F_UNCLE Documentation

Release 0.0

A. Fraser and S. Andrews

May 18, 2016

CONTENTS

Contents:

CHAPTER

ONE

DOCUMENTATION

The FUNCLE module

Functional UNcertainty Constrained by Law and Experiment

CHAPTER

TWO

UTILITIES

These are abstract classes which are used in the analysis

2.1 pyStruc

pyStruc.py

Contains the Struc abstract class definition

2.1.1 Authors

- Stephen Andrews (SA)

2.1.2 Revisions

0 -> Initial class creation (10-09-2015)

2.1.3 To Do

- Nothing

class `FUNCLE.utils.pyStruc.`**`Struc`**(*name*, *def_opts=None*, *informs=None*, *warns=None*, **args*, ***kwargs*)

Abstract object to contain properties and warnings

`name`
str – The name of the object

`def_opts`
dict – Default options and bounds

`informs`
dict – Important user information prompts

`warns`
dict – Optional warnings

`options`
dict – The options as set by the user

`get_inform`(*err_id*)
Returns an inform corresponding to the error code

Parameters **err_id** (*int*) – Error ID number

Returns String containing the error message

Return type (str)

get_option(*name*)

Returns the option corresponding the the given name

Parameters **name** (*str*) – Name of the option

Returns Value of the option corresponding to ‘name’

get_warn(*warn_id*)

Returns an inform corresponding to the warning code

Parameters **warn_id** (*int*) – Warning ID number

Returns String containing the warning message

Return type (str)

plot()

Returns a plot of the object

set_option(*name*, *value*)

Sets the option corresponding to the given name to a specified value.

Enforces the following checks

1. name is valid
2. value is of correct type
3. value is within bounds
4. **Not Implemented** value has correct units

Parameters

- **name** (*str*) – Name of the option to set
- **value** – Value of the option to set

Returns None

write_to_file()

Writes the object to a file

class FUNCLE.utils.pyStruc.**TestObject**(*methodName='runTest'*)

Test of the Struc object

test_bad_get_inform()

get_inform should raise an error if the index is not an int or is out of bounds

test_bad_get_option()

get_option should raise a KeyError if an invalid name is given

test_bad_get_warn()

get_warn should raise an error if the index is not an int or is out of bounds

test_bad_set_option()

Structure should raise a KeyError when given an unknown option

test_get_inform()

get_inform should return a given string for error code 0

test_get_option()
get_option should return the default value if in a vanilla instantiation

test_get_warn()
get_warn should return a given string for error code 0

test_inst_at_bounds()
Structure should accept a value *at* the upper and lower bound for ints and floats

test_inst_bad_option()
Structure should ignore the unknown option "potatoes"

test_inst_bad_type()
Structure should raise a type error when int apples is set to a float

test_inst_over_bound()
Structure should raise a value error if apples is set above its bounds

test_inst_under_bound()
Structure should raise value error if apples is below its bounds

test_list_set_option()
Structure should raise a value error if list item set above bound

test_no_bounds()
Tests the bounds of options which are unbounded. Should be able to be set to very large or small values

test_standard_instantiation()
Test normal usage

2.2 pyContainer

container.pt

Contains the container abstract class definition

2.2.1 Authors

- Stephen Andrews (SA)

2.2.2 Revisions

0 -> Initial class creation (10-09-2015)

2.2.3 To Do

- Nothing

class FUNCLE.utils.pyContainer.**Container**(*name*, *def_opts=None*, *informs=None*, *warns=None*, **args*, ***kwargs*)
An abstract iterable container object

_contents
dict – An integer keyed list. Do not access this list directly use the iterable functions

---

Note: The container does not fill in the gaps when an object is delete, i. e. if the container contained indices 1,2 and 3 and index 2 was deleted, the object would then contain index 1 and 3. If an object were then ap- pended to the list it would have index 4.

`append`(*value*)
Appends the data to the end of contents

`clear`()
Deletes all the container contents

class `FUNCLE.utils.pyContainer.`**`TestContainer`**(*methodName='runTest'*)
Test of the container class

`test_append_to_holy_container`()
Tests appending to a container where an index has been deleted. Should append after the last index

`test_append_to_null`()
Tests appending to an empty container

`test_append_to_pop`()
Tests appending to a populated container

`test_bad_del_object`()
Tests deleting an invalid object

`test_bad_get_object`()
Tests getting an invalid index

`test_bad_set_object`()
Tests setting an object to an invalid index

`test_del_object`()
Tests that an object was deleted.

