

# GPU solvers for SLAE

1.0

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	BlockOperations Namespace Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.2	GaussianElimination Namespace Reference . . . . .	7
4.2.1	Detailed Description . . . . .	7
4.3	GaussJordan Namespace Reference . . . . .	7
4.3.1	Detailed Description . . . . .	7
4.4	Jacobi Namespace Reference . . . . .	8
4.4.1	Detailed Description . . . . .	8
4.5	LuDecomposition Namespace Reference . . . . .	8
4.5.1	Detailed Description . . . . .	8
4.6	MatrixGenerator Namespace Reference . . . . .	8
4.6.1	Detailed Description . . . . .	8
4.7	SparseMatrices Namespace Reference . . . . .	8
4.7.1	Detailed Description . . . . .	8

<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	<a href="#">block_operations_tab.BlockTab Class Reference</a>	9
5.2	<a href="#">gaussian_elimination.GaussianElimination Class Reference</a>	9
5.2.1	Member Function Documentation	10
5.2.1.1	<a href="#">gaussian_elimination(Ab, size, i)</a>	10
5.2.1.2	<a href="#">start(self, A_matrix, b_matrix)</a>	10
5.3	<a href="#">gaussian_elimination_tab.GaussianEliminationTab Class Reference</a>	11
5.4	<a href="#">gauss_jordan.GaussJordan Class Reference</a>	12
5.4.1	Member Function Documentation	12
5.4.1.1	<a href="#">gauss_jordan(A, size, i)</a>	12
5.4.1.2	<a href="#">normalize(A, size)</a>	12
5.4.1.3	<a href="#">start(self, A_matrix, b_vector)</a>	13
5.5	<a href="#">serial_gauss_jordan.GaussJordanSerial Class Reference</a>	13
5.5.1	Member Function Documentation	14
5.5.1.1	<a href="#">elimination(self, A, b)</a>	14
5.6	<a href="#">gauss_jordan_tab.GaussJordanTab Class Reference</a>	14
5.7	<a href="#">gaussian_lu_decomposition.GaussianLUdecomposition Class Reference</a>	15
5.7.1	Member Function Documentation	15
5.7.1.1	<a href="#">gaussian_lu_decomposition(A, L, size, i)</a>	15
5.7.1.2	<a href="#">gen_identity_matrix(self, size)</a>	15
5.7.1.3	<a href="#">get_determinant(self, L, U)</a>	16
5.7.1.4	<a href="#">get_inverse(self, L, U)</a>	16
5.7.1.5	<a href="#">get_solution(self, L, U, b)</a>	17
5.7.1.6	<a href="#">start(self, A_matrix)</a>	17
5.8	<a href="#">jacobi_parallel_chunks.JacobiParallel Class Reference</a>	18
5.8.1	Member Function Documentation	18
5.8.1.1	<a href="#">get_error(x_current, x_next, x_error, rows)</a>	18
5.8.1.2	<a href="#">jacobi(A, b, x_current, x_next, rows, cols, first_row_block, rel)</a>	19
5.8.1.3	<a href="#">start(self, A, b, x_current, first_row_block, rel=1)</a>	20
5.9	<a href="#">jacobi_parallel.JacobiParallel Class Reference</a>	20

5.9.1	Member Function Documentation	20
5.9.1.1	get_error(x_current, x_next, x_error, rows)	20
5.9.1.2	jacobi(A, b, x_current, x_next, n, rel)	21
5.9.1.3	start(self, A, b, niter, tol, rel=1)	22
5.10	jacobi_tab.JacobiTab Class Reference	22
5.11	lu_decomposition_tab.LUDecompositionTab Class Reference	23
5.12	matrix_generator.MatrixGenerator Class Reference	24
5.12.1	Member Function Documentation	24
5.12.1.1	gen_antisymmetric_matrix(size)	24
5.12.1.2	gen_band_matrix(size, k1, k2)	24
5.12.1.3	gen_diagonal_matrix(size)	25
5.12.1.4	gen_dominant(size)	25
5.12.1.5	gen_identity_matrix(size)	25
5.12.1.6	gen_lower_matrix(size)	25
5.12.1.7	gen_random_matrix(size)	26
5.12.1.8	gen_scalar_matrix(size)	26
5.12.1.9	gen_symmetric_matrix(size)	26
5.12.1.10	gen_upper_matrix(size)	27
5.12.1.11	gen_vector(size)	27
5.13	matrix_generator_tab.MatrixGeneratorTab Class Reference	27
5.14	guiNum.PyApp Class Reference	28
5.15	serial_gaussian_elimination.SerialGaussianElimination Class Reference	29
5.15.1	Member Function Documentation	29
5.15.1.1	elimination(self, A, b)	29
5.15.1.2	partial_pivot(self, A, b, k)	29
5.16	jacobi_serial.SerialJacobi Class Reference	30
5.16.1	Member Function Documentation	30
5.16.1.1	get_D_and_U(self, matrix)	30
5.16.1.2	get_error(self, x_vector, xant_vector)	31
5.16.1.3	get_inverse(self, matrixD)	31

5.16.1.4	<a href="#">jacobi(self, A_matrix, b_vector, max_iterations, tolerance, relaxation=1)</a>	32
5.16.1.5	<a href="#">multiply_matrix_matrix(self, matrix1, matrix2)</a>	33
5.16.1.6	<a href="#">multiply_matrix_vector(self, A_matrix, b_vector)</a>	34
5.16.1.7	<a href="#">relaxation(self, x_vector, xant_vector, relaxation)</a>	34
5.16.1.8	<a href="#">sum_vectors(self, vector1, vector2)</a>	35
5.17	<a href="#">serial_decomposition_LU.SerialLUDecomposition Class Reference</a>	35
5.17.1	<a href="#">Member Function Documentation</a>	36
5.17.1.1	<a href="#">decomposition_LU(self, A)</a>	36
5.17.1.2	<a href="#">solve_system(self, L, U, b)</a>	36
5.18	<a href="#">sparse_matrix.SparseMatrix Class Reference</a>	36
5.18.1	<a href="#">Member Function Documentation</a>	37
5.18.1.1	<a href="#">create_sparse_matrix(self, filename, matrix_length, density)</a>	37
5.18.1.2	<a href="#">gen_vector(size)</a>	37
5.18.1.3	<a href="#">load_sparse_matrix(self, filename)</a>	37
5.18.1.4	<a href="#">multiply(self, filename_matrix, vector)</a>	38
5.19	<a href="#">sm_testCSR.SparseMatrix Class Reference</a>	38
5.20	<a href="#">sm_test.SparseMatrix Class Reference</a>	38
5.21	<a href="#">sparse_matrix_tab.SparseMatrixTab Class Reference</a>	38
<b>Index</b>		<b>41</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">BlockOperations</a>	Provides tools to solve large systems of linear algebraic equations by loading submatrices from a coefficient matrices . . . . .	7
<a href="#">GaussianElimination</a>	Solve a system of linear algebraic equations by using the Gaussian Elimination method . . . .	7
<a href="#">GaussJordan</a>	Solve a system of linear algebraic equations by using the Gauss Jordan Elimination method . .	7
<a href="#">Jacobi</a>	Solve a system of linear algebraic equations by using the <a href="#">Jacobi</a> iterative method . . . . .	8
<a href="#">LuDecomposition</a>	Decomposes a matrix A into two matrices L and U . . . . .	8
<a href="#">MatrixGenerator</a>	Generate different types of matrices . . . . .	8
<a href="#">SparseMatrices</a>	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 . . . . .	13
gauss_jordan_tab.GaussJordanTab . . . . .	14
gaussian_lu_decomposition.GaussianLUdecomposition . . . . .	15
jacobi_parallel_chunks.JacobiParallel . . . . .	18
jacobi_parallel.JacobiParallel . . . . .	20
jacobi_tab.JacobiTab . . . . .	22
lu_decomposition_tab.LUdecompositionTab . . . . .	23
matrix_generator.MatrixGenerator . . . . .	24
matrix_generator_tab.MatrixGeneratorTab . . . . .	27
serial_gaussian_elimination.SerialGaussianElimination . . . . .	29
jacobi_serial.SerialJacobi . . . . .	30
serial_decomposition_LU.SerialLUdecomposition . . . . .	35
sparse_matrix.SparseMatrix . . . . .	36
sm_testCSR.SparseMatrix . . . . .	38
sm_test.SparseMatrix . . . . .	38
sparse_matrix_tab.SparseMatrixTab . . . . .	38
Window	
guiNum.PyApp . . . . .	28



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">block_operations_tab.BlockTab</a>	9
<a href="#">gaussian_elimination.GaussianElimination</a>	9
<a href="#">gaussian_elimination_tab.GaussianEliminationTab</a>	11
<a href="#">gauss_jordan.GaussJordan</a>	12
<a href="#">serial_gauss_jordan.GaussJordanSerial</a>	13
<a href="#">gauss_jordan_tab.GaussJordanTab</a>	14
<a href="#">gaussian_lu_decomposition.GaussianLUdecomposition</a>	15
<a href="#">jacobi_parallel_chunks.JacobiParallel</a>	18
<a href="#">jacobi_parallel.JacobiParallel</a>	20
<a href="#">jacobi_tab.JacobiTab</a>	22
<a href="#">lu_decomposition_tab.LUdecompositionTab</a>	23
<a href="#">matrix_generator.MatrixGenerator</a>	24
<a href="#">matrix_generator_tab.MatrixGeneratorTab</a>	27
<a href="#">guiNum.PyApp</a>	28
<a href="#">serial_gaussian_elimination.SerialGaussianElimination</a>	29
<a href="#">jacobi_serial.SerialJacobi</a>	30
<a href="#">serial_decomposition_LU.SerialLUdecomposition</a>	35
<a href="#">sparse_matrix.SparseMatrix</a>	36
<a href="#">sm_testCSR.SparseMatrix</a>	38
<a href="#">sm_test.SparseMatrix</a>	38
<a href="#">sparse_matrix_tab.SparseMatrixTab</a>	38



## Chapter 4

# Namespace Documentation

### 4.1 BlockOperations Namespace Reference

Provides tools to solve large systems of linear algebraic equations by loading submatrices from a coefficient matrices.

#### 4.1.1 Detailed Description

Provides tools to solve large systems of linear algebraic equations by loading submatrices from a coefficient matrices.

### 4.2 GaussianElimination Namespace Reference

Solve a system of linear algebraic equations by using the Gaussian Elimination method.

#### 4.2.1 Detailed Description

Solve a system of linear algebraic equations by using the Gaussian Elimination method.

### 4.3 GaussJordan Namespace Reference

Solve a system of linear algebraic equations by using the Gauss Jordan Elimination method.

#### 4.3.1 Detailed Description

Solve a system of linear algebraic equations by using the Gauss Jordan Elimination method.

## 4.4 Jacobi Namespace Reference

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

### 4.4.1 Detailed Description

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

## 4.5 LuDecomposition Namespace Reference

Decomposes a matrix A into two matrices L and U.

### 4.5.1 Detailed Description

Decomposes a matrix A into two matrices L and U.

## 4.6 MatrixGenerator Namespace Reference

Generate different types of matrices.

### 4.6.1 Detailed Description

Generate different types of matrices.

## 4.7 SparseMatrices Namespace Reference

Represents a matrix with CSR format.

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

Performs Gaussian elimination for each row of a column.

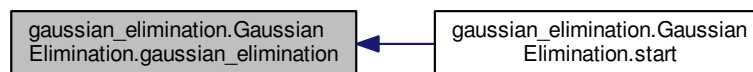
##### Parameters

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

Launches parallel Gaussian elimination for a SLAE and returns its answer.

##### Parameters

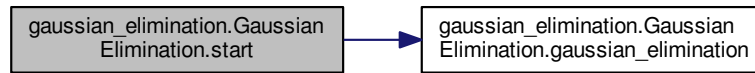
<i>A_matrix</i>	Coefficient matrix of a SLAE.
<i>b_matrix</i>	Linearly independent vector of a SLAE.



#### 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 `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 `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 def gauss\_jordan.GaussJordan.start ( self, A\_matrix, b\_vector )

Launches parallel Gauss Jordan elimination for a SLAE and returns its answer.

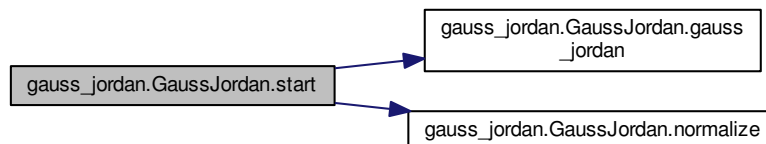
## Parameters

<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 `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.GuassianLUdecomposition 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 def gaussian\_lu\_decomposition.GuassianLUdecomposition.gaussian\_lu\_decomposition ( A, L, size, i )

Performs Gaussian LU elimination.

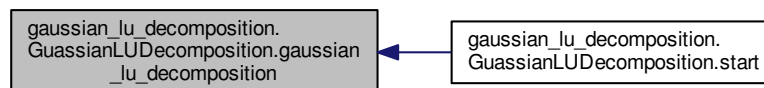
##### Parameters

<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 def gaussian\_lu\_decomposition.GuassianLUdecomposition.gen\_identity\_matrix ( self, size )

Creates an identity matrix given a size.

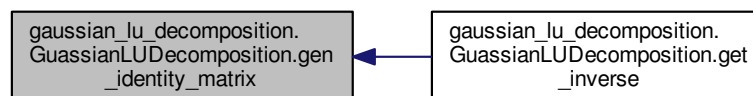
## Parameters

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

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

keyword arguments:

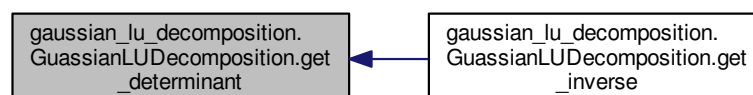
## Parameters

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

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

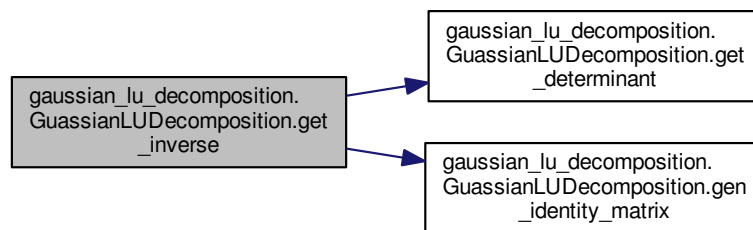
## Parameters

<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 `def gaussian_lu_decomposition.GuassianLUDecomposition.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 `def gaussian_lu_decomposition.GuassianLUDecomposition.start ( self, A_matrix )`

Decomposes *A\_matrix* into two matrices *L* and *U*.

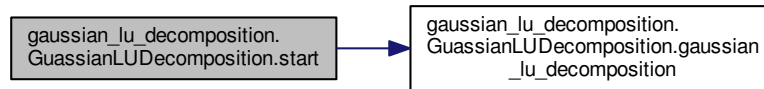
## Parameters

<i>A_matrix</i>	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)  
*Calculates jacobi's maximum error.*
- def [start](#) (self, A, b, x\_current, first\_row\_block, rel=1)  
*Launches parallel jacobi solver for a SLAE and returns its answer.*

### Static Public Attributes

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

#### 5.8.1 Member Function Documentation

##### 5.8.1.1 def jacobi\_parallel\_chunks.JacobiParallel.get\_error ( x\_current, x\_next, x\_error, rows )

Calculates jacobi's maximum error.

#### Parameters

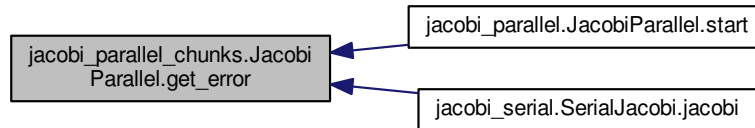
<i>x_current</i>	Pointer to list representing current approximation for vector x in a system $Ax = b$ .
<i>x_next</i>	Pointer to list representing new approximation for vector x in a system $Ax = b$ .
<i>x_error</i>	Pointer to list in which an error for each approximation will be stored.
<i>rows</i>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



## Returns

None

Here is the caller graph for this function:



5.8.1.2 `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.

