm [default= 0.14 cm]
- r_nas

  float The radius of the neutron activation spectrometer in cm.
- nas_mat

  string The material used for the NAS structure.
- t_nas_f

  list of floats The thickness of the neutron activation spectrometer foils in cm [default= [0.1, 0.1, 0.1, 0.1, 0.01] cm]
- r_nas_f

  float The radius of the neutron activation spectrometer foils entered in cm.
- nas_mat_f

  list of strings The material used for the neutron activation spectrometer foils in the in the ETA.
- toad_loc

  str String indicating the material that the TOAD assembly follows after in the stackup.
- toad_mat

  string The material used for the TOAD structure.
- t_toad

  list of floats The thickness of the TOAD foils in cm [default= [,0.0127] cm]
- r_toad

  float The radius of foils entered in cm.
- toad_mat_f

  list of strings The material used for the neutron activation spectrometer foils in the in the ETA.
- holder_mat

  string The material used for the holder structure.
- h_fill_mat

  string The material used to fill the space int he holder structrure.
- t_h

  float The thickness of the holder for the NAS insertion assemblyentered in cm.
- max_vert

  int The maximum number of vertical macrobodies or components in the ETA geometry.
- max_horiz

  int The maximum number of horizontal macrobodies or components in the ETA geometry.
- min_fiss

  float The minimum number of fissions in fissile foil [default=5e8]
- f

9.4.1 Constructor & Destructor Documentation

9.4.1.1 def ETA_Utilities.ETA_Parameters.__init__ (*self*, *min_fiss* = 5E8, *max_weight* = 125.0, *src* = 5E15, *tcc_dist* = 15.24, *debris_shield_thickness* = 0.3, *wall_thickness* = 0.5, *snout_dist* = 52.14, *cover_thickness* = 1.0, *mount_thickness* = 2.4, *face_radius* = 5.48, *eta_or* = 9.39, *cone_angle* = 70.22, *debris_shield_mat* = "Al", *struct_mat* = "Al", *fill_mat* = "Air (dry near sea level)", *fissile_mat* = 'Pb', *nas_th* = 0.014, *nas_r* = 2.69, *nas_mat* = 'Al', *nas_foil_th* = [0.1, *nas_foil_r* = 2.5, *nas_foil_mat* = ['Zr', *Zn*, *In*, *Al*, *Ta*, *toad_loc* = 'In', *toad_mat* = 'Al', *toad_foil_th* = [0.0254, *toad_foil_r* = 1.252, *toad_foil_mat* = ['Au', *Pb*, *holder_mat* = 'Al', *holder_fill_mat* = 'Fe', *holder_thickness* = 2.0, *max_vert* = 3, *max_horiz* = 7)

Creates an ETA object that stores the ETA design parameters, constraints, and objective function.

9.4.2 Member Function Documentation

9.4.2.1 def ETA_Utilities.ETA_Parameters.__repr__ (*self*)

9.4.2.2 def ETA_Utilities.ETA_Parameters.__str__ (*self*)

9.4.2.3 def ETA_Utilities.ETA_Parameters.read_constraints (*self*, *filename*)

Parses an ETA constraints csv file.

The key word options are: Minimum Fissions ETA Max Weight Source Strength

9.4.3 Member Data Documentation

9.4.3.1 ETA_Utilities.ETA_Parameters.ds_mat

string The material used for the conical debris cover.

Must be a naturally occuring element or specified in the materials compendium. [default="Al"]

9.4.3.2 ETA_Utilities.ETA_Parameters.f

9.4.3.3 ETA_Utilities.ETA_Parameters.fill_mat

string The material used to fill all nonspecified volume in the in the ETA.

Must be a naturally occuring element or specified in the materials compendium. [default="Air (dry near sea level)"]

9.4.3.4 ETA_Utilities.ETA_Parameters.fissile_mat

string The fissile material used in the ETA.

Must be a naturally occuring element or specified in the materials compendium. [default='U']

9.4.3.5 ETA_Utilities.ETA_Parameters.h_fill_mat

string The material used to fill the space int he holder structrure.

Must be a naturally occuring element or specified in the materials compendium. [default="Fe"]

9.4.3.6 ETA_Utilities.ETA_Parameters.holder_mat

string The material used for the holder structure.

Must be a naturally occuring element or specified in the materials compendium. [default="Al"]

9.4.3.7 ETA_Utilities.ETA_Parameters.max_horiz

int The maximum number of horizontal macrobodies or components in the ETA geometry.

This cn be reduced to increase run spead or increased to obtain a better result. [default=7]

9.4.3.8 ETA_Utilities.ETA_Parameters.max_vert

int The maximum number of vertical macrobodies or components in the ETA geometry.

This cn be reduced to increase run spead or increased to obtain a better result. [default=3]

9.4.3.9 ETA_Utilities.ETA_Parameters.max_weight

float The maximum weight for the ETA assembly entered in kg [default=125.0 [kg]]

9.4.3.10 ETA_Utilities.ETA_Parameters.min_fiss

float The minimum number of fissions in fissile foil [default=5e8]

9.4.3.11 ETA_Utilities.ETA_Parameters.nas_mat

string The material used for the NAS structure.

Must be a naturally occuring element or specified in the materials compendium. [default="Al"]

9.4.3.12 ETA_Utilities.ETA_Parameters.nas_mat_f

list of strings The material used for the neutron activation spectrometer foils in the in the ETA.

Must be a naturally occuring element or specified in the materials compendium. [default=['Zr', 'Zn', 'In', 'Al', 'Ta']]

9.4.3.13 ETA_Utilities.ETA_Parameters.r_f

float The opening radius of the ETA assembly entered in cm.

Measured from ETA centerline [default=5.48 [cm]]

9.4.3.14 ETA_Utilities.ETA_Parameters.r_nas

float The radius of the neutron activation spectrometer in cm.

[default= 2.69 [cm]]

9.4.3.15 ETA_Utilities.ETA_Parameters.r_nas_f

float The radius of the neutron activation spectrometer foils entered in cm.

[default= 2.5 [cm]]

9.4.3.16 ETA_Utilities.ETA_Parameters.r_o

float The maximum outer radius of the ETA assembly structure entered in cm.

Measured from ETA centerline [default=9.39 [cm]]

9.4.3.17 ETA_Utilities.ETA_Parameters.r_toad

float The radius of foils entered in cm.

[default= 1.252 [cm]]

9.4.3.18 ETA_Utilities.ETA_Parameters.snout_dist

float The distance to where the snout mounts in cm.

Measured from target chamber center (TCC) [default=52.14 [cm]]

9.4.3.19 ETA_Utilities.ETA_Parameters.src

float NIF source neutron in 4pi [default=5E15]

9.4.3.20 ETA_Utilities.ETA_Parameters.struct_mat

string The material used for the ETA structure.

Must be a naturally occuring element or specified in the materials compendium. [default="Al"]

9.4.3.21 ETA_Utilities.ETA_Parameters.t_c

float The thickness of the back cover for the ETA in cm [default=1.0 [cm]]

9.4.3.22 ETA_Utilities.ETA_Parameters.t_ds

float The thickness of the debris cover and conical section of the ETA assembly entered in cm [default=0.3 [cm]]

9.4.3.23 ETA_Utilities.ETA_Parameters.t_h

float The thickness of the holder for the NAS insertion assemblyentered in cm.

[default= 2 [cm]]

9.4.3.24 ETA_Utilities.ETA_Parameters.t_m

float The thickness of the mount connecting the ETA Nose Cone Assembly to the snout in cm [default=2.4 [cm]]

9.4.3.25 ETA_Utilities.ETA_Parameters.t_nas

float The thickness of the neutron activation spectrometer in cm [default= 0.14 cm]

9.4.3.26 ETA_Utilities.ETA_Parameters.t_nas_f

list of floats The thickness of the neutron activation spectrometer foils in cm [default= [0.1, 0.1, 0.1, 0.1, 0.01] cm]

9.4.3.27 ETA_Utilities.ETA_Parameters.t_toad

list of floats The thickness of the TOAD foils in cm [default= [,0.0127] cm]

9.4.3.28 ETA_Utilities.ETA_Parameters.t_w

float The thickness of the walls of the ETA assembly entered in cm [default=0.25 [cm]]

9.4.3.29 ETA_Utilities.ETA_Parameters.tcc_dist

float The distance from the front face of the ETA assembly from target chamber center (TCC) entered in cm [default=15.24 [cm]]

9.4.3.30 ETA_Utilities.ETA_Parameters.theta

float The opening angle of the cone measured in degrees.

Measured from ETA face plane. [default=8.89 [cm]]

9.4.3.31 ETA_Utilities.ETA_Parameters.toad_loc

str String indicating the material that the TOAD assembly follows after in the stackup.

[default=['In']

9.4.3.32 ETA_Utilities.ETA_Parameters.toad_mat

string The material used for the TOAD structure.

Must be a naturally occuring element or specified in the materials compendium. [default="Al"]

9.4.3.33 ETA_Utilities.ETA_Parameters.toad_mat_f

list of strings The material used for the neutron activation spectrometer foils in the in the ETA.

Must be a naturally occuring element or specified in the materials compendium. [default=['Au','U']]

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/ETA_Utilities.py

9.5 Utilities.Event Class Reference

an event object representing a snapshot in the optimization process

Public Member Functions

- def __init__

  Creates an event object representing a snapshot in the optimization process.
- def __repr__
- def __str__

Public Attributes

- g

  The generation the design was arrived at.
- e

  The number of fitness evaluations done to obtain this design.
- f

  The assessed design fitness.
- n

  The number of particles run for that event.
- i

  The identifty of the current top solution.

