

GPU solvers for SLAE

1.0

Generated by Doxygen 1.8.13

Contents

1	Namespace Index	1
1.1	Namespace List	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	Namespace Documentation	7
4.1	Gauss Namespace Reference	7
4.1.1	Detailed Description	7
4.2	Gaussian Namespace Reference	7
4.2.1	Detailed Description	7
4.3	jacobi Namespace Reference	7
4.3.1	Detailed Description	7
4.4	lu_decomposition Namespace Reference	7
4.4.1	Detailed Description	8
4.5	Matrix Namespace Reference	8
4.5.1	Detailed Description	8
4.6	sparseMatrices Namespace Reference	8
4.6.1	Detailed Description	8

5	Class Documentation	9
5.1	block_operations_tab.BlockTab Class Reference	9
5.2	gaussian_elimination.GaussianElimination Class Reference	9
5.2.1	Member Function Documentation	10
5.2.1.1	gaussian_elimination()	10
5.2.1.2	start()	10
5.3	gaussian_elimination_tab.GaussianEliminationTab Class Reference	11
5.4	gauss_jordan.GaussJordan Class Reference	12
5.4.1	Member Function Documentation	12
5.4.1.1	gauss_jordan()	12
5.4.1.2	normalize()	13
5.4.1.3	start()	13
5.5	serial_gauss_jordan.GaussJordanSerial Class Reference	14
5.5.1	Member Function Documentation	14
5.5.1.1	elimination()	14
5.6	gauss_jordan_tab.GaussJordanTab Class Reference	15
5.7	gaussian_lu_decomposition.GaussianLUdecomposition Class Reference	15
5.7.1	Member Function Documentation	15
5.7.1.1	gaussian_lu_decomposition()	15
5.7.1.2	gen_identity_matrix()	16
5.7.1.3	get_determinant()	17
5.7.1.4	get_inverse()	17
5.7.1.5	get_solution()	18
5.7.1.6	start()	18
5.8	jacobi_parallel_chunks.JacobiParallel Class Reference	19
5.8.1	Member Function Documentation	19
5.8.1.1	jacobi()	20
5.9	jacobi_parallel.JacobiParallel Class Reference	21
5.9.1	Member Function Documentation	21
5.9.1.1	get_error()	21

5.9.1.2	jacobi()	22
5.9.1.3	start()	23
5.10	jacobi_tab.JacobiTab Class Reference	23
5.11	lu_decomposition_tab.LUDecompositionTab Class Reference	24
5.12	matrix_generator.MatrixGenerator Class Reference	25
5.12.1	Member Function Documentation	25
5.12.1.1	gen_antisymmetric_matrix()	25
5.12.1.2	gen_band_matrix()	26
5.12.1.3	gen_diagonal_matrix()	26
5.12.1.4	gen_dominant()	26
5.12.1.5	gen_identity_matrix()	27
5.12.1.6	gen_lower_matrix()	27
5.12.1.7	gen_random_matrix()	27
5.12.1.8	gen_scalar_matrix()	28
5.12.1.9	gen_symmetric_matrix()	28
5.12.1.10	gen_upper_matrix()	28
5.12.1.11	gen_vector()	30
5.13	matrix_generator_tab.MatrixGeneratorTab Class Reference	30
5.14	guiNum.PyApp Class Reference	31
5.15	serial_gaussian_elimination.SerialGaussianElimination Class Reference	32
5.15.1	Member Function Documentation	32
5.15.1.1	elimination()	32
5.15.1.2	partial_pivot()	32
5.16	jacobi_serial.SerialJacobi Class Reference	33
5.16.1	Member Function Documentation	34
5.16.1.1	get_D_and_U()	34
5.16.1.2	get_error()	34
5.16.1.3	get_inverse()	35
5.16.1.4	jacobi()	35
5.16.1.5	multiply_matrix_matrix()	37

5.16.1.6	multiply_matrix_vector()	37
5.16.1.7	relaxation()	38
5.16.1.8	sum_vectors()	39
5.17	serial_decomposition_LU.SerialLUDecomposition Class Reference	39
5.17.1	Member Function Documentation	39
5.17.1.1	decomposition_LU()	39
5.17.1.2	solve_system()	40
5.18	sparse_matrix.SparseMatrix Class Reference	40
5.18.1	Member Function Documentation	41
5.18.1.1	create_sparse_matrix()	41
5.18.1.2	gen_vector()	41
5.18.1.3	load_sparse_matrix()	42
5.18.1.4	multiply()	42
5.19	sm_testCSR.SparseMatrix Class Reference	42
5.20	sm_test.SparseMatrix Class Reference	43
5.21	sparse_matrix_tab.SparseMatrixTab Class Reference	43
Index		45

Chapter 1

Namespace Index

1.1 Namespace List

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

Gauss	Jordan Solve a system of linear algebraic equations by using the Gauss Jordan Elimination method	7
Gaussian	Elimination Solve a system of linear algebraic equations by using the Gaussian Elimination method	7
jacobi	Solve a system of linear algebraic equations by using the Jacobi Iterative method	7
lu_decomposition	Decomposes a matrix A into two matrices L and U	7
Matrix	Generator Generate different types of matrices	8
sparseMatrices	Represents a matrix with CSR format	8

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

block_operations_tab.BlockTab	9
gaussian_elimination.GaussianElimination	9
gaussian_elimination_tab.GaussianEliminationTab	11
gauss_jordan.GaussJordan	12
serial_gauss_jordan.GaussJordanSerial	14
gauss_jordan_tab.GaussJordanTab	15
gaussian_lu_decomposition.GaussianLUdecomposition	15
jacobi_parallel_chunks.JacobiParallel	19
jacobi_parallel.JacobiParallel	21
jacobi_tab.JacobiTab	23
lu_decomposition_tab.LUdecompositionTab	24
matrix_generator.MatrixGenerator	25
matrix_generator_tab.MatrixGeneratorTab	30
serial_gaussian_elimination.SerialGaussianElimination	32
jacobi_serial.SerialJacobi	33
serial_decomposition_LU.SerialLUdecomposition	39
sparse_matrix.SparseMatrix	40
sm_testCSR.SparseMatrix	42
sm_test.SparseMatrix	43
sparse_matrix_tab.SparseMatrixTab	43
Window	
guiNum.PyApp	31

Chapter 3

Class Index

3.1 Class List

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

block_operations_tab.BlockTab	9
gaussian_elimination.GaussianElimination	9
gaussian_elimination_tab.GaussianEliminationTab	11
gauss_jordan.GaussJordan	12
serial_gauss_jordan.GaussJordanSerial	14
gauss_jordan_tab.GaussJordanTab	15
gaussian_lu_decomposition.GaussianLUdecomposition	15
jacobi_parallel_chunks.JacobiParallel	19
jacobi_parallel.JacobiParallel	21
jacobi_tab.JacobiTab	23
lu_decomposition_tab.LUdecompositionTab	24
matrix_generator.MatrixGenerator	25
matrix_generator_tab.MatrixGeneratorTab	30
guiNum.PyApp	31
serial_gaussian_elimination.SerialGaussianElimination	32
jacobi_serial.SerialJacobi	33
serial_decomposition_LU.SerialLUdecomposition	39
sparse_matrix.SparseMatrix	40
sm_testCSR.SparseMatrix	42
sm_test.SparseMatrix	43
sparse_matrix_tab.SparseMatrixTab	43

Chapter 4

Namespace Documentation

4.1 Gauss Namespace Reference

Jordan Solve a system of linear algebraic equations by using the [Gauss](#) Jordan Elimination method.

4.1.1 Detailed Description

Jordan Solve a system of linear algebraic equations by using the [Gauss](#) Jordan Elimination method.

4.2 Gaussian Namespace Reference

Elimination Solve a system of linear algebraic equations by using the [Gaussian](#) Elimination method.

4.2.1 Detailed Description

Elimination Solve a system of linear algebraic equations by using the [Gaussian](#) Elimination method.

4.3 jacobi Namespace Reference

Solve a system of linear algebraic equations by using the Jacobi Iterative method.

4.3.1 Detailed Description

Solve a system of linear algebraic equations by using the Jacobi Iterative method.

4.4 lu_decomposition Namespace Reference

Decomposes a matrix A into two matrices L and U.

4.4.1 Detailed Description

Decomposes a matrix A into two matrices L and U .

4.5 Matrix Namespace Reference

Generator Generate different types of matrices.

4.5.1 Detailed Description

Generator Generate different types of matrices.

4.6 sparseMatrices Namespace Reference

Represents a matrix with CSR format.

4.6.1 Detailed Description

Represents a matrix with CSR format.

Chapter 5

Class Documentation

5.1 `block_operations_tab.BlockTab` Class Reference

Public Member Functions

- `def __init__ (self)`
- `def get_tab (self)`
- `def load_matrix (self, widget, data=None)`
- `def load_vector (self, widget, data=None)`
- `def jacobi_by_blocks (self, widget, data=None)`
- `def save (self, widget, data=None)`

Public Attributes

- `niter_entry`
- `A_matrix`
- `b_vector`
- `x_vector`
- `size_entry`
- `rows_entry`
- `tol_entry`

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

- `block_operations/block_operations_tab.py`

5.2 `gaussian_elimination.GaussianElimination` Class Reference

Public Member Functions

- `def gaussian_elimination (Ab, size, i)`
*Performs **Gaussian** elimination for each row of a column.*
- `def start (self, A_matrix, b_matrix)`
*Launches parallel **Gaussian** elimination for a SLAE and returns its answer.*

Static Public Attributes

- **target**
- **nopython**

5.2.1 Member Function Documentation

5.2.1.1 gaussian_elimination()

```
def gaussian_elimination.GaussianElimination.gaussian_elimination (
    Ab,
    size,
    i )
```

Performs [Gaussian](#) elimination for each row of a column.

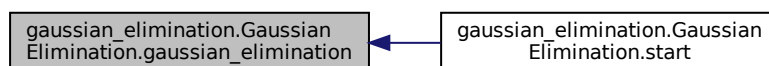
Parameters

<i>A</i>	Augmented matrix representing a SLAE.
<i>size</i>	Size of coefficiente matrix.
<i>i</i>	Integer representing the current column in which all threads are performing row operations.

Returns

None

Here is the caller graph for this function:



5.2.1.2 start()

```
def gaussian_elimination.GaussianElimination.start (
    self,
    A_matrix,
    b_matrix )
```

Launches parallel [Gaussian](#) elimination for a SLAE and returns its answer.

Parameters

<i>A_matrix</i>	Coefficient matrix of a SLAE.
<i>b_matrix</i>	Linearly independent vector of a SLAE.

Returns

None

Here is the call graph for this function:



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

- gaussian_elimination/gaussian_elimination.py

5.3 gaussian_elimination_tab.GaussianEliminationTab Class Reference

Public Member Functions

- def **__init__** (self)
- def **get_tab** (self)
- def **load_matrix** (self, widget, data=None)
- def **load_vector** (self, widget, data=None)
- def **gaussParallel** (self, widget, data=None)
- def **gaussSerial** (self, widget, data=None)
- def **save** (self, widget, data=None)

Public Attributes

- **gaussian_elimination**
- **serial_gaussian_elimination**
- **A_matrix**
- **b_vector**
- **x_vector**

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

- gaussian_elimination/gaussian_elimination_tab.py

5.4 gauss_jordan.GaussJordan Class Reference

Public Member Functions

- def `gauss_jordan` (A, size, i)
Performs [Gauss](#) Jordan elimination for each row of a column.
- def `normalize` (A, size)
Ensures every diagonal element of the augmented matrix A is set to one.
- def `start` (self, A_matrix, b_vector)
Launches parallel [Gauss](#) Jordan elimination for a SLAE and returns its answer.

5.4.1 Member Function Documentation

5.4.1.1 gauss_jordan()

```
def gauss_jordan.GaussJordan.gauss_jordan (
    A,
    size,
    i )
```

Performs [Gauss](#) Jordan elimination for each row of a column.

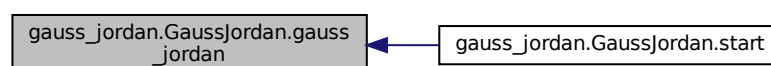
Parameters

<i>A</i>	Augmented matrix representing a SLAE.
<i>size</i>	Size of coefficiente matrix.
<i>i</i>	Integer representing the current column in which all threads are performing row operations.

Returns

None

Here is the caller graph for this function:



5.4.1.2 normalize()

```
def gauss_jordan.GaussJordan.normalize (
    A,
    size )
```

Ensures every diagonal element of the augmented matrix A is set to one.

Parameters

<i>A</i>	Augmented matrix representing a SLAE.
<i>size</i>	Size of coefficiente matrix.

Returns

None

Here is the caller graph for this function:



5.4.1.3 start()

```
def gauss_jordan.GaussJordan.start (
    self,
    A_matrix,
    b_vector )
```

Launches parallel [Gauss](#) Jordan elimination for a SLAE and returns its answer.

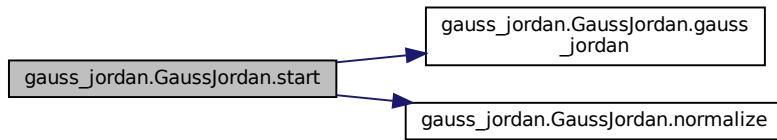
Parameters

<i>A_matrix</i>	Coefficient matrix of a SLAE.
<i>b_vector</i>	Linearly independent vector of a SLAE.

Returns

float64[:]

Here is the call graph for this function:



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

- `gauss_jordan/gauss_jordan.py`

5.5 serial_gauss_jordan.GaussJordanSerial Class Reference

Public Member Functions

- def `elimination` (self, A, b)
Takes a system of linear equations represented by a matrix and a vector and returns the answer applying Gauss-Jordan method.

5.5.1 Member Function Documentation

5.5.1.1 `elimination()`

```
def serial_gauss_jordan.GaussJordanSerial.elimination (
    self,
    A,
    b )
```

Takes a system of linear equations represented by a matrix and a vector and returns the answer applying Gauss-Jordan method.

Returns

A The coefficient matrix of the system.
 b The linearly independent vector.
 float128[:]

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

- `gauss_jordan/serial_gauss_jordan.py`

5.6 gauss_jordan_tab.GaussJordanTab Class Reference

Public Member Functions

- def **__init__** (self)
- def **get_tab** (self)
- def **load_matrix** (self, widget, data=None)
- def **load_vector** (self, widget, data=None)
- def **gaussParallel** (self, widget, data=None)
- def **gaussSerial** (self, widget, data=None)
- def **save** (self, widget, data=None)

Public Attributes

- **gauss_jordan**
- **gauss_jordan_serial**
- **A_matrix**
- **b_vector**
- **x_vector**

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

- gauss_jordan/ gauss_jordan_tab.py

5.7 gaussian_lu_decomposition.GaussianLUdecomposition Class Reference

Public Member Functions

- def **gaussian_lu_decomposition** (A, L, size, i)
Performs [Gaussian](#) LU elimination.
- def **start** (self, A_matrix)
Decomposes A_matrix into two matrices L and U.
- def **get_solution** (self, L, U, b)
Solves a LU system.
- def **gen_identity_matrix** (self, size)
Creates an identity matrix given a size.
- def **get_inverse** (self, L, U)
Returns the inverse of a given matrix by means of LU decomposition.
- def **get_determinant** (self, L, U)
Returns the determinant of a given matrix by means of LU decomposition.

5.7.1 Member Function Documentation

5.7.1.1 gaussian_lu_decomposition()

```
def gaussian_lu_decomposition.GaussianLUdecomposition.gaussian_lu_decomposition (
    A,
    L,
    size,
    i )
```

Performs [Gaussian](#) LU elimination.

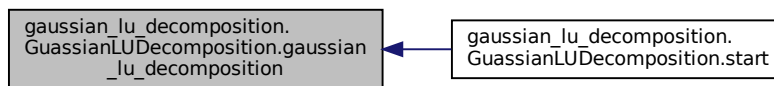
Parameters

<i>A</i>	Coefficient matrix A.
<i>L</i>	Matrix in which to store the multipliers.
<i>size</i>	Size of coefficient matrix.
<i>i</i>	Integer representing the current column in which all threads are performing row operations.

Returns

None

Here is the caller graph for this function:

5.7.1.2 `gen_identity_matrix()`

```
def gaussian_lu_decomposition.GuassianLUdecomposition.gen_identity_matrix (
    self,
    size )
```

Creates an identity matrix given a size.

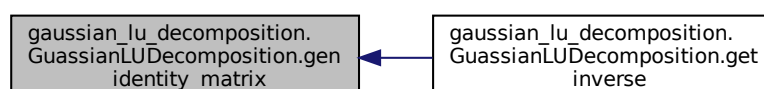
Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float64[:,:]

Here is the caller graph for this function:



5.7.1.3 get_determinant()

```
def gaussian_lu_decomposition.GuassianLUDecomposition.get_determinant (
    self,
    L,
    U )
```

Returns the determinant of a given matrix by means of LU decomposition.

keyword arguments:

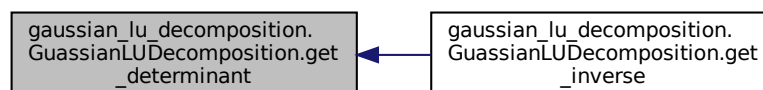
Parameters

<i>L</i>	The lower triangular matrix of the system.
<i>U</i>	The upper triangular matrix of the system.

Returns

float64

Here is the caller graph for this function:



5.7.1.4 get_inverse()

```
def gaussian_lu_decomposition.GuassianLUDecomposition.get_inverse (
    self,
    L,
    U )
```

Returns the inverse of a given matrix by means of LU decomposition.

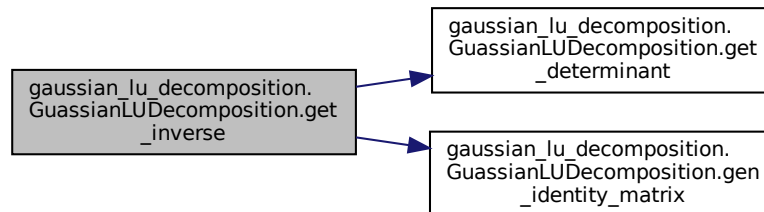
Parameters

<i>L</i>	The lower triangular matrix of the system.
<i>U</i>	The upper triangular matrix of the system.

Returns

float64[:,:]

Here is the call graph for this function:

**5.7.1.5 get_solution()**

```
def gaussian_lu_decomposition.GaussianLUDecomposition.get_solution (
    self,
    L,
    U,
    b )
```

Solves a LU system.

Parameters

<i>L</i>	The lower triangular matrix of the system.
<i>U</i>	The upper triangular matrix of the system.
<i>b</i>	Linearly independent vector.

Returns

float64[:,:]

5.7.1.6 start()

```
def gaussian_lu_decomposition.GaussianLUDecomposition.start (
    self,
    A_matrix )
```

Decomposes *A_matrix* into two matrices *L* and *U*.

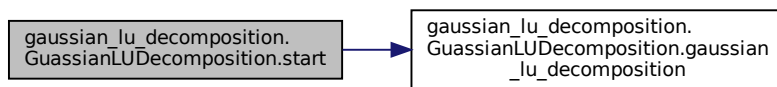
Parameters

<code>A_matrix</code>	Coefficient matrix.
-----------------------	---------------------

Returns

`float64[:,:], float64[:,:]`

Here is the call graph for this function:



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

- `lu_decomposition/gaussian_lu_decomposition.py`

5.8 jacobi_parallel_chunks.JacobiParallel Class Reference

Public Member Functions

- def `jacobi` (`A`, `b`, `x_current`, `x_next`, `rows`, `cols`, `first_row_block`, `rel`)
Performs jacobi for every thread in matrix A boundaries.
- def `get_error` (`x_current`, `x_next`, `x_error`, `rows`)
- def `start` (`self`, `A`, `b`, `x_current`, `first_row_block`, `rel=1`)

Static Public Attributes

- `target`
- `nopython`

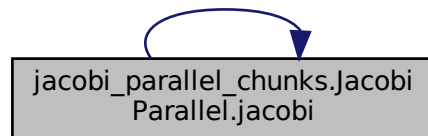
5.8.1 Member Function Documentation

5.8.1.1 jacobi()

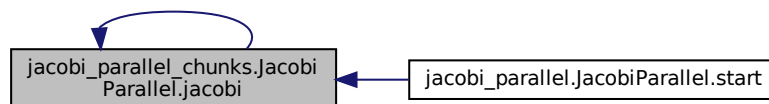
```
def jacobi_parallel_chunks.JacobiParallel.jacobi (
    A,
    b,
    x_current,
    x_next,
    rows,
    cols,
    first_row_block,
    rel )
```

Performs jacobi for every thread in matrix A boundaries.

Key arguments: A – [Matrix](#) extracted from the coefficient matrix A. b – Vector extracted from Linearly independent vector b. x_current – Current answer's approximation. x_next – vector in which to store new answer. rows – Number of rows read (i.e. number of rows in the block). cols – Number of columns from the original matrix. first_row_block – Integer indicating the first row of the block by using an index from the coefficient matrix A (i.e. What is the correspondence between the first block's row and A). rel – Relaxation coefficient. Here is the call graph for this function:



Here is the caller graph for this function:



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

- block_operations/jacobi_parallel_chunks.py

5.9 jacobi_parallel.JacobiParallel Class Reference

Public Member Functions

- def [jacobi](#) (A, b, x_current, x_next, n, rel)
Runs jacobi for every thread in matrix A boundaries.
- def [get_error](#) (x_current, x_next, x_error, rows)
Calculates jacobi's maximum error.
- def [start](#) (self, A, b, niter, tol, rel=1)
Launches parallel jacobi solver for a SLAE and returns its answer.

Static Public Attributes

- **target**
- **nopython**

5.9.1 Member Function Documentation

5.9.1.1 get_error()

```
def jacobi_parallel.JacobiParallel.get_error (
    x_current,
    x_next,
    x_error,
    rows )
```

Calculates jacobi's maximum error.

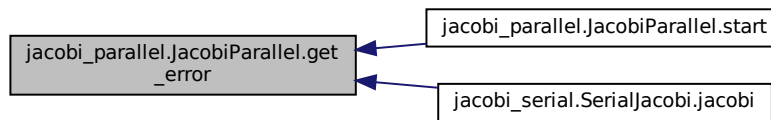
Parameters

<i>x_current</i>	Pointer to list representing current approximation for vector x in a system $Ax = b$.
<i>x_next</i>	Pointer to list representing new approximation for vector x in a system $Ax = b$.
<i>x_error</i>	Pointer to list in which an error for each approximation will be stored.
<i>rows</i>	Coefficient matrix A number of rows.

Returns

None

Here is the caller graph for this function:

**5.9.1.2 jacobi()**

```
def jacobi_parallel.JacobiParallel.jacobi (
    A,
    b,
    x_current,
    x_next,
    n,
    rel )
```

Runs jacobi for every thread in matrix A boundaries.

Parameters

<i>A</i>	Coefficient matrix.
<i>b</i>	Linearly independent vector.
<i>x_current</i>	Current answer's approximation.
<i>x_next</i>	vector in which to store new answer.
<i>n</i>	Coefficient matrix' size.
<i>rel</i>	Relaxation coefficient.

Returns

None

Here is the caller graph for this function:



5.9.1.3 start()

```
def jacobi_parallel.JacobiParallel.start (
    self,
    A,
    b,
    niter,
    tol,
    rel = 1 )
```

Launches parallel jacobi solver for a SLAE and returns its answer.

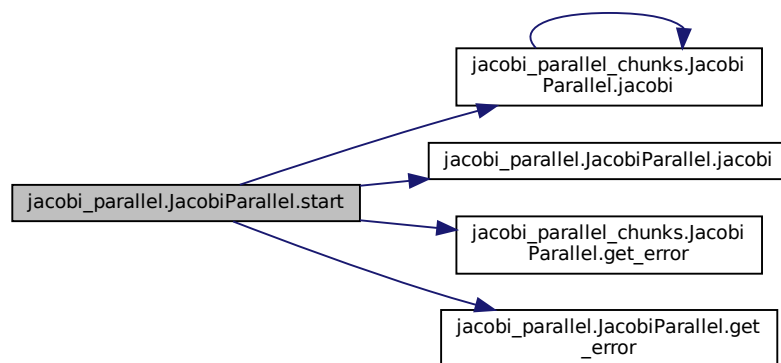
Parameters

<i>A</i>	Coefficient matrix of a SLAE.
<i>b</i>	Linearly independent vector of a SLAE.
<i>niter</i>	Maximum number of iterations before jacobi stops.
<i>tol</i>	Maximum error reached by jacobi when solving the system
<i>rel</i>	Relaxation coefficient.

