

---

# **pyCollocation Documentation**

*Release 0.3.0-alpha*

**David R. Pugh**

July 20, 2016



<b>1 License</b>	<b>3</b>
<b>2 pycollocation</b>	<b>5</b>
2.1 pycollocation package . . . . .	5
<b>3 Indices and tables</b>	<b>9</b>
<b>Python Module Index</b>	<b>11</b>



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.





## 2.1 pycollocation package

### 2.1.1 Submodules

### 2.1.2 pycollocation.boundary\_value\_problems module

### 2.1.3 pycollocation.differential\_equations module

### 2.1.4 pycollocation.models module

### 2.1.5 pycollocation.orthogonal\_polynomials module

### 2.1.6 pycollocation.solvers module

Objects imported here will live in the *pycollocation.solvers* namespace

```
class pycollocation.solvers.LeastSquaresSolver(basis_functions)
    Bases: pycollocation.solvers.solvers.Solver
```

#### Attributes

---

`basis_functions` Functions used to approximate the solution to a boundary value problem.

---

#### Methods

---

`solve`(*basis\_kwargs*, *boundary\_points*, ...) Solve a boundary value problem using the collocation method.

---

**solve** (*basis\_kwargs*, *boundary\_points*, *coefs\_array*, *nodes*, *problem*, *\*\*solver\_options*)  
Solve a boundary value problem using the collocation method.

**Parameters** `basis_kwargs` : dict

Dictionary of keyword arguments used to build basis functions.

`coefs_array` : numpy.ndarray

Array of coefficients for basis functions defining the initial condition.

**problem** : `bvp.TwoPointBVPLike`

A two-point boundary value problem (BVP) to solve.

**solver\_options** : dict

Dictionary of options to pass to the non-linear equation solver.

**class** `pycollocation.solvers.Solver` (*basis\_functions*)

Bases: `pycollocation.solvers.solvers.SolverLike`

### Attributes

---

`basis_functions` Functions used to approximate the solution to a boundary value problem.

---

### Methods

---

`solve`(*basis\_kwargs*, *boundary\_points*, ...) Solve a boundary value problem using the collocation method.

---

**solve** (*basis\_kwargs*, *boundary\_points*, *coefs\_array*, *nodes*, *problem*, *\*\*solver\_options*)

Solve a boundary value problem using the collocation method.

**Parameters** *basis\_kwargs* : dict

Dictionary of keyword arguments used to build basis functions.

**coefs\_array** : `numpy.ndarray`

Array of coefficients for basis functions defining the initial condition.

**problem** : `bvp.TwoPointBVPLike`

A two-point boundary value problem (BVP) to solve.

**solver\_options** : dict

Dictionary of options to pass to the non-linear equation solver.

**class** `pycollocation.solvers.Solution` (*basis\_kwargs*, *functions*, *nodes*, *problem*, *residual\_function*, *result*)

Bases: `pycollocation.solvers.solutions.SolutionLike`

Class representing the solution to a Boundary Value Problem (BVP).

### Attributes

---

`basis_kwargs`

`functions`

`nodes`

`problem`

`residual_function`

`result`

---

## Methods

---

<code>evaluate_residual(points)</code>	
<code>evaluate_solution(points)</code>	
<code>normalize_residuals(points)</code>	Normalize residuals by the level of the variable.

---

**evaluate\_residual** (*points*)

**evaluate\_solution** (*points*)

**normalize\_residuals** (*points*)

Normalize residuals by the level of the variable.

**class** `pycollocation.solvers.SolutionLike`

Bases: `object`

## Attributes

---

<code>basis_kwargs</code>
<code>functions</code>
<code>nodes</code>
<code>problem</code>
<code>residual_function</code>
<code>result</code>

---

**basis\_kwargs**

**functions**

**nodes**

**problem**

**residual\_function**

**result**

**class** `pycollocation.solvers.SolverLike`

Bases: `object`

Class describing the protocol the all SolverLike objects should satisfy.

## Notes

Subclasses should implement *solve* method as described below.

## Attributes

---

<code>basis_functions</code>	Functions used to approximate the solution to a boundary value problem.
------------------------------	---

---

## Methods

---

`solve(basis_kwargs, boundary_points, ...)` Solve a boundary value problem using the collocation method.

---

### **basis\_functions**

Functions used to approximate the solution to a boundary value problem.

**Getter** Return the current basis functions.

**Type** `basis_functions.BasisFunctions`

**solve** (`basis_kwargs`, `boundary_points`, `coefs_array`, `nodes`, `problem`, `**solver_options`)

Solve a boundary value problem using the collocation method.

**Parameters** `basis_kwargs` : dict

Dictionary of keyword arguments used to build basis functions.

**coefs\_array** : `numpy.ndarray`

Array of coefficients for basis functions defining the initial condition.

**problem** : `bvp.TwoPointBVPLike`

A two-point boundary value problem (BVP) to solve.

**solver\_options** : dict

Dictionary of options to pass to the non-linear equation solver.

## **2.1.7 pycollocation.symbolics module**

## **2.1.8 pycollocation.version module**

## **2.1.9 pycollocation.visualizers module**

## **2.1.10 Module contents**

Objects imported here will live in the `pycollocation` namespace

---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



**p**

`pycollocation`, 8

`pycollocation.solvers`, 5





**B**

basis\_functions (pycollocation.solvers.SolverLike attribute), 8  
basis\_kwargs (pycollocation.solvers.SolutionLike attribute), 7  
solve() (pycollocation.solvers.SolverLike method), 8  
Solver (class in pycollocation.solvers), 6  
SolverLike (class in pycollocation.solvers), 7

**E**

evaluate\_residual() (pycollocation.solvers.Solution method), 7  
evaluate\_solution() (pycollocation.solvers.Solution method), 7

**F**

functions (pycollocation.solvers.SolutionLike attribute), 7

**L**

LeastSquaresSolver (class in pycollocation.solvers), 5

**N**

nodes (pycollocation.solvers.SolutionLike attribute), 7  
normalize\_residuals() (pycollocation.solvers.Solution method), 7

**P**

problem (pycollocation.solvers.SolutionLike attribute), 7  
pycollocation (module), 8  
pycollocation.solvers (module), 5

**R**

residual\_function (pycollocation.solvers.SolutionLike attribute), 7  
result (pycollocation.solvers.SolutionLike attribute), 7

**S**

Solution (class in pycollocation.solvers), 6  
SolutionLike (class in pycollocation.solvers), 7  
solve() (pycollocation.solvers.LeastSquaresSolver method), 5  
solve() (pycollocation.solvers.Solver method), 6