`test_get_len`()
Tests the len function, ensures it updates after a delete

`test_iterable`()
Tests the iterable generation

`test_set_get_object`()
Tests setting an object in the container and getting it back

CHAPTER

THREE

MODELS

The pysics models used in the analysys

3.1 pyIsentrope

pyIsentrope

Abstract class for an isentrope

3.1.1 Authors

- Stephen Andrews (SA)
- Andrew M. Fraiser (AMF)

3.1.2 Revisions

0 -> Initial class creation (03-16-2016)

class `FUNCLE.pyIsentrope.`**`EOSBump`**(*name='Bump EOS'*, **args*, ***kwargs*)
Model of an ideal isentrope with gausian bumps

`__call__`(*vs*)
Solve the EOS

Calculates the pressure for a given volume, replicates the EOS model but uses underlying equation rather than the spline

Parameters **vs** (*float*) – Specific volume

Returns **pr** – Pressure

Return type float

`__init__`(*name='Bump EOS'*, **args*, ***kwargs*)
Instantiate the bump EOS

Parameters

- **`*args`** – Variable length argument list.
- **`**kwargs`** – Arbitrary keyword arguments.

Keyword Arguments **name** (*str*) – Name if the isentrope *Def 'Bump EOS'*

derivative(*n=1*)
Returns the nth order derrivative

Keyword Arguments **n** (*int*) – The order of the derrivative. *Def 1*

Retrun

d1_fun(function): Function object yeilded first derrivative of pressure w.r.t volume

class FUNCLE.pyIsentrope.**EOSModel**(*p_fun*, *name='Equation of State Spline'*, **args*, ***kwargs*)
Spline based EOS model

__init__(*p_fun*, *name='Equation of State Spline'*, **args*, ***kwargs*)
Arguments

Parameters

- **p_fun** (*function*) – A function defining the initial EOS
- ***args** – Variable length argument list.
- ****kwargs** – Arbitrary keyword arguments.

Keyword Arguments **name** (*str*) – Name of the isentrope *Def 'Equation of State Spline'*

_on_update_prior(*prior*, **args*, ***kwargs*)
Updated the values and statistics of the prior

** Arguments **

•prior -> function: A function which defines the prior EOS shape

class FUNCLE.pyIsentrope.**Isentrope**(*name='Isentrope'*, **args*, ***kwargs*)
Abstract class for an isentrope

__init__(*name='Isentrope'*, **args*, ***kwargs*)

Parameters

- ***args** – Variable length argument list.
- ****kwargs** – Arbitrary keyword arguments.

Keyword Arguments **name** (*str*) – Name if the isentrope *Def 'Isentrope'*

class FUNCLE.pyIsentrope.**Spline**(*x*, *y*, *w=None*, *bbox=[None, None]*, *k=3*, *ext=0*, *check_finite=False*)
Overloaded scipy spline to work with like_eos

Child of the Scipy IU spline class which provides access to details on the knots

get_c()
Return the coefficients for the basis functions

Returns basis function spline coefficients

Return type (numpy.ndarray)

get_t()
Gives the knot locations

Returns knot locations

Return type (numpy.ndarray)

new_c(*c_in*)
Return a new spline with updated coefficients

Return a new Spline_eos instance that is copy of self except that the coefficients for the basis functions are c.