Returns

float64[:]

Here is the call graph for this function:



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

- `jacobi/jacobi_parallel.py`

5.10 jacobi_tab.JacobiTab Class Reference

Public Member Functions

- `def __init__(self)`

- def **get_tab** (self)
- def **load_matrix** (self, widget, data=None)
- def **load_vector** (self, widget, data=None)
- def **jacobi_parallel** (self, widget, data=None)
- def **jacobi_serial** (self, widget, data=None)
- def **save** (self, widget, data=None)

Public Attributes

- **jacobiParallel**
- **jacobiSerial**
- **niter_entry**
- **A_matrix**
- **b_vector**
- **x_vector**
- **error_entry**
- **rel_entry**

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

- jacobi/jacobi_tab.py

5.11 lu_decomposition_tab.LUDecompositionTab Class Reference

Public Member Functions

- def **__init__** (self)
- def **get_tab** (self)
- def **load_matrix** (self, widget, data=None)
- def **load_vector** (self, widget, data=None)
- def **lu_decomposition** (self, widget, data=None)
- def **serial_lu** (self, widget, data=None)
- def **substitution** (self, widget, data=None)
- def **get_determinant** (self, widget, data=None)
- def **get_inverse** (self, widget, data=None)
- def **save_lu** (self, widget, data=None)
- def **save_inverse** (self, widget, data=None)
- def **save_x** (self, widget, data=None)

Public Attributes

- **gaussian_lu_decomposition**
- **serial_lu_decomposition**
- **A_matrix**
- **b_vector**
- **L_matrix**
- **U_matrix**
- **inverse**
- **x_vector**
- **U**

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

- lu_decomposition/lu_decomposition_tab.py

5.12 matrix_generator.MatrixGenerator Class Reference

Static Public Member Functions

- def [gen_vector](#) (size)
Creates a random vector given a size.
- def [gen_dominant](#) (size)
Creates a diagonally dominant matrix given a size.
- def [gen_symmetric_matrix](#) (size)
Creates a symmetric matrix given a size.
- def [gen_random_matrix](#) (size)
Creates a random matrix given a size.
- def [gen_band_matrix](#) (size, k1, k2)
Creates a band matrix given a size.
- def [gen_identity_matrix](#) (size)
Creates an identity matrix given a size.
- def [gen_diagonal_matrix](#) (size)
Creates a diagonal matrix given a size.
- def [gen_scalar_matrix](#) (size)
Creates a scalar matrix given a size.
- def [gen_antisymmetric_matrix](#) (size)
Creates an anti-symmetric matrix given a size.
- def [gen_lower_matrix](#) (size)
Creates a lower triangular matrix given a size.
- def [gen_upper_matrix](#) (size)
Creates an upper triangular matrix given a size.

5.12.1 Member Function Documentation

5.12.1.1 gen_antisymmetric_matrix()

```
def matrix_generator.MatrixGenerator.gen_antisymmetric_matrix (
    size ) [static]
```

Creates an anti-symmetric matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:,]

5.12.1.2 `gen_band_matrix()`

```
def matrix_generator.MatrixGenerator.gen_band_matrix (
    size,
    k1,
    k2 ) [static]
```

Creates a band matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
<i>k1</i>	Number of diagonals with non-zero elements below the main diagonal (Inclusive).
<i>k2</i>	Number of diagonals with non-zero elements above the main diagonal (Inclusive).

Returns

`float128[:,:], float128[:,], float128[:,]`

5.12.1.3 `gen_diagonal_matrix()`

```
def matrix_generator.MatrixGenerator.gen_diagonal_matrix (
    size ) [static]
```

Creates a diagonal matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

`float128[:,:], float128[:,], float128[:,]`

5.12.1.4 `gen_dominant()`

```
def matrix_generator.MatrixGenerator.gen_dominant (
    size ) [static]
```

Creates a diagonally dominant matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:,]

5.12.1.5 gen_identity_matrix()

```
def matrix_generator.MatrixGenerator.gen_identity_matrix (
    size ) [static]
```

Creates an identity matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:,]

5.12.1.6 gen_lower_matrix()

```
def matrix_generator.MatrixGenerator.gen_lower_matrix (
    size ) [static]
```

Creates a lower triangular matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:,]

5.12.1.7 gen_random_matrix()

```
def matrix_generator.MatrixGenerator.gen_random_matrix (
    size ) [static]
```

Creates a random matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:]

5.12.1.8 gen_scalar_matrix()

```
def matrix_generator.MatrixGenerator.gen_scalar_matrix (  
    size ) [static]
```

Creates a scalar matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:]

5.12.1.9 gen_symmetric_matrix()

```
def matrix_generator.MatrixGenerator.gen_symmetric_matrix (  
    size ) [static]
```

Creates a symmetric matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:]

5.12.1.10 gen_upper_matrix()

```
def matrix_generator.MatrixGenerator.gen_upper_matrix (  
    size ) [static]
```

Creates an upper triangular matrix given a size.

Parameters

<i>size</i>	Number of rows and columns that the matrix will have.
-------------	---

Returns

float128[:,:], float128[:,], float128[:,]

5.12.1.11 gen_vector()

```
def matrix_generator.MatrixGenerator.gen_vector (
    size ) [static]
```

Creates a random vector given a size.

Parameters

<i>size</i>	Length of the vector that will be created.
-------------	--

Returns

float128[:,]

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

- matrixGenerator/matrix_generator.py

5.13 matrix_generator_tab.MatrixGeneratorTab Class Reference**Public Member Functions**

- def **__init__** (self)
- def **get_tab** (self)
- def **set_generator** (self, button, name)
- def **gen_matrix** (self, widget, data=None)

Public Attributes

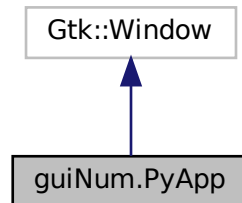
- **matrix_filename_entry**
- **vector_filename_entry**
- **length_entry**
- **selected_generator**

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

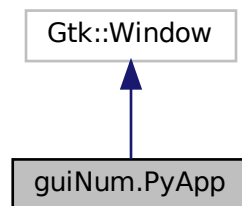
- matrixGenerator/matrix_generator_tab.py

5.14 guiNum.PyApp Class Reference

Inheritance diagram for guiNum.PyApp:



Collaboration diagram for guiNum.PyApp:



Public Member Functions

- `def __init__(self)`

Public Attributes

- `sparse_matrix_tab`
- `matrix_generator_tab`
- `jacobi_tab`
- `gauss_jordan_tab`
- `gaussian_elimination_tab`
- `lu_decomposition_tab`
- `blocks_tab`

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

- `guiNum.py`

5.15 serial_gaussian_elimination.SerialGaussianElimination Class Reference

Public Member Functions

- def [elimination](#) (self, A, b)
Takes a system of linear equations represented by a matrix and a vector and returns the answer applying [Gaussian elimination](#) method.
- def [partial_pivot](#) (self, A, b, k)
Applies the partial pivot strategy to a system of linear equations.

5.15.1 Member Function Documentation

5.15.1.1 [elimination\(\)](#)

```
def serial_gaussian_elimination.SerialGaussianElimination.elimination (
    self,
    A,
    b )
```

Takes a system of linear equations represented by a matrix and a vector and returns the answer applying [Gaussian elimination](#) method.

Parameters

<i>A</i>	The coefficient matrix of the system.
<i>b</i>	The linearly independent vector.

Returns

float128[:]

Here is the call graph for this function:



5.15.1.2 [partial_pivot\(\)](#)

```
def serial_gaussian_elimination.SerialGaussianElimination.partial_pivot (
    self,
```

A ,
 b ,
 k)

Applies the partial pivot strategy to a system of linear equations.

Parameters

A	The coefficient matrix of the system.
b	The linearly independent vector.
k	The current elimination stage.

Returns

float128[:,:], float128[:]

Here is the caller graph for this function:



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

- gaussian_elimination/serial_gaussian_elimination.py

5.16 jacobi_serial.SerialJacobi Class Reference

Public Member Functions

- def [multiply_matrix_vector](#) (self, A_matrix, b_vector)
Returns the dot product between a matrix and a vector.
- def [multiply_matrix_matrix](#) (self, matrix1, matrix2)
Returns the dot product between two matrices.
- def [get_D_and_U](#) (self, matrix)
Split a given matrix into two matrices D and U (lower and upper triangular matrices)
- def [get_inverse](#) (self, matrixD)
Returns the inverse of a LOWER TRIANGULAR MATRIX.
- def [sum_vectors](#) (self, vector1, vector2)
Takes two vector and sum them.
- def [get_error](#) (self, x_vector, xant_vector)
Returns the norm of two given vectors, which represents the error of the current method.
- def [relaxation](#) (self, x_vector, xant_vector, relaxation)
Applies the relaxation method to Jacobi.
- def [jacobi](#) (self, A_matrix, b_vector, max_iterations, tolerance, [relaxation](#)=1)
Applies Jacobi method to a system of linear equations and returns its answer (except if it was not found), number of iterations executed and the maximum error.

5.16.1 Member Function Documentation

5.16.1.1 `get_D_and_U()`

```
def jacobi_serial.SerialJacobi.get_D_and_U (
    self,
    matrix )
```

Split a given matrix into two matrices D and U (lower and upper triangular matrices)

Parameters

<i>matrix</i>	The matrix to be splitted.
---------------	----------------------------

Returns

float128[:,:],float128[:,:]

Here is the caller graph for this function:



5.16.1.2 `get_error()`

```
def jacobi_serial.SerialJacobi.get_error (
    self,
    x_vector,
    xant_vector )
```

Returns the norm of two given vectors, which represents the error of the current method.

Parameters

<i>x_vector</i>	The vector of the current stage of the method.
<i>xant_vector</i>	The vector of the previous stage of the method.

Returns

float128

Here is the caller graph for this function:

5.16.1.3 `get_inverse()`

```
def jacobi_serial.SerialJacobi.get_inverse (
    self,
    matrixD )
```

Returns the inverse of a LOWER TRIANGULAR MATRIX.

Parameters

<i>matrixD</i>	The matrix base to calculate the inverse.
----------------	---

Returns

float128[:,:]

Here is the caller graph for this function:

5.16.1.4 `jacobi()`

```
def jacobi_serial.SerialJacobi.jacobi (
    self,
```

```

A_matrix,
b_vector,
max_iterations,
tolerance,
relaxation = 1 )

```

Applies Jacobi method to a system of linear equations and returns its answer (except if it was not found), number of iterations executed and the maximum error.

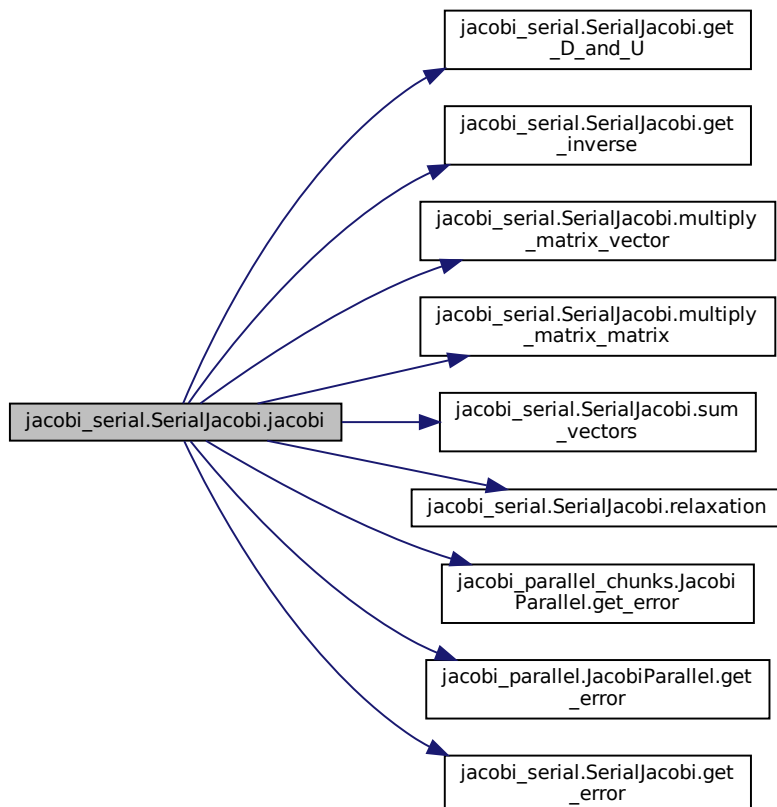
Parameters

<i>A_matrix</i>	The coefficient matrix of the system.
<i>b_vector</i>	The linearly independent vector.
<i>max_iterations</i>	Maximum number of iterations of the method.
<i>tolerance</i>	The tolerance of the method
<i>relaxation</i>	The number that will be used in the relaxation of the method.