9.5.1 Detailed Description

an event object representing a snapshot in the optimization process

9.5.2 Constructor & Destructor Documentation

9.5.2.1 def Utilities.Event.__init__ (*self*, *generation*, *evaluations*, *fitness*, *nps*, *ident*)

Creates an event object representing a snapshot in the optimization process.

Returns

None

9.5.3 Member Function Documentation

9.5.3.1 def Utilities.Event.__repr__ (*self*)

9.5.3.2 def Utilities.Event.__str__ (*self*)

9.5.4 Member Data Documentation

9.5.4.1 Utilities.Event.e

The number of fitness evaluations done to obtain this design.

9.5.4.2 Utilities.Event.f

The assessed design fitness.

9.5.4.3 Utilities.Event.g

The generation the design was arrived at.

9.5.4.4 Utilities.Event.i

The identifty of the current top solution.

9.5.4.5 Utilities.Event.n

The number of particles run for that event.

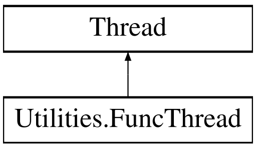
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py

9.6 Utilities.FuncThread Class Reference

Creates a Thread class object to run functions without returns in parallel.

Inheritance diagram for Utilities.FuncThread:



Public Member Functions

- def __init__

  The constructor.
- def run

9.6.1 Detailed Description

Creates a Thread class object to run functions without returns in parallel.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 def Utilities.FuncThread.__init__ (*self, target, args*)

The constructor.

Parameters

target	The function to be executed
args	The functions arguments

9.6.3 Member Function Documentation

9.6.3.1 def Utilities.FuncThread.run (*self*)

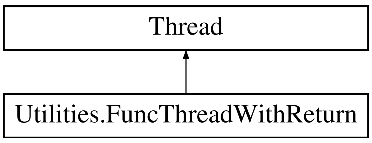
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py

9.7 Utilities.FuncThreadWithReturn Class Reference

Creates a Thread class object to run functions containing returns in parallel.

Inheritance diagram for Utilities.FuncThreadWithReturn:



Public Member Functions

- def __init__
- def run
- def join

9.7.1 Detailed Description

Creates a Thread class object to run functions containing returns in parallel.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 def Utilities.FuncThreadWithReturn.__init__ (*self, args, kwargs*)

9.7.3 Member Function Documentation

9.7.3.1 def Utilities.FuncThreadWithReturn.join (*self, args, kwargs*)

9.7.3.2 def Utilities.FuncThreadWithReturn.run (*self*)

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py

9.8 Gnowee_Utilities.Gnowee_Settings Class Reference

Public Member Functions

- def __init__

  Creates an object representing the settings for the optimization algorithm.
- def __repr__
- def __str__
- def read_settings

  Parses a Gnowee settings csv input file.

Public Attributes

- p

  integer The number of parents in each generation [Default: 25]
- s

  string The method used to sample the phase space and create the initial population Valid('random','nolh','nolh-rp','nolh-cdr',and 'lhc') [Default: 'LHC']

- fd

  float Discovery probability [Default: 0.25]
- fe

  float Elite fraction of population [Default: 0.20]
- fl

  float Fraction of Levy population engaging in Levy flights in a given generation [Default: 0.40]
- gm

  integer The maximum number of generations to search [Default: 10000]
- em

  integer The maximum number of objective function evaluations [Default: 100000]
- ct

  float The minimum change of the best objective value before the search terminates [Default: 1e-6]
- sl

  integer The maximum number of genrations to search without a decrease exceeding conv_tol [Default: 200]
- of

  float The best know fitness value for the problem considered [Default: 0.0]
- ot

  float The maximum deviation from the best know fitness value before the search terminates [Default: 1e-2]
- a

  float Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process [Default: 1.5]
- g

  float Gamma - Scale unit of process for Levy flights [Default: 1.0]
- n

  integer Number of independent variables - can be used to reduce Levy flight variance [Default: 1]
- sf

  scalar Step size scaling factor used to adjust Levy flights to length scale of system [Default: 10]
- f

9.8.1 Constructor & Destructor Documentation

9.8.1.1 def Gnowee_Utilities.Gnowee_Settings.__init__ (*self*, *population* = `25`, *initial_sampling* = `'lhc'`, *frac_discovered* = `0.25`, *frac_elite* = `0.20`, *frac_levy* = `0.4`, *max_gens* = `10000`, *feval_max* = `100000`, *conv_tol* = `1e-6`, *stall_iter_limit* = `200`, *optimal_fitness* = `0.01`, *opt_conv_tol* = `1e-2`, *alpha* = `1.5`, *gamma* = `1.0`, *n* = `1`, *scaling_factor* = `10.0`)

Creates an object representing the settings for the optimization algorithm.

9.8.2 Member Function Documentation

9.8.2.1 def Gnowee_Utilities.Gnowee_Settings.__repr__ (*self*)

9.8.2.2 def Gnowee_Utilities.Gnowee_Settings.__str__ (*self*)

9.8.2.3 def Gnowee_Utilities.Gnowee_Settings.read_settings (*self*, *filename*)

Parses a Gnowee settings csv input file.

The key word options are: Population Size Initial Sampling Method Discovery Fraction Elite Fraction Levy Fraction Max Generations Max Function Evaluations Stall Convergence Tolerance Stall Iteration Limit Optimal Fitness Optimal Convergence Tolerance Levy Alpha Levy Gamma Levy Indepentent Variables Step Size Scaling Factor

9.8.3 Member Data Documentation

9.8.3.1 Gnowee_Utilities.Gnowee_Settings.a

float Levy exponent - defines the index of the distribution and controls scale properties of the stochastic process [Default: 1.5]

9.8.3.2 Gnowee_Utilities.Gnowee_Settings.ct

float The minimum change of the best objective value before the search terminates [Default: 1e-6]

9.8.3.3 Gnowee_Utilities.Gnowee_Settings.em

integer The maximum number of objective function evaluations [Default: 100000]

9.8.3.4 Gnowee_Utilities.Gnowee_Settings.f

9.8.3.5 Gnowee_Utilities.Gnowee_Settings.fd

float Discovery probability [Default: 0.25]

9.8.3.6 Gnowee_Utilities.Gnowee_Settings.fe

float Elite fraction of population [Default: 0.20]

9.8.3.7 Gnowee_Utilities.Gnowee_Settings.fl

float Fraction of Levy population engaging in Levy flights in a given generation [Default: 0.40]

9.8.3.8 Gnowee_Utilities.Gnowee_Settings.g

float Gamma - Scale unit of process for Levy flights [Default: 1.0]

9.8.3.9 Gnowee_Utilities.Gnowee_Settings.gm

integer The maximum number of generations to search [Default: 10000]

9.8.3.10 Gnowee_Utilities.Gnowee_Settings.n

integer Number of independent variables - can be used to reduce Levy flight variance [Default: 1]

9.8.3.11 Gnowee_Utilities.Gnowee_Settings.of

float The best know fitness value for the problem considered [Default: 0.0]

9.8.3.12 Gnowee_Utilities.Gnowee_Settings.ot

float The maximum deviation from the best know fitness value before the search terminates [Default: 1e-2]

9.8.3.13 Gnowee_Utilities.Gnowee_Settings.p

integer The number of parents in each generation [Default: 25]

9.8.3.14 Gnowee_Utilities.Gnowee_Settings.s

string The method used to sample the phase space and create the initial population Valid('random','nolh','nolh-rp','nolh-cdr',and 'lhc') [Default: 'LHC']

9.8.3.15 Gnowee_Utilities.Gnowee_Settings.sf

scalar Step size scaling factor used to adjust Levy flights to length scale of system [Default: 10]

9.8.3.16 Gnowee_Utilities.Gnowee_Settings.sl

integer The maximum number of genrations to search without a decrease exceeding conv_tol [Default: 200]

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Gnowee_Utilities.py

9.9 MCNP_Utilities.MCNP_Cell Class Reference

Public Member Functions

- def __init__

  Creates a MCNP cell object.
- def __repr__
- def __str__

Public Attributes

- name

  int Cell number
- m

  int Material number.
- units

  string Acceptable values are "atom", "mass", and "void".
- d

  float The density of the cell
- geom

  string Specification of the Boolean geometry of the cell.
- imp

  int tuple Specification of the importance of the regions for (neutron, photons)
- comment

  string Comment describing the surface feature.

9.9.1 Constructor & Destructor Documentation

9.9.1.1 def MCNP_Utilities.MCNP_Cell.__init__ (*self, name, mat, units, dens, geom, imp, comment = ''*)

Creates a MCNP cell object.

9.9.2 Member Function Documentation

9.9.2.1 def MCNP_Utilities.MCNP_Cell.__repr__ (*self*)

9.9.2.2 def MCNP_Utilities.MCNP_Cell.__str__ (*self*)

9.9.3 Member Data Documentation

9.9.3.1 MCNP_Utilities.MCNP_Cell.comment

string Comment describing the surface feature.

Can be used to find the surface corresponding to a particular geometric feature [Default=”]

9.9.3.2 MCNP_Utilities.MCNP_Cell.d

float The density of the cell

9.9.3.3 MCNP_Utilities.MCNP_Cell.geom

string Specification of the Boolean geometry of the cell.

9.9.3.4 MCNP_Utilities.MCNP_Cell.imp

int tuple Specification of the importance of the regions for (neutron, photons)

9.9.3.5 MCNP_Utilities.MCNP_Cell.m

int Material number.

0 for a void cell

9.9.3.6 MCNP_Utilities.MCNP_Cell.name

int Cell number

9.9.3.7 MCNP_Utilities.MCNP_Cell.units

string Acceptable values are "atom", "mass", and "void".

Capitalization does not matter

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/MCNP_Utilities.py

9.10 MCNP_Utilities.MCNP_Geometry Class Reference

Public Member Functions

- def __init__

  Creates the geometry for running the MCNP radiation trasport code.
- def __repr__

- def __str__
- def init_geom

  Builds the inital surface list, cells dictionary, and materials list for the ETA geometry envelope.
- def fin_geom

  Finishes the geometry by adding the filler and void cells, surfaces, and materials to the geometry.
- def add_surf

  Adds new surface object to geometry surface list.
- def add_cell

  Adds new cell object to geometry cells list.
- def add_matls

  Add materials to the matls list.
- def read_geom

  Builds the geometry object from an MCNP input file.

Public Attributes

- surfaces
- cells
- matls

9.10.1 Constructor & Destructor Documentation

9.10.1.1 def MCNP_Utilities.MCNP_Geometry.__init__ (*self*)

Creates the geometry for running the MCNP radiation trasport code.

9.10.2 Member Function Documentation

9.10.2.1 def MCNP_Utilities.MCNP_Geometry.__repr__ (*self*)

9.10.2.2 def MCNP_Utilities.MCNP_Geometry.__str__ (*self*)

9.10.2.3 def MCNP_Utilities.MCNP_Geometry.add_cell (*self, adds*)

Adds new cell object to geometry cells list.

Parameters

adds	A list of the cell objects to add

9.10.2.4 def MCNP_Utilities.MCNP_Geometry.add_matls (*self, mat_lib, adds*)

Add materials to the matls list.

Checks for materials existing in the materials library.

Parameters

mat_lib	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes

adds	A list of the names of the materials to add to the matls list

9.10.2.5 def MCNP_Utilities.MCNP_Geometry.add_surf (*self, adds*)

Adds new surface object to geometry surface list.

Parameters

add	A list of the surface objects to add

9.10.2.6 def MCNP_Utilities.MCNP_Geometry.fin_geom (*self, eta, mats*)

Finishes the geometry by adding the filler and void cells, surfaces, and materials to the geometry.

Parameters

eta	[ETA parameters object] An object that contains all of the constraints required to initialize the geometry
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

9.10.2.7 def MCNP_Utilities.MCNP_Geometry.init_geom (*self, eta, mats*)

Builds the inital surface list, cells dictionary, and materials list for the ETA geometry envelope.

Parameters

eta	[ETA parameters object] An object that contains all of the constraints required to initialize the geometry
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

9.10.2.8 def MCNP_Utilities.MCNP_Geometry.read_geom (*self, path, mats*)

Builds the geometry object from an MCNP input file.

Fairly specific to the current ETA design.

Parameters

path	String The path, including filename, to the MCNP output file to be read
mats	[dictionary of material objects] A materials library containing all relevant nulcear data required to run radiation transport codes. Isotopic densities are in atoms/b-cm

Returns

nps integer Number of particles for the MCNP run

9.10.3 Member Data Documentation

9.10.3.1 MCNP_Utilities.MCNP_Geometry.cells

9.10.3.2 MCNP_Utilities.MCNP_Geometry.matls

9.10.3.3 MCNP_Utilities.MCNP_Geometry.surfaces

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/MCNP_Utilities.py

9.11 MCNP_Utilities.MCNP_Settings Class Reference

Public Member Functions

- def __init__

  Creates a object representing the settings for running the MCNP radiation trasport code.
- def __repr__
- def __str__
- def read_settings

  Parses a MCNP settings csv input file.
- def read_source

  Parses an source input csv file.
- def set_tallies

  Sets the standard tallies to be used.

Public Attributes

- phys

  str The physics cards for run parameters
- nps

  int The starting number of particles to run.
- source

  array Stores the upper energy bin bounds and source strength for each bin [default=[]]
- tally

  str The tallies for the problem.
- f

9.11.1 Constructor & Destructor Documentation

9.11.1.1 def MCNP_Utilities.MCNP_Settings.__init__ (*self*, *physics* = "MODE n\n", *nps* = 1E6, *tally* = "", *source* = [])

Creates a object representing the settings for running the MCNP radiation trasport code.

9.11.2 Member Function Documentation

9.11.2.1 def MCNP_Utilities.MCNP_Settings.__repr__ (*self*)

9.11.2.2 def MCNP_Utilities.MCNP_Settings.__str__ (*self*)

9.11.2.3 def MCNP_Utilities.MCNP_Settings.read_settings (*self*, *filename*)

Parses a MCNP settings csv input file.

The key word options are: Physics NPS

9.11.2.4 def MCNP_Utilities.MCNP_Settings.read_source (*self*, *filename*)

Parses an source input csv file.

The first column contains the upper energy bin boundaries. The second column contains the flux/fluence of the bin.

9.11.2.5 def MCNP_Utilities.MCNP_Settings.set_tallies (*self*, *cell*, *mat*)

Sets the standard tallies to be used.

Parameters

cell	int the cell for volume tallies
mat	int the amterial number for reaction tallies

9.11.3 Member Data Documentation

9.11.3.1 MCNP_Utilities.MCNP_Settings.f

9.11.3.2 MCNP_Utilities.MCNP_Settings.nps

int The starting number of particles to run.

The number ran by the code will depend on generation and fitness. [Default: 1E6]

9.11.3.3 MCNP_Utilities.MCNP_Settings.phys

str The physics cards for run parameters

9.11.3.4 MCNP_Utilities.MCNP_Settings.source

array Stores the upper energy bin bounds and source strength for each bin [default=[]]

9.11.3.5 MCNP_Utilities.MCNP_Settings.tally

str The tallies for the problem.

[Default: ""]

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/MCNP_Utilities.py

9.12 MCNP_Utilities.MCNP_Surface Class Reference

Public Member Functions

- def __init__

  Creates a MCNP surface object.
- def __repr__
- def __str__

Public Attributes

- name

  Surface number.
- s_type

  The type of MCNP surface.
- r

  A radius in cm.
- d

  A location in cm.
- x_min

  The minimum x location in cm.
- x_max

  The maximum x location in cm.
- y_min

  The minimum y location in cm.
- y_max

  The maximum y location in cm.
- z_min

  The minimum z location in cm.
- z_max

  The maximum z location in cm.
- vx

  The x location of the center of the base in cm.
- vy

  The y location of the center of the base in cm.
- vz

  The z location of the center of the base in cm.
- hx

  The change in x for the height vector in cm.
- hy

  The change in y for the height vector in cm.
- hz

  The change in z for the height vector in cm.
- r1

  The radius of the lower cone base in cm.
- r2

  The radius of the upper cone base in cm.
- c

  describing the surface feature.

9.12.1 Constructor & Destructor Documentation

9.12.1.1 def MCNP_Utilities.MCNP_Surface.__init__ (*self*, *name*, *s_type*, *r* = `-1`, *d* = `-1`, *x_min* = `-1`, *x_max* = `-1`, *y_min* = `-1`, *y_max* = `-1`, *z_min* = `-1`, *z_max* = `-1`, *vx* = `-1`, *vy* = `-1`, *vz* = `-1`, *hx* = `-1`, *hy* = `-1`, *hz* = `-1`, *r1* = `-1`, *r2* = `-1`, *comment* = `" "`)

Creates a MCNP surface object.

Currently can handle SO, (PX,PY,PZ), (CX,CY,CZ), RCC, RPP, and TRC surfaces and macrobodies. All others will throw an exception. Attributes not used are specified as -1. All atribute names follow those shown in Table 3.1 and Section 3.III.D in Volume II of the MCNP manual

9.12.2 Member Function Documentation

9.12.2.1 def MCNP_Utilities.MCNP_Surface.__repr__ (*self*)

9.12.2.2 def MCNP_Utilities.MCNP_Surface.__str__ (*self*)

9.12.3 Member Data Documentation

9.12.3.1 MCNP_Utilities.MCNP_Surface.c

describing the surface feature.

Can be used to find the surface corresponding to a particular geometric feature

9.12.3.2 MCNP_Utilities.MCNP_Surface.d

A location in cm.

Used for the PX, PY, and PZ surfaces [Default: -1]

9.12.3.3 MCNP_Utilities.MCNP_Surface.hx

The change in x for the height vector in cm.

Used for the RCC and TRC macrobody [Default: -1]

9.12.3.4 MCNP_Utilities.MCNP_Surface.hy

The change in y for the height vector in cm.

Used for the RCC and TRC macrobody [Default: -1]

9.12.3.5 MCNP_Utilities.MCNP_Surface.hz

The change in z for the height vector in cm.

Used for the RCC and TRC macrobody [Default: -1]

9.12.3.6 MCNP_Utilities.MCNP_Surface.name

Surface number.

9.12.3.7 MCNP_Utilities.MCNP_Surface.r

A radius in cm.

Used for the SO, CX, CY, CZ, and RCC surfaces [Default: -1]

9.12.3.8 MCNP_Utilities.MCNP_Surface.r1

The radius of the lower cone base in cm.

Used for the TRC macrobody [Default: -1]

9.12.3.9 MCNP_Utilities.MCNP_Surface.r2

The radius of the upper cone base in cm.

Used for the TRC macrobody [Default: -1]

9.12.3.10 MCNP_Utilities.MCNP_Surface.s_type

The type of MCNP surface.

Currently can specify "SO", ("PX","PY","PZ"), ("CX","CY","CZ"), "RCC", "RPP", and ("KX","KY","KY")

9.12.3.11 MCNP_Utilities.MCNP_Surface.vx

The x location of the center of the base in cm.

Used for the RCC and TRC macrobody [Default: -1]

9.12.3.12 MCNP_Utilities.MCNP_Surface.vy

The y location of the center of the base in cm.

Used for the RCC and TRC macrobody [Default: -1]

9.12.3.13 MCNP_Utilities.MCNP_Surface.vz

The z location of the center of the base in cm.

Used for the RCC and TRC macrobody [Default: -1]

9.12.3.14 MCNP_Utilities.MCNP_Surface.x_max

The maximum x location in cm.

Used for the RPP macrobody [Default: -1]

9.12.3.15 MCNP_Utilities.MCNP_Surface.x_min

The minimum x location in cm.

Used for the RPP macrobody [Default: -1]

9.12.3.16 MCNP_Utilities.MCNP_Surface.y_max

The maximum y location in cm.

Used for the RPP macrobody [Default: -1]

9.12.3.17 MCNP_Utilities.MCNP_Surface.y_min

The minimum y location in cm.

Used for the RPP macrobody [Default: -1]

9.12.3.18 MCNP_Utilities.MCNP_Surface.z_max

The maximum z location in cm.

Used for the RPP macrobody [Default: -1]

9.12.3.19 MCNP_Utilities.MCNP_Surface.z_min

The minimum z location in cm.

Used for the RPP macrobody [Default: -1]

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/MCNP_Utilities.py

9.13 Utilities.Meta_Stats Class Reference

Stores and prints effectiveness stats for each metaheuristic search method.

Public Member Functions

- def __init__

  Initializer.
- def __repr__
- def __str__
- def update

  Adds val tuples to the algorithm arg's tuples.
- def write

  Create and open input file.

Public Attributes

- algorithms

  dictionary string name of each algorithms
- fname

  str Name and path of the file to store the timeline for post processing

9.13.1 Detailed Description

Stores and prints effectiveness stats for each metaheuristic search method.

9.13.2 Constructor & Destructor Documentation

9.13.2.1 def Utilities.Meta_Stats.__init__ (*self*, *mat_levy* = (0,0, *cell_levy* = (0,0, *elite_cross* = (0,0, *part_inv* = (0,0, *mutate* = (0,0, *two_opt* = (0,0, *crossover* = (0,0, *three_op* = (0,0, *discard* = (0,0, *fname* = os.path.abspath(os.path.join(os.getcwd(), os.pardir), *Results*, *meta_stats*, *txt*)

Initializer.

Parameters

mat_levy	tuple contains the changes and total number of function evaluations for the Mat_Levy_Flights function
cell_levy	tuple contains the changes and total number of function evaluations for the Cell_Levy_Flights function
elite_cross	tuple contains the changes and total number of function evaluations for the Mutate_Mats function
part_inv	tuple contains the changes and total number of function evaluations for the Partial_Inversion function
mutate	tuple contains the changes and total number of function evaluations for the Mutate function
two_opt	tuple contains the changes and total number of function evaluations for the 2-opt function
crossover	tuple contains the changes and total number of function evaluations for the Crossover function
three_op	tuple contains the changes and total number of function evaluations for the Three_opt function
discard	tuple contains the changes and total number of function evaluations for the Discard function

9.13.3 Member Function Documentation

9.13.3.1 def Utilities.Meta_Stats.__repr__ (*self*)

9.13.3.2 def Utilities.Meta_Stats.__str__ (*self*)

9.13.3.3 def Utilities.Meta_Stats.update (*self, alg, val*)

Adds val tuples to the algorithm arg's tuples.

Parameters

alg	str name of the algorithm selected
val	tuple value to be added

9.13.3.4 def Utilities.Meta_Stats.write (*self, header =* `False`)

Create and open input file.

9.13.4 Member Data Documentation

9.13.4.1 Utilities.Meta_Stats.algorithms

dictionary string name of each algorithms

9.13.4.2 Utilities.Meta_Stats.fname

str Name and path of the file to store the timeline for post processing

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py

9.14 NuclearData.Moderating_Ratio Class Reference

Creates a moderating ratio object.

Public Member Functions

- def __init__
- def __repr__
- def __str__

Public Attributes

- name

  str The material name
- mr_1MeV

  int The moderating ratio at 1 MeV
- mr_14MeV

  int The moderating ratio at 14 MeV

9.14.1 Detailed Description

Creates a moderating ratio object.

9.14.2 Constructor & Destructor Documentation

9.14.2.1 def NuclearData.Moderating_Ratio.__init__ (*self, name, mr_1MeV* = 0, *mr_14MeV* = 0)

9.14.3 Member Function Documentation

9.14.3.1 def NuclearData.Moderating_Ratio.__repr__ (*self*)

9.14.3.2 def NuclearData.Moderating_Ratio.__str__ (*self*)

9.14.4 Member Data Documentation

9.14.4.1 NuclearData.Moderating_Ratio.mr_14MeV

int The moderating ratio at 14 MeV

9.14.4.2 NuclearData.Moderating_Ratio.mr_1MeV

int The moderating ratio at 1 MeV

9.14.4.3 NuclearData.Moderating_Ratio.name

str The material name

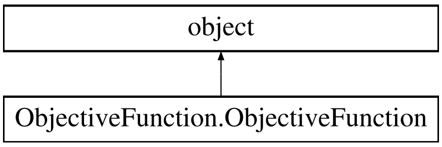
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/NuclearData.py

9.15 ObjectiveFunction.ObjectiveFunction Class Reference

The class creates a ObjectiveFunction object that can be used in optimization algorithms.

Inheritance diagram for ObjectiveFunction.ObjectiveFunction:



Public Member Functions

- def __init__

  Constructor to build the ObjectiveFunction class.
- def __repr__

  ObjectiveFunction class param print function.
- def __str__

  Human readable ObjectiveFunction print function.
- def set_obj_func

  Converts an input string name for a function to a function handle.
- def u_opt

  Calculated the fitness of a series of values using the U-Optimality condition.
- def least_squares

  Calculated the fitness of a series of values using the least squares condition.
- def relative_least_squares

  Calculated the fitness of a series of values using the relative least squares condition.

Public Attributes

- func
- funcTally
- objType
- objForm
- objective

9.15.1 Detailed Description

The class creates a ObjectiveFunction object that can be used in optimization algorithms.

9.15.2 Constructor & Destructor Documentation

9.15.2.1 def ObjectiveFunction.ObjectiveFunction.__init__ (*self*, *method* = `None`, *tallyNum* = `None`, *objType* = `None`, *objForm* = `None`, *objective* = `None`)

Constructor to build the ObjectiveFunction class.

Parameters

self	*object pointer* The object pointer.

_FUNCT_DICT	*dictionary* A mapping from string function names to function handles.
method	*string* The name of the objective function to evaluate.
tallyNum	*string* An associated MCNP tally number that is to be used to provide the input for the objective function calculation.
objType	*string* The type of objective. Valid entries are "spectrum" or • "total".
objForm	*integer* The type of objective. Valid entries are 0-4. 0 = "mcnp" 1 = "normalized" 2 = "differential" 3 = "normalized_differential" 4 = "lethargy" 5 = "normalized_lethargy"
objective	*integer, float, or numpy array* The desired objective associated with the optimization. The chosen value and type must be compatible with the optiization function chosen.

9.15.3 Member Function Documentation

9.15.3.1 def ObjectiveFunction.ObjectiveFunction.__repr__ (*self*)

ObjectiveFunction class param print function.

Parameters

self	*pointer* The ObjectiveFunction pointer.

9.15.3.2 def ObjectiveFunction.ObjectiveFunction.__str__ (*self*)

Human readable ObjectiveFunction print function.

Parameters

self	*pointer* The ObjectiveFunction pointer.

9.15.3.3 def ObjectiveFunction.ObjectiveFunction.least_squares (*self, c*)

Calculated the fitness of a series of values using the least squares condition.

Parameters

self	*pointer* The ObjectiveFunction pointer.
c	*numpy array* The array of results corresponding to a candidate design. For example, this can be an energy spectra of a flux.

Returns

float: The least squares criteria based fitness for a design.

9.15.3.4 def ObjectiveFunction.ObjectiveFunction.relative_least_squares (*self*, *c*, *project* = `True`)

Calculated the fitness of a series of values using the relative least squares condition.

This provides equal weighting to all bins in the data set being evaluated, regardless of overall magnitude.

Parameters

self	*pointer* The ObjectiveFunction pointer.
c	*numpy array* The array of results corresponding to a candidate design. For example, this can be an energy spectra of a flux.
project	*boolean* A flag on wether to project a reasonable guess to bins that have zero for values. The projected value is a simple linear exprapolation.

Returns

float: The relative_least_squares criteria based fitness for a design.

9.15.3.5 def ObjectiveFunction.ObjectiveFunction.set_obj_func (*self*, *funcName*)

Converts an input string name for a function to a function handle.

Parameters

self	*pointer* The ObjectiveFunction pointer.
funcName	*string* A string identifying the objective function to be used.

9.15.3.6 def ObjectiveFunction.ObjectiveFunction.u_opt (*self*, *c*)

Calculated the fitness of a series of values using the U-Optimality condition.

From: "Relationships among Several Optimality Criteria E" by E.A. Rady.

Parameters

self	*pointer* The ObjectiveFunction pointer.
c	*numpy array* The array of results corresponding to a candidate design. For example, this can be an energy spectra of a flux.

Returns

float: The u-optimality criteria based fitness for a design.

9.15.4 Member Data Documentation

9.15.4.1 ObjectiveFunction.ObjectiveFunction.func

9.15.4.2 ObjectiveFunction.ObjectiveFunction.funcTally

9.15.4.3 ObjectiveFunction.ObjectiveFunction.objective

9.15.4.4 ObjectiveFunction.ObjectiveFunction.objForm

9.15.4.5 ObjectiveFunction.ObjectiveFunction.objType

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/ObjectiveFunction.py

9.16 Gnowee_Utilities.Parent Class Reference

Public Member Functions

- def __init__

  Creates a parent object representing a current design.
- def __repr__
- def __str__

Public Attributes

- ident

  integer A set identifier tying a parent to a folder set
- geom

  object An object containing the design geometry variables
- rset

  [MCNP_Settings object] An object representing the settings for running the MCNP radiation trasport code.
- fit

  scalar The assessed design fitness
- fixed_mats

  integer Number of fixed materials in the geometry.

9.16.1 Constructor & Destructor Documentation

9.16.1.1 def Gnowee_Utilities.Parent.__init__ (*self, identifier, eta, geometry, GS, mcnp, mats, ex, i =* `0`, *fitness =* `1E15`, *build_geom =* `True`)

Creates a parent object representing a current design.

Parameters

identifier	integer A set identifier tying a parent to a folder set
eta	[ETA parameters object] An object that contains all of the constraints required to initialize the geometry
geometry	[MCNP_Geometry object] The geometry for running the MCNP radiation trasport code. Contains the surfaces, cells, and material information
GS	[Gnowee Settings object] An object representing the settings for the optimization algorithm
mcnp	[MCNP_Settings object] An object representing the settings for running the MCNP radiation trasport code. Contains the source, physics, and tally information.
mats	[dict of material objects] A dictionary of the material objects from which ETA materials can be selected [Default: {}]
ex	list A list of materials to be excluded
i	integer (optional) Parent indexed location for initial LHC sampling purposes fitness float (optional) The assessed design fitness
build_geom	boolean (optional) Used to indicate if the geometry needs to be build for a new parent. Turned off if the complete geometry is passed in.

9.16.2 Member Function Documentation

9.16.2.1 def Gnowee_Utilities.Parent.__repr__ (*self*)

9.16.2.2 def Gnowee_Utilities.Parent.__str__ (*self*)

9.16.3 Member Data Documentation

9.16.3.1 Gnowee_Utilities.Parent.fit

scalar The assessed design fitness

9.16.3.2 Gnowee_Utilities.Parent.fixed_mats

integer Number of fixed materials in the geometry.

This accounts for structural materials and foils that shouldn't change. Determine number of fixed mats

9.16.3.3 Gnowee_Utilities.Parent.geom

object An object containing the design geometry variables

9.16.3.4 Gnowee_Utilities.Parent.ident

integer A set identifier tying a parent to a folder set

9.16.3.5 Gnowee_Utilities.Parent.rset

[MCNP_Settings object] An object representing the settings for running the MCNP radiation trasport code.

Contains the source, physics, and tally information.

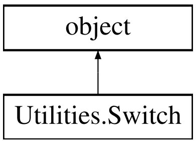
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Gnowee_Utilities.py

9.17 Utilities.Switch Class Reference

Creates a switch class object to switch between cases.

Inheritance diagram for Utilities.Switch:



Public Member Functions

- def __init__

  The constructor.
- def __iter__

  Return the match method once, then stop.
- def match

  PrintIndicate whether or not to enter a case suite.

Public Attributes

- value

  string case selector value
- fall

  boolean based on match

9.17.1 Detailed Description

Creates a switch class object to switch between cases.

9.17.2 Constructor & Destructor Documentation

9.17.2.1 def Utilities.Switch.__init__ (*self*, *value*)

The constructor.

Parameters

value	selector value

9.17.3 Member Function Documentation

9.17.3.1 def Utilities.Switch.__iter__ (*self*)

Return the match method once, then stop.

9.17.3.2 def Utilities.Switch.match (*self*, *args*)

PrintIndicate whether or not to enter a case suite.

Parameters

args	list of arguments to match with

9.17.4 Member Data Documentation

9.17.4.1 Utilities.Switch.fall

boolean based on match

9.17.4.2 Utilities.Switch.value

string case selector value

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py

9.18 Gnowee_Utilities.Timeline Class Reference

Public Member Functions

- def __init__

  An object that stores event objects to track optimization progress.
- def __repr__
- def __str__
- def update
- def write

Public Attributes

- tline

  list of event objects Contains a list of event objects detailing the optimization history
- fname

  str Name and path of the file to store the timeline for post processing

9.18.1 Constructor & Destructor Documentation

9.18.1.1 def Gnowee_Utilities.Timeline.__init__ (*self*, *tline* = `[]`, *fname* = `os.path.abspath(os.path.join(os.getcwd(), os.pardir),` *Results*, *timeline*, *txt*)

An object that stores event objects to track optimization progress.

9.18.2 Member Function Documentation

9.18.2.1 def Gnowee_Utilities.Timeline.__repr__ (*self*)

9.18.2.2 def Gnowee_Utilities.Timeline.__str__ (*self*)

9.18.2.3 def Gnowee_Utilities.Timeline.update (*self*, *pop*, *gen*, *feval*)

9.18.2.4 def Gnowee_Utilities.Timeline.write (*self*)

9.18.3 Member Data Documentation

9.18.3.1 Gnowee_Utilities.Timeline.fname

str Name and path of the file to store the timeline for post processing

9.18.3.2 Gnowee_Utilities.Timeline.tline

list of event objects Contains a list of event objects detailing the optimization history

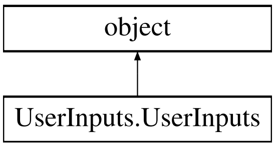
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Gnowee_Utilities.py

9.19 UserInputs.UserInputs Class Reference

The class creates a UserInputs object to store the user input file locations, read the user inputs, and set the appropriate classes required to run Coeus.

Inheritance diagram for UserInputs.UserInputs:



Public Member Functions

- def __init__

    Constructor to build the UserInputs class.
- def __repr__

    UserInputs print function.
- def __str__

    Human readable UserInputs print function.
- def read_coeus_settings

    Reads the input file and creates the corresponding objects and populates their attributes.

Public Attributes

- coeusInput

    A path for the Coeus input file.
- mcnpInput

9.19.1 Detailed Description

The class creates a UserInputs object to store the user input file locations, read the user inputs, and set the appropriate classes required to run Coeus.

9.19.2 Constructor & Destructor Documentation

9.19.2.1 def UserInputs.UserInputs.__init__ (*self*, *coeusInputPath* = `None`, *mcnpInputPath* = `None`)

Constructor to build the UserInputs class.

If paths is specified, the object attributes are populated.

Parameters

self	*object pointer* The objeUserInputsct pointer.
coeusInputPath	*string* The path to the coues input file.
mcnpInputPath	*string* The path to the mcnp input file.

9.19.3 Member Function Documentation

9.19.3.1 def UserInputs.UserInputs.__repr__ (*self*)

UserInputs print function.

Parameters

self	*object pointer* The UserInputs pointer.

9.19.3.2 def UserInputs.UserInputs.__str__ (*self*)

Human readable UserInputs print function.

Parameters

self	*object pointer* The UserInputs pointer.

9.19.3.3 def UserInputs.UserInputs.read_coeus_settings (*self*)

Reads the input file and creates the corresponding objects and populates their attributes.

Parameters

self	*object pointer* The UserInputs pointer.

Returns

Objective Function Object : An ObjectiveFunction object initialized with the user input parameters.

9.19.4 Member Data Documentation

9.19.4.1 def UserInputs.UserInputs.coeusInput

A path for the Coeus input file.

9.19.4.2 UserInputs.UserInputs.mcnpInput

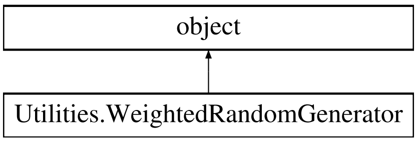
The documentation for this class was generated from the following file:

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/UserInputs.py

9.20 Utilities.WeightedRandomGenerator Class Reference

Defines a class of weights to be used to select number of instances in array randomly with linear weighting.

Inheritance diagram for Utilities.WeightedRandomGenerator:



Public Member Functions

- def __init__
- def next
- def __call__

Public Attributes

- totals

9.20.1 Detailed Description

Defines a class of weights to be used to select number of instances in array randomly with linear weighting.

9.20.2 Constructor & Destructor Documentation

9.20.2.1 def Utilities.WeightedRandomGenerator.__init__ (*self*, *weights*)

9.20.3 Member Function Documentation

9.20.3.1 def Utilities.WeightedRandomGenerator.__call__ (*self*)

9.20.3.2 def Utilities.WeightedRandomGenerator.next (*self*)

9.20.4 Member Data Documentation

9.20.4.1 Utilities.WeightedRandomGenerator.totals

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

- /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py

Chapter 10

File Documentation

10.1 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/_init_.py File Reference

Namespaces

- _init_

10.2 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Plotting/_init_.py File Reference

Namespaces

- _init_

10.3 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/_init_.py File Reference

Namespaces

- _init_

10.4 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/ADVANTG_Utilities.py File Reference

Classes

- class ADVANTG_Utilities.ADVANTG_Settings

Namespaces

- ADVANTG_Utilities

Functions

- def ADVANTG_Utilities.Print_ADVANTG_Input

  Print the generated MCNP input deck to file.

Variables

- tuple ADVANTG_Utilities.module_logger = logging.getLogger('Coesh.ADVANTG_Utilities')

10.5 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Coeus.py File Reference

Namespaces

- Coeus

Functions

- def Coeus.print_MCNP_input_files

  Print MCNP input Files for each algorithm.
- def Coeus.run_MCNP_on_algo

  Run MCNP for each algorithm.
- def Coeus.main

  Entry point for the Coeus program.

10.6 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Coeus_local.py File Reference

Namespaces

- Coeus_local

Functions

- def Coeus_local.main
- def Coeus_local.print_MCNP_input_files

  Local Function definitions.
- def Coeus_local.run_MCNP_on_algo

Variables

- tuple Coeus_local.start_time = time.time()
- tuple Coeus_local.rundir = os.path.abspath(os.path.join(os.path.abspath(os.getcwd()),os.pardir))
- tuple Coeus_local.logger = logging.getLogger('Coeus')
- tuple Coeus_local.hdlr = logging.FileHandler('{}/Results/logfile.log'.format(os.path.abspath(os.path.join(os.path.abspath(os.getcwd()), os.pardir))))
- tuple Coeus_local.formatter = logging.Formatter('%(asctime)s %(levelname)s %(message)s')
- tuple Coeus_local.restart = kwargs.get('restart')
- tuple Coeus_local.obj_path = kwargs.get('obj_path')

- tuple Coeus_local.eta_path = kwargs.get('eta_constraints_path')
- tuple Coeus_local.gs_path = kwargs.get('gs_settings_path')
- tuple Coeus_local.advantg_path = kwargs.get('advantg_settings_path')
- tuple Coeus_local.mcnp_path = kwargs.get('mcnp_settings_path')
- tuple Coeus_local.materials_library_path = kwargs.get('materials_library_path')
- tuple Coeus_local.source_path = kwargs.get('source_path')
- tuple Coeus_local.eta_params = ETA_Parameters()
- tuple Coeus_local.g_set = Gnowee_Settings()
- tuple Coeus_local.advantg_set = ADVANTG_Settings()
- tuple Coeus_local.mcnp_set = MCNP_Settings(eta_params)
- tuple Coeus_local.mat_lib = Build_Matlib(materials_library_path)
- list Coeus_local.pop = [ ]
- tuple Coeus_local.base_eta = MCNP_Geometry()
- tuple Coeus_local.eta = MCNP_Geometry()
- tuple Coeus_local.nps = eta.read_geom(rundir+str(i)+"/ETA.inp", mat_lib)
- list Coeus_local.ids = [ ]

  33 if history.tline[-1].g>10: converge=True 33
- list Coeus_local.particles = [ ]
- tuple Coeus_local.stats = Meta_Stats()
- tuple Coeus_local.history = Timeline()
- tuple Coeus_local.mod_rat = Calc_Moderating_Ratio(mat_lib)
- tuple Coeus_local.new_pop = Partial_Inversion(pop,mod_rat,mat_lib,g_set)

  Partial Inversion ########.
- Coeus_local.converge = False

  Test Convergence ######## Test generational convergence.

10.7 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Constraints.py File Reference

Classes

- class Constraints.Constraints

  The class creates a Constraints object that can be used in optimization algorithms.

Namespaces

- Constraints
- Coeus

Variables

- tuple Constraints.module_logger = logging.getLogger('Coeus.Constraints')

10.8 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/ETA_Utilities.py File Reference

Classes

- class ETA_Utilities.ETA_Parameters

Namespaces

- ETA_Utilities

Variables

- tuple ETA_Utilities.module_logger = logging.getLogger('Coeus.ETA_Utilities')

10.9 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Gnowee_-Utilities.py File Reference

Classes

- class Gnowee_Utilities.Gnowee_Settings
- class Gnowee_Utilities.Parent
- class Gnowee_Utilities.Timeline

Namespaces

- Gnowee_Utilities

Functions

- def Gnowee_Utilities.Calc_Fitness

  Print the generated MCNP input deck to file.
- def Gnowee_Utilities.Pop_Update

  Updates the population based on the assessed fitness values.
- def Gnowee_Utilities.Rejection_Bounds

  Application of problem boundaries to generated solutions.
- def Gnowee_Utilities.Simple_Bounds

  Application of problem boundaries to generated solutions.

Variables

- tuple Gnowee_Utilities.module_logger = logging.getLogger('Coeus.Gnowee_Utilities')

10.10 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/MCNP_Utilities.py File Reference

Classes

- class MCNP_Utilities.MCNP_Settings
- class MCNP_Utilities.MCNP_Geometry
- class MCNP_Utilities.MCNP_Surface
- class MCNP_Utilities.MCNP_Cell

Namespaces

- MCNP_Utilities

Functions

- def MCNP_Utilities.Print_MCNP_Input

  Print the generated MCNP input deck to file.
- def MCNP_Utilities.Read_Tally_Output

  Read the generated MCNP output and return the tally results.
- def MCNP_Utilities.Read_MCNP_Output

  Read the generated MCNP output and return the tally results.

Variables

- tuple MCNP_Utilities.module_logger = logging.getLogger('Coeus.MCNP_Utilities')

10.11 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Metaheuristics.py File Reference

Namespaces

- Metaheuristics

Functions

- def Metaheuristics.Mat_Levy_Flights

  Change cell materials based on Levy draw.
- def Metaheuristics.Cell_Levy_Flights

  Cell Levy Flight: Change all cell and foil starting locations and cell deltas based on Levy draw.
- def Metaheuristics.Elite_Crossover

  Change the materials between the top parent and an elite parent based on moderating ratio.
- def Metaheuristics.Partial_Inversion

  Invert materials based on moderating ratio gradient.
- def Metaheuristics.Two_opt

  Implement 2_opt by reordering layers for top parents.
- def Metaheuristics.Crossover

  For each parent in top S.fe parents, N1, randomly select a parent, N2.
- def Metaheuristics.Three_opt

  Perform for horizontal macrobodies if the number of cells is greater than 6.
- def Metaheuristics.Discard_Cells

  Discard a cell from fd parents.
- def Metaheuristics.Mutate

  Mutate parent population and build new ones.

Variables

- tuple Metaheuristics.module_logger = logging.getLogger('Coeus.Metaheuristics')

10.12 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/NuclearData.py File Reference

Classes

- class NuclearData.Moderating_Ratio

  Creates a moderating ratio object.

Namespaces

- NuclearData

Functions

- def NuclearData.Build_Matlib

  Builds and initializes a library of elements and materials provided by user using PyNE material library functions.
- def NuclearData.Set_Density

  Initialized the material density for the elemental library.
- def NuclearData.Strip_Undesireables

  Removes materials from library that don't work from an engineering, safety, or cost perspective.
- def NuclearData.Calc_Moderating_Ratio

  Calculated and returns the moderating ratio for each material in a materials library.

Variables

- tuple NuclearData.module_logger = logging.getLogger('Coeus.NuclearData')

10.13 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/ObjectiveFunction.py File Reference

Classes

- class ObjectiveFunction.ObjectiveFunction

  The class creates a ObjectiveFunction object that can be used in optimization algorithms.

Namespaces

- ObjectiveFunction
- Coeus

Variables

- tuple ObjectiveFunction.module_logger = logging.getLogger('Coeus.ObjectiveFunction')

10.14 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Plotting/OptiPlot.py File Reference

Namespaces

- OptiPlot

Functions

- def OptiPlot.Plot_Vars
- def OptiPlot.Plot_Hist
- def OptiPlot.Plot_Feval_Hist
- def OptiPlot.Plot_TLF
- def OptiPlot.Plot_Meta_Optimization

10.15 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/__init__.py File Reference

Namespaces

- pyDOE

Variables

- string pyDOE.__author__ = ’Abraham Lee’
- string pyDOE.__version__ = ’0.3.8’

10.16 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/__init__.py File Reference

Namespaces

- pyDOE

10.17 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/build_regression_matrix.py File Reference

Namespaces

- pyDOE.build_regression_matrix

Functions

- def pyDOE.build_regression_matrix.grep
- def pyDOE.build_regression_matrix.build_regression_matrix

10.18 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/build_regression_matrix.py File Reference

Namespaces

- pyDOE.build_regression_matrix

Functions

- def pyDOE.build_regression_matrix.grep
- def pyDOE.build_regression_matrix.build_regression_matrix

10.19 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_box_behnken.py File Reference

Namespaces

- pyDOE.doe_box_behnken

Functions

- def pyDOE.doe_box_behnken.bbdesign

Variables

- list pyDOE.doe_box_behnken.__all__ = ['bbdesign']

10.20 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_box_behnken.py File Reference

Namespaces

- pyDOE.doe_box_behnken

Functions

- def pyDOE.doe_box_behnken.bbdesign

10.21 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_composite.py File Reference

Namespaces

- pyDOE.doe_composite

Functions

- def pyDOE.doe_composite.ccdesign

Variables

- list pyDOE.doe_composite.__all__ = ['ccdesign']

10.22 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_composite.py File Reference

Namespaces

- pyDOE.doe_composite

Functions

- def pyDOE.doe_composite.ccdesign

10.23 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_factorial.py File Reference

Namespaces

- pyDOE.doe_factorial

Functions

- def pyDOE.doe_factorial.fullfact
- def pyDOE.doe_factorial.ff2n
- def pyDOE.doe_factorial.fracfact

Variables

- list pyDOE.doe_factorial.__all__ = ['np', 'fullfact', 'ff2n', 'fracfact']

10.24 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_factorial.py File Reference

Namespaces

- pyDOE.doe_factorial

Functions

- def pyDOE.doe_factorial.fullfact
- def pyDOE.doe_factorial.ff2n
- def pyDOE.doe_factorial.fracfact

10.25 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_fold.py File Reference

Namespaces

- pyDOE.doe_fold

Functions

- def pyDOE.doe_fold.fold

Variables

- list pyDOE.doe_fold.__all__ = ['fold']

10.26 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_fold.py File Reference

Namespaces

- pyDOE.doe_fold

Functions

- def pyDOE.doe_fold.fold

10.27 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_lhs.py File Reference

Namespaces

- pyDOE.doe_lhs

Functions

- def pyDOE.doe_lhs.lhs

Variables

- list pyDOE.doe_lhs.__all__ = ['lhs']

10.28 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_lhs.py File Reference

Namespaces

- pyDOE.doe_lhs

Functions

- def pyDOE.doe_lhs.lhs

10.29 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_plackett_burman.py File Reference

Namespaces

- pyDOE.doe_plackett_burman

Functions

- def pyDOE.doe_plackett_burman.pbdesign

Variables

- list pyDOE.doe_plackett_burman.__all__ = ['pbdesign']

10.30 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_plackett_burman.py File Reference

Namespaces

- pyDOE.doe_plackett_burman

Functions

- def pyDOE.doe_plackett_burman.pbdesign

10.31 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_repeat_center.py File Reference

Namespaces

- pyDOE.doe_repeat_center

Functions

- def pyDOE.doe_repeat_center.repeat_center

10.32 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_repeat_center.py File Reference

Namespaces

- pyDOE.doe_repeat_center

Functions

- def pyDOE.doe_repeat_center.repeat_center

10.33 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_star.py File Reference

Namespaces

- pyDOE.doe_star

Functions

- def pyDOE.doe_star.star

10.34 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_star.py File Reference

Namespaces

- pyDOE.doe_star

Functions

- def pyDOE.doe_star.star

10.35 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/doe_union.py File Reference

Namespaces

- pyDOE.doe_union

Functions

- def pyDOE.doe_union.union

10.36 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/doe_union.py File Reference

Namespaces

- pyDOE.doe_union

Functions

- def pyDOE.doe_union.union

10.37 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/build/lib/pyDOE/var_regression_matrix.py File Reference

Namespaces

- pyDOE.var_regression_matrix

Functions

- def pyDOE.var_regression_matrix.var_regression_matrix

10.38 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/pyDOE/var_regression_matrix.py File Reference

Namespaces

- pyDOE.var_regression_matrix

Functions

- def pyDOE.var_regression_matrix.var_regression_matrix

10.39 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/pyDOE-0.3.8/setup.py File Reference

Namespaces

- setup

Functions

- def setup.read

10.40 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/-SamplingMethods.py File Reference

Namespaces

- SamplingMethods

Functions

- def SamplingMethods.Initial_Samples
- def SamplingMethods.integrand
- def SamplingMethods.Levy_Function
- def SamplingMethods.Levy
- def SamplingMethods.TLF

- def SamplingMethods.NOLH
- def SamplingMethods.params
- def SamplingMethods.Get_CDR_Permutations

Variables

- tuple SamplingMethods.module_logger = logging.getLogger('METACODE.SamplingMethods')
- tuple SamplingMethods.parser
- string SamplingMethods.help = "The configuration vector given as a list N1 N2 ... Nm"
- tuple SamplingMethods.args = parser.parse_args()

10.41 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Sampling/SamplingMethods_Full.py File Reference

Namespaces

- SamplingMethods_Full

Functions

- def SamplingMethods_Full.Initial_Samples
- def SamplingMethods_Full.Plot_Samples
- def SamplingMethods_Full.integrand
- def SamplingMethods_Full.Levy_Function
- def SamplingMethods_Full.Levy
- def SamplingMethods_Full.TLF
- def SamplingMethods_Full.Plot_Levy
- def SamplingMethods_Full.NOLH
- def SamplingMethods_Full.params
- def SamplingMethods_Full.Get_CDR_Permutations

Variables

- tuple SamplingMethods_Full.parser
- string SamplingMethods_Full.help = "The configuration vector given as a list N1 N2 ... Nm"
- tuple SamplingMethods_Full.args = parser.parse_args()

10.42 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Tests/test_ADVANTG_Utilities.py File Reference

Namespaces

- test_ADVANTG_Utilities

Functions

- def test_ADVANTG_Utilities.test_ADVANTG_setings
- def test_ADVANTG_Utilities.test_ADVANTG_settings_repr
- def test_ADVANTG_Utilities.test_ADVANTG_settings_str
- def test_ADVANTG_Utilities.test_ADVANTG_settings_read_settings
- def test_ADVANTG_Utilities.test_Print_ADVANTG_Input

Variables

- tuple test_ADVANTG_Utilities.constraint_path = os.getcwd()
- tuple test_ADVANTG_Utilities.mat_path = os.getcwd()
- tuple test_ADVANTG_Utilities.set_path = os.getcwd()
- string test_ADVANTG_Utilities.test_settings_repr = 'ADVANTG Settings(dplus, cadis, mcnp silo, 24, True, 0.01, 1, 0.5, 0.5, 0.5, 0.25, 0.25, 0.05, 1.0)'
- string test_ADVANTG_Utilities.test_settings_str

10.43 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Tests/test_ETA_Utilities.py File Reference

Namespaces

- test_ETA_Utilities

Functions

- def test_ETA_Utilities.test_eta_parameters
- def test_ETA_Utilities.test_eta_parameters_repr
- def test_ETA_Utilities.test_eta__parameters_str
- def test_ETA_Utilities.test_eta_parameters_read_obj
- def test_ETA_Utilities.test_eta_parameters_read_constraints

Variables

- tuple test_ETA_Utilities.eta_path = os.getcwd()
- tuple test_ETA_Utilities.obj_path = os.getcwd()
- tuple test_ETA_Utilities.spectrum
- string test_ETA_Utilities.test_parameters_repr = "ETA_Params(normalized differential, [], 1.59896054911e-06, 125.0, 5e+15, 15.24, 0.3, 0.5, 52.14, 1.0, 2.4, 5.48, 9.39, 70.22, Al, Al, Air (dry near sea level), Pb, 0.014, 2.69, Al, [0.1, 0.1, 0.1, 0.1, 0.01], 2.5, ['Zr', 'Zn', 'In', 'Al', 'Ta'], In, Al, [0.0254, 0.0127], 1.252, ['Au', 'Pb'], Al, Fe, 2.0, 3, 7)"
- string test_ETA_Utilities.test_parameters_str

10.44 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Tests/test_Gnowee_Utilities.py File Reference

Namespaces

- test_Gnowee_Utilities

Functions

- def test_Gnowee_Utilities.test_gnowee_settings
- def test_Gnowee_Utilities.test_gnowee_settings_read_settings
- def test_Gnowee_Utilities.test_parent
- def test_Gnowee_Utilities.test_timeline
- def test_Gnowee_Utilities.test_pop_update
- def test_Gnowee_Utilities.test_Rejection_Bounds

Variables

- string test_Gnowee_Utilities.test_repr = 'Gnowee Settings(25, lhc, 0.25, 0.2, 0.4, 10000, 100000, 1e-06, 200, 0.01, 0.01, 1.5, 1.0, 1, 10.0)'
- string test_Gnowee_Utilities.test_str = '\nGnowee Optimization Settings:\nPopulation Size = 25\nInitial sampling method = lhc\nDiscovery Fraction = 0.2\nElite fraction = 0.2\nLevy fraction = 0.4\nMaximum number of genrations = 10000\nMaximum number of function evaluations = 100000\nStall convergence tolerance = 1e-06\nStall iteration limit = 200\nOptimal fitness = 0.01\nOptimal convergence tolerance = 0.01\nLevy exponent = 1.5\nLevy scale unit = 1.0\nLevy independent variables = 1\nStep size scaling factor = 10.0\n'
- tuple test_Gnowee_Utilities.set_path = os.getcwd()
- tuple test_Gnowee_Utilities.mcnp_path = os.getcwd()
- tuple test_Gnowee_Utilities.source_path = os.getcwd()
- tuple test_Gnowee_Utilities.eta_path = os.getcwd()
- tuple test_Gnowee_Utilities.mat_path = os.getcwd()
- tuple test_Gnowee_Utilities.obj_path = os.getcwd()

10.45 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Tests/test_MCNP_Utilities.py File Reference

