Peer-methods

Generated by Doxygen 1.8.17

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 return_values Struct Reference

Public Attributes

- double * **yT**
- int **yT_size**
- double * **y**
- int **y_rows**
- int **y_cols**
- double * **t**
- int **t_size**

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

- /home/vincenzo/Github/Peer-Methods/peer_methods_C/peerMethods/include/peerMethods.h

Chapter 4

File Documentation

4.1 /home/vincenzo/Github/Peer-Methods/peer_methods_C/peer↩ Methods/include/peerMethods.h File Reference

The library provides an implementation for the main function for solving peer method.

Classes

- struct return_values

Macros

- #define **STAGES** 2

Functions

- void initReturnStruct (return_values ∗rv)

  Initialize the struct return_values.
- int saveResultsInFile (const char ∗fileName, return_values result)

  Save the struct return_values in a file.
- void computeLMatrix (double ∗∗L, int ∗LSize, double Delta_x)

  Build the matrix L. This is an helping function that builds the matrix L.
- double ∗ Sherratt (const double ∗y0, int y0Size, const double ∗L, int Lsize, int ∗sherrattSize)

  Applies the Sherratt method.
- double ∗ RungeKutta4th (double h, double t0, const double ∗y0, int y0Size, const double ∗L, int Lsize, int ∗ySize)

  Implicit fourth order method to solving ODE (Ordinary Differential Equation).
- void fPeerClassic_twoStages (int N, double ∗t_span, int t_span_size, const double ∗L, int Lsize, const double ∗y0, int y0_size, return_values ∗collect_result)

  Compute the PDE (Partial Differential Equation) using the MOL (Method Of Lines). The function computes the PDE (Partial Differential Equation) using MOL (Method Of Lines) and deriving a large system of ODE (Ordinary Differential Equation). Than, it solves the ODE system using the Runge Kutta method of the fourth order.
- void ∗ Malloc (size_t size)

Function wrapper for malloc() function.

- void * Calloc (size_t nmemb, size_t size)

  Function wrapper for calloc() function.
- void **initializeRandomVector** (double *vector, int N)
- void **initializeRandomMatrix** (double *matrix, int M, int N)
- int **initMatrixByRowWithValuesFromVector** (double *matrix, int M, int N, double *vector, int vector_size)
- void **initVectorWAnotherVector** (double *newVector, double *oldVector, int n)
- void **freeEverything** (void *arg1,...)
- double * **intervalDiscretization** (double first, double last, double step, int *N)
- double * **eyeD** (int N)
- double * **onesD** (int N)
- double * **zerosD** (int N)
- double * **zerosMatrixD** (int M, int N)
- double * **diagD** (double *vector, int size, int k, int *matrix_size)
- double * **packThreeMatrices** (int n, double *A, double *B, double *C)
- double * **threeBlockDiagD** (int n, double *A, double *B, double *C, int *blckSize)
- double * **packThreeVectors** (int n, double *A, double *B, double *C, int *newDimension)
- double * **linspace** (double x1, double x2, int n)

Variables

- double **a**
- double **B1**
- double **B2**
- double **F**
- double **H**
- double **S**
- double **d**
- double **D**
- double **L**
- int **M**

4.1.1 Detailed Description

The library provides an implementation for the main function for solving peer method.

Author

Vincenzo Iannucci

Version

0.1

Date

2022-11-29

4.1.2 Function Documentation

4.1.2.1 Calloc()

```
void* Calloc (
            size_t nmemb,
            size_t size )
```

Function wrapper for calloc() function.

Parameters

in	*nmemb*	number of elements to allocate
in	*size*	Size of the memory allocated

Returns

a pointer to the allocated memory

4.1.2.2 computeLMatrix()

```
void computeLMatrix (
            double ** L,
            int * LSize,
            double Delta_x )
```

Build the matrix L. This is an helping function that builds the matrix L.

Parameters

out	*L*	returning pointer to the matrix
in	*LSize*	return the size of the matrix
in	*Delta_x*	the value of the delta

4.1.2.3 fPeerClassic_twoStages()

```
void fPeerClassic_twoStages (
            int N,
            double * t_span,
            int t_span_size,
            const double * L,
            int Lsize,
            const double * y0,
            int y0_size,
            return_values * collect_result )
```

Compute the PDE (Partial Differential Equation) using the MOL (Method Of Lines). The function computes the PDE (Partial Differential Equation) using MOL (Method Of Lines) and deriving a large system of ODE (Ordinary Differential Equation). Than, it solves the ODE system using the Runge Kutta method of the fourth order.

Parameters

in	*N*	the size of the temporal grid
in	*t_span*	an array representing the temporal grid itself
in	*t_span_size*	the spatial dimension of the temporal grid

Parameters

in	*L*	pointer to the matrix L
in	*LSize*	size of the matrix
in	*y0*	pointer to the y0 vector
in	*y0Size*	size of the y0 vector
out	*collect_result*	size of the result vector

Returns

a pointer to the y resulting vector.

4.1.2.4 initReturnStruct()

```
void initReturnStruct (
            return_values * rv )
```

Initialize the struct return_values.

Parameters

rv	pointer to the struct return_values

4.1.2.5 Malloc()

```
void* Malloc (
            size_t size )
```

Function wrapper for malloc() function.

Parameters

in	*size*	Size of the memory allocated

Returns

a pointer to the allocated memory

4.1.2.6 RungeKutta4th()

```
double* RungeKutta4th (
            double h,
```

```
            double t0,
            const double * y0,
            int y0Size,
            const double * L,
            int Lsize,
            int * ySize )
```

Implicit fourth order method to solving ODE (Ordinary Differential Equation).

Parameters

in	*h*	number of conditions to achieve the solution
in	*t0*	starting time
in	*y0*	pointer to the y0 vector
in	*y0Size*	size of the y0 vector
in	*L*	pointer to the matrix L
in	*LSize*	size of the matrix
out	*ySize*	size of the result vector

Returns

a pointer to the y resulting vector.

4.1.2.7 saveResultsInFile()

```
int saveResultsInFile (
            const char * fileName,
            return_values result )
```

Save the struct return_values in a file.

Parameters

in	*fileName*	the name of the file
out	*rv*	pointer to the struct return

Returns

0 if ok, 1 otherwise.

4.1.2.8 Sherratt()

```
double* Sherratt (
            const double * y0,
            int y0Size,
```

```
            const double * L,
            int Lsize,
            int * sherrattSize )
```

Applies the Sherratt method.

Parameters

in	*y0*	pointer to the y0 vector
in	*y0Size*	size of the y0 vector
in	*L*	pointer to the matrix L
in	*LSize*	size of the matrix
out	*sherrattSize*	returing size of the vector calculated by the function

Returns

a pointer the resulting vector after applying the Sherratt method.

Index