Returns

float128[:] or None, int32, float128

Here is the call graph for this function:



5.16.1.5 multiply_matrix_matrix()

```
def jacobi_serial.SerialJacobi.multiply_matrix_matrix (
    self,
    matrix1,
    matrix2 )
```

Returns the dot product between two matrices.

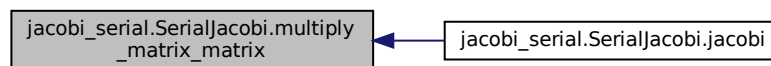
Parameters

<i>matrix1</i>	The first matrix to be multiplied.
<i>matrix2</i>	The second matrix to be multiplied.

Returns

float128[:]

Here is the caller graph for this function:



5.16.1.6 multiply_matrix_vector()

```
def jacobi_serial.SerialJacobi.multiply_matrix_vector (
    self,
    A_matrix,
    b_vector )
```

Returns the dot product between a matrix and a vector.

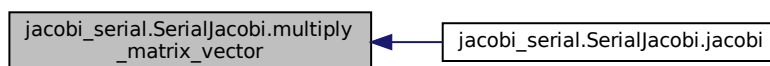
Parameters

<i>A_matrix</i>	The matrix to be multiplied.
<i>b_vector</i>	The vector to be multiplied.

Returns

float128[:]

Here is the caller graph for this function:

**5.16.1.7 relaxation()**

```
def jacobi_serial.SerialJacobi.relaxation (
    self,
    x_vector,
    xant_vector,
    relaxation )
```

Applies the relaxation method to Jacobi.

Parameters

<i>x_vector</i>	The vector of the current stage of the method.
<i>xant_vector</i>	The vector of the previous stage of the method.
<i>relaxation</i>	The number that will be used in the relaxation of the method.

Returns

float128[:]

Here is the caller graph for this function:



5.16.1.8 sum_vectors()

```
def jacobi_serial.SerialJacobi.sum_vectors (
    self,
    vector1,
    vector2 )
```

Takes two vector and sum them.

Parameters

<i>vector1</i>	The first vector to be added.
<i>vector2</i>	The second vector to be added.

Returns

float128[:]

Here is the caller graph for this function:



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

- `jacobi/jacobi_serial.py`

5.17 serial_decomposition_LU.SerialLUDecomposition Class Reference

Public Member Functions

- def [decomposition_LU](#) (self, A)
Splits a given matrix into two matrices (lower and upper triangular matrices).
- def [solve_system](#) (self, L, U, b)
Solves a LU system.

5.17.1 Member Function Documentation

5.17.1.1 decomposition_LU()

```
def serial_decomposition_LU.SerialLUDecomposition.decomposition_LU (
    self,
    A )
```

Splits a given matrix into two matrices (lower and upper triangular matrices).

It is based on multiplication of matrices.

Parameters

<i>A</i>	The coefficient matrix to be splitted.
----------	--

Returns

float128[:,:], float128[:,:]

5.17.1.2 solve_system()

```
def serial_decomposition_LU.SerialLUdecomposition.solve_system (
    self,
    L,
    U,
    b )
```

Solves a LU system.

Parameters

<i>L</i>	The lower triangular matrix of the system.
<i>U</i>	The upper triangular matrix of the system.
<i>b</i>	Linearly independent vector.

Returns

float128[:]

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

- lu_decomposition/serial_decomposition_LU.py

5.18 sparse_matrix.SparseMatrix Class Reference**Public Member Functions**

- def [create_sparse_matrix](#) (self, filename, matrix_length, density)
Creates a sparse matrix with CSR format (four arrays)
- def [load_sparse_matrix](#) (self, filename)
Takes a file and get the values array of it.
- def [multiply](#) (self, filename_matrix, vector)
Takes a file with a sparse matrix in CSR format and multiply it with a vector.

Static Public Member Functions

- def [gen_vector](#) (size)
Creates a random vector given a size.

5.18.1 Member Function Documentation

5.18.1.1 create_sparse_matrix()

```
def sparse_matrix.SparseMatrix.create_sparse_matrix (
    self,
    filename,
    matrix_length,
    density )
```

Creates a sparse matrix with CSR format (four arrays)

Parameters

<i>filename</i>	The file name where will be stored the final result.
<i>matrix_length</i>	The length of the matrix.
<i>density</i>	percentage of non-zeros elements

Returns

float128[:,:], str, float128[:,], float128[:,]

5.18.1.2 gen_vector()

```
def sparse_matrix.SparseMatrix.gen_vector (
    size ) [static]
```

Creates a random vector given a size.

Parameters

<i>size</i>	Length of the vector that will be created.
-------------	--

Returns

float128[:,]

5.18.1.3 load_sparse_matrix()

```
def sparse_matrix.SparseMatrix.load_sparse_matrix (
    self,
    filename )
```

Takes a file and get the values array of it.

Parameters

<i>filename</i>	The file name where arrayes are stored.
-----------------	---

Returns

None

5.18.1.4 multiply()

```
def sparse_matrix.SparseMatrix.multiply (
    self,
    filename_matrix,
    vector )
```

Takes a file with a sparse matrix in CSR format and multiply it with a vector.

Parameters

<i>filename_matrix</i>	The filename where the CSR matrix is located.
<i>vector</i>	The vector to multiply with the matrix

Returns

128[:]

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

- sparseMatrices/sparse_matrix.py

5.19 sm_testCSR.SparseMatrix Class Reference

Public Member Functions

- def **create_sparse_matrix** (self, filename, matrix_length, density)

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

- sparseMatrices/sm_testCSR.py

5.20 sm_test.SparseMatrix Class Reference

Public Member Functions

- def **create_sparse_matrix** (self, filename, matrix_length, density)

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

- sparseMatrices/sm_test.py

5.21 sparse_matrix_tab.SparseMatrixTab Class Reference

Public Member Functions

- def **__init__** (self)
- def **get_sparse_tab** (self)
- def **create_sparse_matrix** (self, widget, data=None)
- def **multiply** (self, widget, data=None)
- def **save_result** (self, widget, data=None)

Public Attributes

- **sparseMatrix**
- **filename_entry**
- **matrix_length_entry**
- **matrix_density_entry**
- **filename**
- **res**

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

- sparseMatrices/sparse_matrix_tab.py

Index

block_operations_tab.BlockTab, 9

create_sparse_matrix
 sparse_matrix::SparseMatrix, 41

decomposition_LU
 serial_decomposition_LU::SerialLUDecomposition, 39

elimination
 serial_gauss_jordan::GaussJordanSerial, 14
 serial_gaussian_elimination::SerialGaussianElimination, 32

Gauss, 7

gauss_jordan
 gauss_jordan::GaussJordan, 12

gauss_jordan.GaussJordan, 12

gauss_jordan::GaussJordan
 gauss_jordan, 12
 normalize, 12
 start, 13

gauss_jordan_tab.GaussJordanTab, 15

Gaussian, 7

gaussian_elimination
 gaussian_elimination::GaussianElimination, 10

gaussian_elimination.GaussianElimination, 9

gaussian_elimination::GaussianElimination
 gaussian_elimination, 10
 start, 10

gaussian_elimination_tab.GaussianEliminationTab, 11

gaussian_lu_decomposition
 gaussian_lu_decomposition::GuassianLUDecomposition, 15

gaussian_lu_decomposition.GuassianLUDecomposition, 15

gaussian_lu_decomposition::GuassianLUDecomposition
 gaussian_lu_decomposition, 15
 gen_identity_matrix, 16
 get_determinant, 17
 get_inverse, 17
 get_solution, 18
 start, 18

gen_antisymmetric_matrix
 matrix_generator::MatrixGenerator, 25

gen_band_matrix
 matrix_generator::MatrixGenerator, 25

gen_diagonal_matrix
 matrix_generator::MatrixGenerator, 26

gen_dominant
 matrix_generator::MatrixGenerator, 26

gen_identity_matrix
 gaussian_lu_decomposition::GuassianLUDecomposition, 16
 matrix_generator::MatrixGenerator, 27

gen_lower_matrix
 matrix_generator::MatrixGenerator, 27

gen_random_matrix
 matrix_generator::MatrixGenerator, 27

gen_scalar_matrix
 matrix_generator::MatrixGenerator, 28

gen_symmetric_matrix
 matrix_generator::MatrixGenerator, 28

gen_upper_matrix
 matrix_generator::MatrixGenerator, 28

gen_vector
 matrix_generator::MatrixGenerator, 30
 sparse_matrix::SparseMatrix, 41

get_D_and_U
 jacobi_serial::SerialJacobi, 34

get_determinant
 gaussian_lu_decomposition::GuassianLUDecomposition, 17

get_error
 jacobi_parallel::JacobiParallel, 21
 jacobi_serial::SerialJacobi, 34

get_inverse
 gaussian_lu_decomposition::GuassianLUDecomposition, 17
 jacobi_serial::SerialJacobi, 35

get_solution
 gaussian_lu_decomposition::GuassianLUDecomposition, 18

guiNum.PyApp, 31

jacobi, 7
 jacobi_parallel::JacobiParallel, 22
 jacobi_parallel_chunks::JacobiParallel, 19
 jacobi_serial::SerialJacobi, 35

jacobi_parallel.JacobiParallel, 21

jacobi_parallel::JacobiParallel
 get_error, 21
 jacobi, 22
 start, 22

jacobi_parallel_chunks.JacobiParallel, 19

jacobi_parallel_chunks::JacobiParallel
 jacobi, 19

jacobi_serial.SerialJacobi, 33

jacobi_serial::SerialJacobi
 get_D_and_U, 34

- get_error, [34](#)
 - get_inverse, [35](#)
 - jacobi, [35](#)
 - multiply_matrix_matrix, [36](#)
 - multiply_matrix_vector, [37](#)
 - relaxation, [38](#)
 - sum_vectors, [38](#)
- jacobi_tab.JacobiTab, [23](#)
- load_sparse_matrix
 - sparse_matrix::SparseMatrix, [41](#)
- lu_decomposition, [7](#)
- lu_decomposition_tab.LUDecompositionTab, [24](#)
- Matrix, [8](#)
- matrix_generator.MatrixGenerator, [25](#)
- matrix_generator::MatrixGenerator
 - gen_antisymmetric_matrix, [25](#)
 - gen_band_matrix, [25](#)
 - gen_diagonal_matrix, [26](#)
 - gen_dominant, [26](#)
 - gen_identity_matrix, [27](#)
 - gen_lower_matrix, [27](#)
 - gen_random_matrix, [27](#)
 - gen_scalar_matrix, [28](#)
 - gen_symmetric_matrix, [28](#)
 - gen_upper_matrix, [28](#)
 - gen_vector, [30](#)
- matrix_generator_tab.MatrixGeneratorTab, [30](#)
- multiply
 - sparse_matrix::SparseMatrix, [42](#)
- multiply_matrix_matrix
 - jacobi_serial::SerialJacobi, [36](#)
- multiply_matrix_vector
 - jacobi_serial::SerialJacobi, [37](#)
- normalize
 - gauss_jordan::GaussJordan, [12](#)
- partial_pivot
 - serial_gaussian_elimination::SerialGaussianElimination, [32](#)
- relaxation
 - jacobi_serial::SerialJacobi, [38](#)
- serial_decomposition_LU.SerialLUDecomposition, [39](#)
- serial_decomposition_LU::SerialLUDecomposition
 - decomposition_LU, [39](#)
 - solve_system, [40](#)
- serial_gauss_jordan.GaussJordanSerial, [14](#)
- serial_gauss_jordan::GaussJordanSerial
 - elimination, [14](#)
- serial_gaussian_elimination.SerialGaussianElimination, [32](#)
- serial_gaussian_elimination::SerialGaussianElimination
 - elimination, [32](#)
 - partial_pivot, [32](#)
- sm_test.SparseMatrix, [43](#)
- sm_testCSR.SparseMatrix, [42](#)
- solve_system
 - serial_decomposition_LU::SerialLUDecomposition, [40](#)
- sparse_matrix.SparseMatrix, [40](#)
- sparse_matrix::SparseMatrix
 - create_sparse_matrix, [41](#)
 - gen_vector, [41](#)
 - load_sparse_matrix, [41](#)
 - multiply, [42](#)
- sparse_matrix_tab.SparseMatrixTab, [43](#)
- sparseMatrices, [8](#)
- start
 - gauss_jordan::GaussJordan, [13](#)
 - gaussian_elimination::GaussianElimination, [10](#)
 - gaussian_lu_decomposition::GaussianLUDecomposition, [18](#)
 - jacobi_parallel::JacobiParallel, [22](#)
- sum_vectors
 - jacobi_serial::SerialJacobi, [38](#)