Namespaces

- test_MCNP_Utilities

Functions

- def test_MCNP_Utilities.test_mcnp_surface1
- def test_MCNP_Utilities.test_mcnp_surface2
- def test_MCNP_Utilities.test_mcnp_surface3
- def test_MCNP_Utilities.test_mcnp_surface4
- def test_MCNP_Utilities.test_mcnp_surface5
- def test_MCNP_Utilities.test_mcnp_surface6
- def test_MCNP_Utilities.test_mcnp_surface7
- def test_MCNP_Utilities.test_mcnp_surface8
- def test_MCNP_Utilities.test_mcnp_surface9
- def test_MCNP_Utilities.test_mcnp_surface10
- def test_MCNP_Utilities.test_mcnp_surface_repr
- def test_MCNP_Utilities.test_mcnp_surface_str
- def test_MCNP_Utilities.test_mcnp_cell1
- def test_MCNP_Utilities.test_mcnp_cell_repr
- def test_MCNP_Utilities.test_mcnp_cell_str1
- def test_MCNP_Utilities.test_mcnp_cell_str2
- def test_MCNP_Utilities.test_mcnp_cell_str3
- def test_MCNP_Utilities.test_mcnp_cell_str4
- def test_MCNP_Utilities.test_mcnp_addMat
- def test_MCNP_Utilities.test_mcnp_addSurf
- def test_MCNP_Utilities.test_mcnp_addCell
- def test_MCNP_Utilities.test_mcnp_geometry
- def test_MCNP_Utilities.test_init_geometry
- def test_MCNP_Utilities.test_mcnp_settings_read_source
- def test_MCNP_Utilities.test_Print_MCNP_Input
- def test_MCNP_Utilities.test_mcnp_settings_read_settings
- def test_MCNP_Utilities.test_Read_Tally_Output
- def test_MCNP_Utilities.test_Read_MCNP_Output
- def test_MCNP_Utilities.test_Read_MCNP_Output2

Variables

- list test_MCNP_Utilities.source
- tuple test_MCNP_Utilities.set_path = os.getcwd()
- tuple test_MCNP_Utilities.src_path = os.getcwd()
- tuple test_MCNP_Utilities.constraint_path = os.getcwd()
- tuple test_MCNP_Utilities.materials_library_path = os.getcwd()
- string test_MCNP_Utilities.test_surf_repr = 'MCNP Surface(600, TRC, vx=1.0, vy=2.0, vz=3.0, hx=2.5, hy=23.6, hz=23.56, r1=3.4, r2=1.0, c=test)'
- string test_MCNP_Utilities.test_surf_str = '700 px 2.00000 $test\n'
- string test_MCNP_Utilities.test_cell_repr = 'MCNP Cell:(1, mat=10, units=atom, density=0.0422, booleam geom=500 -501, n imp=1, p_imp=0, comment=)'
- string test_MCNP_Utilities.test_cell_str1 = '1 10 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n'
- string test_MCNP_Utilities.test_cell_str2 = '1 10 -4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n'
- string test_MCNP_Utilities.test_cell_str3 = '1 10 500 -501 imp:n=1 imp:p=0 $\n'
- string test_MCNP_Utilities.test_cell_str4 = '1 10 -4.22300e-02 (500 -501):(502 -503):(504 -505):(506 -507)-:(508\n -509):(509 -510) imp:n=1 imp:p=0 $\n'
- string test_MCNP_Utilities.mat_card = "C name: Air (dry near sea level)\nC density = 0.0\nm?\n 6012 -1.2256e-04\n 6013 -1.4365e-06\n 7014 -7.5527e-01\n 8016 -2.3178e-01\n 18036 -3.8527e-05\n 18038 -7.6673e-06\n 18040 -1.2781e-02\n"
- string test_MCNP_Utilities.test_geom_str1 = 'MCNP geometry instance properties:\nMCNP Surfaces:\n509 TRC 1.00000 2.00000 3.00000 2.50000 23.60000 23.56000 \n 3.40000 1.00000 $one\n\n504 px 2.00000 $two\n\n505 Py -2.00000 $three\n\nMCNP Cells:\n1 11 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\n2 12 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\n3 13 4.22000e-02 500 -501 imp:n=1 imp:p=0 $\n\nMCNP Materials:\nAir (dry near sea level)\nAl\n'
- string test_MCNP_Utilities.base_geom = 'MCNP geometry instance properties:\nMCNP Surfaces:\n500 TRC 0.00000 0.00000 16.12650 0.00000 0.00000 14.35147 \n 0.00001 5.16119 $inner debris cover\n\n501 TRC 0.00000 0.00000 15.24000 0.00000 0.00000 15.23797 \n 0.00001 5.48000 $outer debris cover\n\n502 TRC 0.00000 0.00000 30.77797 0.00000 0.00000 10.57235 \n 5.26908 9.07119 $inner cone\n\n503 TRC 0.00000 0.00000 30.47797 0.00000 0.00000 10.87235 \n 5.48000 9.39000 $outer cone\n\n504 RCC 0.00000 0.00000 41.35032 0.00000 0.00000 9.78968 \n 8.89000 $inner cylinder\n\n505 RCC 0.00000 0.00000 41.35032 0.00000 0.00000 9.78968 \n 9.39000 $outer cylinder\n\n506 RCC 0.00000 0.00000 51.14000 0.00000 0.00000 1.00000 \n 9.39000 $cover\n\n507 RCC 0.00000 0.00000 52.14000 0.00000 0.00000 2.40000 \n 5.63400 $adapter\n\nMCNP Cells:\n1 1 -2.70000e+00 500 -501 imp:n=1 imp:p=0 $\n\n2 1 -2.70000e+00 502 -503 imp:n=1 imp:p=0 $\n\n3 1 -2.70000e+00 504 -505 imp:n=1 imp:p=0 $\n\n4 1 -2.70000e+00 -506 imp:n=1 imp:p=0 $\n\n5 1 -2.70000e+00 -507 imp:n=1 imp:p=0 $\n\nMCNP Materials:\nAl\nZr\nZn\nIn\nTa\nAu\nPb\nFe\n'

10.46 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Tests/test_Metaheuristics.py File Reference

Namespaces

- test_Metaheuristics

Functions

- def test_Metaheuristics.test_Mat_Levy_Flights

Variables

- tuple test_Metaheuristics.set_path = os.getcwd()
- tuple test_Metaheuristics.mat_path = os.getcwd()

10.47 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Tests/test_NuclearData.py File Reference

Namespaces

- test_NuclearData

Functions

- def test_NuclearData.test_set_density1
- def test_NuclearData.test_set_density
- def test_NuclearData.test_strip_undesirables1
- def test_NuclearData.test_strip_undesirables2
- def test_NuclearData.test_build_matlib
- def test_NuclearData.test_Moderating_Ratio
- def test_NuclearData.test_Calc_Moderating_Ratio

Variables

- tuple test_NuclearData.mat_path = os.getcwd()

10.48 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Tests/test_Utilities.py File Reference

Namespaces

- test_Utilities

Functions

- def test_Utilities.test_cmd_thread
- def test_Utilities.test_run_transport_pp
- def test_Utilities.test_run_transport_threads
- def test_Utilities.test_to_normdiff
- def test_Utilities.test_Uopt
- def test_Utilities.test_LeastSquares
- def test_Utilities.test_RelativeLeastSquares
- def test_Utilities.test_functhreadwithreturn
- def test_Utilities.test_functhread
- def test_Utilities.test_Event
- def test_Utilities.test_Meta_Stats

Variables

- string test_Utilities.test_mcnp = 'mcnp6 ../NSA_Proposal_ETA.inp NSA_Proposal_ETA.out'

10.49 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/UserInputs.py File Reference

Classes

- class UserInputs.UserInputs

  The class creates a UserInputs object to store the user input file locations, read the user inputs, and set the appropriate classes required to run Coeus.

Namespaces

- UserInputs
- Coeus

Variables

- tuple UserInputs.module_logger = logging.getLogger('Coeus.UserInputs')

10.50 /home/pyne-user/Dropbox/UCB/Research/ETAs/Design/Coeus/Code/Utilities.py File Reference

Classes

- class Utilities.Switch

  Creates a switch class object to switch between cases.
- class Utilities.Cmd_Thread

  Creates a Thread class object to run command line programs in parallel.
- class Utilities.FuncThread

  Creates a Thread class object to run functions without returns in parallel.
- class Utilities.FuncThreadWithReturn

  Creates a Thread class object to run functions containing returns in parallel.
- class Utilities.Event

  an event object representing a snapshot in the optimization process
- class Utilities.WeightedRandomGenerator

  Defines a class of weights to be used to select number of instances in array randomly with linear weighting.
- class Utilities.Meta_Stats

  Stores and prints effectiveness stats for each metaheuristic search method.

Namespaces

- Utilities

Functions

- def Utilities.Run_Transport_Threads

  Runs a multi-threaded transport calculation.
- def Utilities.Run_CmdLine

  A callable function to execute a command line program.
- def Utilities.Run_Transport_PP

Runs a multi-threaded transport calculation.

- def Utilities.Run_Transport

  Build a Slurm Batch script using the Jobs Array feature to run transport calculations.
- def Utilities.Build_Batch

  Build a Slurm Batch script using the Jobs Array feature to run transport calculations.
- def Utilities.to_Norm

  Normalizes a MCNP tallied flux.
- def Utilities.to_NormDiff

  Converts a MCNP tallied flux to a Normalized Differential flux.

Variables

- tuple Utilities.module_logger = logging.getLogger('Coeus.Utilities')

10.51 mainpage.py File Reference

Namespaces

- mainpage

Index