Parameters **c_in** (*numpy.ndarray*) – The new set of spline coefficeints

Return rv(Spline): A copy of self with the coefficients replaced by c

class FUNCLE.pyIsentrope.**TestBumpEOS**(*methodName='runTest'*)
Test of the bump EOS

test_bad_derivative()
Tests that derrivative errors are caught

test_bad_instatntiation()
Test improper instantiation

test_call()
Test that the bump EOS can be called

test_custom_instatntiation()
Test non default instantiation

test_derivative()
Test the derrivative function

test_instantiation()
Tests that the object is properly instantiated

class FUNCLE.pyIsentrope.**TestIsentrope**(*methodName='runTest'*)
Test of the isentrope object

test_custom_instantiation()
Test instantiation with non default values

test_standard_instantiation()
Test basic use of isentrope

class FUNCLE.pyIsentrope.**test_eos_model**(*methodName='runTest'*)
Test the spline EOS functions

setUp()
Create test eos function

test_bad_instantiation()
Test inproper instantiation

test_custom_instantiation()
Tets instantiation with non default values

test_spline_get_c()
Test spline interaction method, get coefficients

test_spline_get_t()
Test spline interaction method, get knots

test_spline_new_c()
Test spline interaction method, set new coefficients

test_standard_instantiation()
Test normal instantiation of the EOS

3.2 pyGun Model

pyGunModel

Toy computational experiment to

3.2.1 Authors

- Stephen Andrews (SA)
- Andrew M. Fraiser (AMF)

3.2.2 Revisions

0 -> Initial class creation (03-16-2016)

class `FUNCLE.pyGunModel.`**`Gun`**(*eos*, *name='Gun Toy Computational Experiment'*, **args*, ***kwargs*)

A toy physics model representing a gun type experiment

`const`

dict – A dictionary of conversion factors

`__init__`(*eos*, *name='Gun Toy Computational Experiment'*, **args*, ***kwargs*)

Instantiate the Experiment object

Parameters **`eos`** (`Isentrope`) – The equation of state model used in the toy computational experiment

Keyword Arguments **name** (*str*) – A name. (Default = 'Gun Toy Computational Experiment')

`_e`(*x*)

Integrates the force up to position x

Parameters **`x`** (*`float`*) – Scalar position

Returns The intergral of the foce over the distance to x

Return type (float)

`_f`(*x*)

Calculates the force on the prjectile

Parameters **`x`** (*`float`*) – The scalar position

Retun (float): The force in dynes

`_shoot`(*t_min*, *t_max*, *n_t*)

Run a simulation and return the results: t, [x,v]

Solves the ODE

$$F(x, v, t) = \frac{d}{dt}(x, v)$$

Parameters

- **`t_min`** (*`float`*) – start time of the solution
- **`t_max`** (*`float`*) – end time of the solution

Returns

(list): elements are

- [0] -> np.ndarray: position
- [1] -> np.ndarray: velocity

`_x_dot`(*x*)

Calculate the projectile velocity

Calculates at a single position x, or if x is an array, calculate the velocity for each element of x

Parameters **x** (*float or np.ndarray*) – scalar position

Return v(np.ndarray): velocity

CHAPTER

FOUR

ANALYSYS

The methods used for the actual optimization

4.1 pyBayesian

pyBayesian

An object to extract properties of the bayesian analysis of experiments

4.1.1 Authors

- Stephen Andrews (SA)
- Andrew M. Fraiser (AMF)

4.1.2 Revisions

0 -> Initial class creation (03-16-2016)

class `FUNCLE.pyBayesian.`**`Bayesian`**(*model*, *data*, *prior*, *name='Bayesian'*, **args*, ***kwargs*)

A calss for performing bayesian inference on a model given data

`sim_exp`

Experiment – The simulated experimental data

`true_exp`

Experiment – The true experimental data

`prior`

Experiment – The prior for the physics model

`get_exp_sens`()

Gets the sensitivity of the experimental data to changes to the EOS

`get_sim_sens`()

Gets the sensitivity of the simulated experiment to the EOS

`update`(*sim_exp=None*, *true_exp=None*, *prior=None*)

Updates the properties of the bayesian analtsis

Keyword Arguments

- **sim_exp** (*Experiment*) – The simulated experimental data (Default None)
- **true_exp** (*Experiment*) – The true experimental data (Default None)

- **prior** (*Experiment*) – The prior for the physics model (Default None)

Returns None

class `FUNCLE.pyBayesian.`**`TestBayesian`**(*methodName='runTest'*)

Test class for the bayesian object

`setUp`()

Setup script for each test

`test_bad_instantiaion`()

Tets impropper instantiation raises the correct errors

`test_instantiation`()

Test that the object can instantiate correctly

CHAPTER FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX