
pyCollocation Documentation

Release 0.3.0-alpha

David R. Pugh

April 30, 2015

1	License	3
2	pycollocation	5
2.1	pycollocation package	5
3	Indices and tables	13
	Python Module Index	15

Contents:

License

The MIT License (MIT)

Copyright (c) 2015 David R. Pugh

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

pycollocation

2.1 pycollocation package

2.1.1 Submodules

2.1.2 pycollocation.boundary_value_problems module

Classes for constructing two-point boundary value problems.

@author : davidrpugh

class pycollocation.boundary_value_problems.**BoundaryValueProblem**

Bases: `object`

Attributes

`boundary_conditions` Boundary conditions for the problem.

boundary_conditions

Boundary conditions for the problem.

Getter Return the boundary conditions for the problem.

Setter Set new boundary conditions for the problem.

Type dict

class pycollocation.boundary_value_problems.**SymbolicBoundaryValueProblem** (*boundary_conditions,*
de-
pen-
dent_vars,
in-
de-
pen-
dent_var,
rhs,
params)

Bases: `pycollocation.boundary_value_problems.BoundaryValueProblem,`
`pycollocation.symbolics.SymbolicBoundaryValueProblemLike,`
`pycollocation.differential_equations.SymbolicDifferentialEquation`

Class for representing two-point boundary value problems.

Attributes

<code>boundary_conditions</code>	Boundary conditions for the problem.
<code>dependent_vars</code>	Model dependent variables.
<code>independent_var</code>	Symbolic variable representing the independent variable.
<code>params</code>	Dictionary of model parameters.
<code>rhs</code>	Symbolic representation of the right-hand side of a system of differential/difference equations.

2.1.3 `pycollocation.differential_equations` module

Classes for constructing systems of ordinary differential equations.

@author : davidrpugh

```
class pycollocation.differential_equations.DifferentialEquation(dependent_vars,  
independent_var,  
rhs, params)  
  
Bases: pycollocation.models.ModelLike
```

Attributes

<code>dependent_vars</code>	Model dependent variables.
<code>independent_var</code>	Symbolic variable representing the independent variable.
<code>params</code>	Dictionary of model parameters.
<code>rhs</code>	Symbolic representation of the right-hand side of a system of differential/difference equations.

```
class pycollocation.differential_equations.SymbolicDifferentialEquation(dependent_vars,  
inde-  
pen-  
dent_var,  
rhs,  
params)  
  
Bases: pycollocation.differential_equations.DifferentialEquation,  
pycollocation.symbolics.SymbolicModelLike
```

Attributes

<code>dependent_vars</code>	Model dependent variables.
<code>independent_var</code>	Symbolic variable representing the independent variable.
<code>params</code>	Dictionary of model parameters.
<code>rhs</code>	Symbolic representation of the right-hand side of a system of differential/difference equations.

2.1.4 `pycollocation.models` module

```
class pycollocation.models.ModelLike  
Bases: object
```

Attributes

<code>dependent_vars</code>	Model dependent variables.
<code>independent_var</code>	Symbolic variable representing the independent variable.
<code>params</code>	Dictionary of model parameters.
<code>rhs</code>	Symbolic representation of the right-hand side of a system of differential/difference equations.

dependent_vars

Model dependent variables.

Getter Return the model dependent variables.

Type list

independent_var

Symbolic variable representing the independent variable.

Getter Return the symbol representing the independent variable.

Type sympy.Symbol

params

Dictionary of model parameters.

Getter Return the current parameter dictionary.

Setter Set a new parameter dictionary.

Type dict

rhs

Symbolic representation of the right-hand side of a system of differential/difference equations.

Getter Return the right-hand side of the system of equations.

Type dict

2.1.5 pycollocation.orthogonal_polynomials module

Classes for solving models using collocation with orthogonal polynomials as the underlying basis functions.

@author: davidrpugh

class pycollocation.orthogonal_polynomials.**OrthogonalPolynomialBasis**

Bases: `object`

Class for constructing orthogonal polynomial basis functions.

Attributes

<code>degrees</code>	Degrees used when constructing the orthogonal polynomials.
<code>domain</code>	Domain over which the approximated solution is valid.
<code>kind</code>	Kind of polynomials to use when constructing the approximation.

degrees

Degrees used when constructing the orthogonal polynomials.

Getter Return the *degrees* attribute.

Type dict

domain

Domain over which the approximated solution is valid.

Getter Return the *domain* attribute.

Type list

kind

Kind of polynomials to use when constructing the approximation.

Getter Return the *kind* of orthogonal polynomials.

Type string

```
class pycollocation.orthogonal_polynomials.OrthogonalPolynomialSolver(model)
    Bases: pycollocation.orthogonal_polynomials.OrthogonalPolynomialBasis,
           pycollocation.solvers.Solver
```

Attributes

<code>coefficients</code>	Coefficients to use when constructing the approximating polynomials.
<code>degrees</code>	Degrees used when constructing the orthogonal polynomials.
<code>derivatives</code>	Derivatives of the approximating basis functions.
<code>domain</code>	Domain over which the approximated solution is valid.
<code>functions</code>	The basis functions used to approximate the solution to the model.
<code>kind</code>	Kind of polynomials to use when constructing the approximation.
<code>model</code>	Symbolic representation of the model to solve.
<code>residual_functions</code>	Residual functions
<code>result</code>	Result object

Methods

<code>solve(kind, coefs_dict, domain[, method])</code>	Solve a boundary value problem using orthogonal collocation.
--	--

solve (*kind*, *coefs_dict*, *domain*, *method*='hybr', ***kwargs*)
Solve a boundary value problem using orthogonal collocation.

2.1.6 pycollocation.solvers module

```
class pycollocation.solvers.Solver(model)
```

Bases: `object`

Base class for all Solvers.

Attributes

<code>coefficients</code>	Coefficients to use when constructing the approximating polynomials.
---------------------------	--

Continued on next page

Table 2.9 – continued from previous page

<code>derivatives</code>	Derivatives of the approximating basis functions.
<code>functions</code>	The basis functions used to approximate the solution to the model.
<code>model</code>	Symbolic representation of the model to solve.
<code>residual_functions</code>	Residual functions
<code>result</code>	Result object

coefficients

Coefficients to use when constructing the approximating polynomials.

Getter Return the *coefficients* attribute.

Type dict

derivatives

Derivatives of the approximating basis functions.

Getter Return the *derivatives* attribute.

Type dict

functions

The basis functions used to approximate the solution to the model.

Getter Return the *functions* attribute.

Type dict

model

Symbolic representation of the model to solve.

Getter Return the current model.

Setter Set a new model to solve.

Type models.Model

residual_functions

Residual functions

Getter Return the current residual functions.

result

Result object

Getter Return the current result object.

Type optimize.Result

2.1.7 pycollocation.symbolics module

Classes for constructing symbolic models.

@author : davidrpugh

class pycollocation.symbolics.SymbolicBase

Bases: object

class pycollocation.symbolics.SymbolicBoundaryValueProblemLike

Bases: pycollocation.symbolics.SymbolicModelLike

class pycollocation.symbolics.SymbolicModelLike

Bases: pycollocation.symbolics.SymbolicBase

2.1.8 pycollocation.version module

2.1.9 pycollocation.visualizers module

Base class for all Visualizer objects.

class `pycollocation.visualizers.Visualizer(solver)`

Bases: `object`

Base class for all Visualizer objects.

Attributes

<code>interpolation_knots</code>	Interpolation knots to use when computing the solution.
<code>normalized_residuals</code>	Absolute values of the solution residuals normalized by the value of the solution.
<code>residuals</code>	Solution residuals.
<code>result</code>	An instance of the <code>optimize.optimize.OptimizeResult</code> class that stores the raw output of a <code>solvers.Solver</code> object.
<code>solution</code>	Solution to the model represented as a Pandas <i>DataFrame</i> .

interpolation_knots

Interpolation knots to use when computing the solution.

Getter Return the array of interpolation knots.

Setter Set a new array of interpolation knots.

Type `numpy.ndarray`

normalized_residuals

Absolute values of the solution residuals normalized by the value of the solution.

Getter Return the normalized solution residuals.

Type `pandas.DataFrame`

residuals

Solution residuals.

Getter Return the solution residuals.

Type `pandas.DataFrame`

result

An instance of the `optimize.optimize.OptimizeResult` class that stores the raw output of a `solvers.Solver` object.

Getter Return the *result* attribute.

Type `optimize.optimize.OptimizeResult`

solution

Solution to the model represented as a Pandas *DataFrame*.

Getter Return the *DataFrame* representing the current solution.

Type `pandas.DataFrame`

2.1.10 Module contents

Objects imported here will live in the *pycollocation* namespace

```
class pycollocation.SymbolicBoundaryValueProblem (boundary_conditions, dependent_vars,  
                                                independent_var, rhs, params)  
    Bases: pycollocation.boundary_value_problems.BoundaryValueProblem,  
            pycollocation.symbolics.SymbolicBoundaryValueProblemLike,  
            pycollocation.differential_equations.SymbolicDifferentialEquation
```

Class for representing two-point boundary value problems.

Attributes

<code>boundary_conditions</code>	Boundary conditions for the problem.
<code>dependent_vars</code>	Model dependent variables.
<code>independent_var</code>	Symbolic variable representing the independent variable.
<code>params</code>	Dictionary of model parameters.
<code>rhs</code>	Symbolic representation of the right-hand side of a system of differential/difference equations.

```
class pycollocation.OrthogonalPolynomialSolver (model)  
    Bases: pycollocation.orthogonal_polynomials.OrthogonalPolynomialBasis,  
            pycollocation.solvers.Solver
```

Attributes

<code>coefficients</code>	Coefficients to use when constructing the approximating polynomials.
<code>degrees</code>	Degrees used when constructing the orthogonal polynomials.
<code>derivatives</code>	Derivatives of the approximating basis functions.
<code>domain</code>	Domain over which the approximated solution is valid.
<code>functions</code>	The basis functions used to approximate the solution to the model.
<code>kind</code>	Kind of polynomials to use when constructing the approximation.
<code>model</code>	Symbolic representation of the model to solve.
<code>residual_functions</code>	Residual functions
<code>result</code>	Result object

Methods

<code>solve(kind, coefs_dict, domain[, method])</code>	Solve a boundary value problem using orthogonal collocation.
--	--

```
solve (kind, coefs_dict, domain, method='hybr', **kwargs)  
    Solve a boundary value problem using orthogonal collocation.
```

```
class pycollocation.Visualizer (solver)  
    Bases: object
```

Base class for all Visualizer objects.

Attributes

<code>interpolation_knots</code>	Interpolation knots to use when computing the solution.
<code>normalized_residuals</code>	Absolute values of the solution residuals normalized by the value of the solution.
<code>residuals</code>	Solution residuals.
<code>result</code>	An instance of the <i>optimize.optimize.OptimizeResult</i> class that stores the raw output of a <i>solvers.Solver</i> .
<code>solution</code>	Solution to the model represented as a Pandas <i>DataFrame</i> .

interpolation_knots

Interpolation knots to use when computing the solution.

Getter Return the array of interpolation knots.

Setter Set a new array of interpolation knots.

Type `numpy.ndarray`

normalized_residuals

Absolute values of the solution residuals normalized by the value of the solution.

Getter Return the normalized solution residuals.

Type `pandas.DataFrame`

residuals

Solution residuals.

Getter Return the solution residuals.

Type `pandas.DataFrame`

result

An instance of the *optimize.optimize.OptimizeResult* class that stores the raw output of a *solvers.Solver* object.

Getter Return the *result* attribute.

Type `optimize.optimize.OptimizeResult`

solution

Solution to the model represented as a Pandas *DataFrame*.

Getter Return the *DataFrame* representing the current solution.

Type `pandas.DataFrame`

Indices and tables

- *genindex*
- *modindex*
- *search*

p

- `pycollocation`, [11](#)
- `pycollocation.boundary_value_problems`,
[5](#)
- `pycollocation.differential_equations`, [6](#)
- `pycollocation.models`, [6](#)
- `pycollocation.orthogonal_polynomials`, [7](#)
- `pycollocation.solvers`, [8](#)
- `pycollocation.symbolics`, [9](#)
- `pycollocation.visualizers`, [10](#)

B

boundary_conditions (pycollocation.boundary_value_problems.BoundaryValueProblem attribute), 5
BoundaryValueProblem (class in pycollocation.boundary_value_problems), 5

C

coefficients (pycollocation.solvers.Solver attribute), 9

D

degrees (pycollocation.orthogonal_polynomials.OrthogonalPolynomialBasis attribute), 7
dependent_vars (pycollocation.models.ModelLike attribute), 7
derivatives (pycollocation.solvers.Solver attribute), 9
DifferentialEquation (class in pycollocation.differential_equations), 6
domain (pycollocation.orthogonal_polynomials.OrthogonalPolynomialBasis attribute), 8

F

functions (pycollocation.solvers.Solver attribute), 9

I

independent_var (pycollocation.models.ModelLike attribute), 7
interpolation_knots (pycollocation.Visualizer attribute), 12
interpolation_knots (pycollocation.visualizers.Visualizer attribute), 10

K

kind (pycollocation.orthogonal_polynomials.OrthogonalPolynomialBasis attribute), 8

M

model (pycollocation.solvers.Solver attribute), 9
ModelLike (class in pycollocation.models), 6

N

normalized_residuals (pycollocation.Visualizer attribute), 12
normalized_residuals (pycollocation.visualizers.Visualizer attribute), 10

O

OrthogonalPolynomialBasis (class in pycollocation.orthogonal_polynomials), 7
OrthogonalPolynomialSolver (class in pycollocation), 11
OrthogonalPolynomialSolver (class in pycollocation.orthogonal_polynomials), 8

P

params (pycollocation.models.ModelLike attribute), 7
pycollocation (module), 11
pycollocation.boundary_value_problems (module), 5
pycollocation.differential_equations (module), 6
pycollocation.models (module), 6
pycollocation.orthogonal_polynomials (module), 7
pycollocation.solvers (module), 8
pycollocation.symbolics (module), 9
pycollocation.visualizers (module), 10

R

residual_functions (pycollocation.solvers.Solver attribute), 9
residuals (pycollocation.Visualizer attribute), 12
residuals (pycollocation.visualizers.Visualizer attribute), 10
result (pycollocation.solvers.Solver attribute), 9
result (pycollocation.Visualizer attribute), 12
result (pycollocation.visualizers.Visualizer attribute), 10
rhs (pycollocation.models.ModelLike attribute), 7

S

solution (pycollocation.Visualizer attribute), 12
solution (pycollocation.visualizers.Visualizer attribute), 10

`solve()` (`pycollocation.orthogonal_polynomials.OrthogonalPolynomialSolver`
method), 8

`solve()` (`pycollocation.OrthogonalPolynomialSolver`
method), 11

`Solver` (class in `pycollocation.solvers`), 8

`SymbolicBase` (class in `pycollocation.symbolics`), 9

`SymbolicBoundaryValueProblem` (class in `pycollocation`), 11

`SymbolicBoundaryValueProblem` (class in `pycollocation.boundary_value_problems`), 5

`SymbolicBoundaryValueProblemLike` (class in `pycollocation.symbolics`), 9

`SymbolicDifferentialEquation` (class in `pycollocation.differential_equations`), 6

`SymbolicModelLike` (class in `pycollocation.symbolics`), 9

V

`Visualizer` (class in `pycollocation`), 11

`Visualizer` (class in `pycollocation.visualizers`), 10