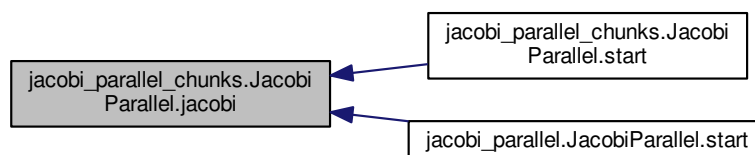
## Parameters

<i>A</i>	Matrix extracted from the coefficient matrix A.
<i>b</i>	Vector extracted from Linearly independent vector b.
<i>x_current</i>	Current answer's approximation.
<i>x_next</i>	vector in which to store new answer.
<i>rows</i>	Number of rows read (i.e. number of rows in the block).
<i>cols</i>	Number of columns from the original matrix.
<i>first_row_block</i>	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).
<i>rel</i>	Relaxation coefficient.

## Returns

None

Here is the caller graph for this function:



5.8.1.3 `def jacobi_parallel_chunks.JacobiParallel.start ( self, A, b, x_current, first_row_block, rel = 1 )`

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

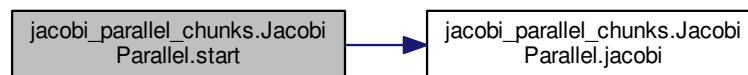
#### Parameters

<i>A</i>	Coefficient matrix of a SLAE.
<i>b</i>	Linearly independent vector of a SLAE.
<i>x_current</i>	Pointer to list representing current approximation for vector x in a system $Ax = b$ .
<i>first_row_block</i>	Absolute position of the block's row in the complete matrix.
<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:

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

Calculates jacobi's maximum error.

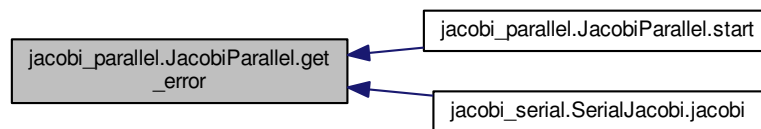
## Parameters

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

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

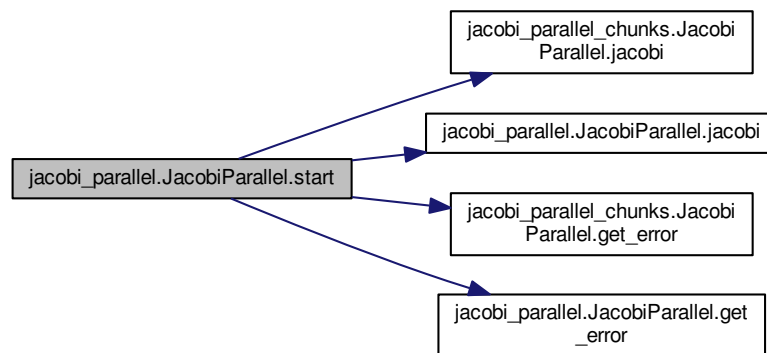
#### Parameters

<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 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 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 def matrix\_generator.MatrixGenerator.gen\_diagonal\_matrix ( size ) [static]

Creates a diagonal matrix given a size.

## Parameters

size	Number of rows and columns that the matrix will have.
------	---

## Returns

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

## 5.12.1.4 def matrix\_generator.MatrixGenerator.gen\_dominant ( size ) [static]

Creates a diagonally dominant matrix given a size.

## Parameters

size	Number of rows and columns that the matrix will have.
------	---

## Returns

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

## 5.12.1.5 def matrix\_generator.MatrixGenerator.gen\_identity\_matrix ( size ) [static]

Creates an identity matrix given a size.

## Parameters

size	Number of rows and columns that the matrix will have.
------	---

## Returns

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

## 5.12.1.6 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 `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 `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 `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 def matrix\_generator.MatrixGenerator.gen\_upper\_matrix ( size ) [static]

Creates an upper triangular matrix given a size.

## Parameters

size	Number of rows and columns that the matrix will have.
------	---

## Returns

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

## 5.12.1.11 def matrix\_generator.MatrixGenerator.gen\_vector ( size ) [static]

Creates a random vector given a size.

## Parameters

size	Length of the vector that will be created.
------	--

## Returns

float128[:]

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

- matrix\_generator/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

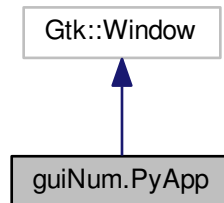
- **matrix\_filename\_entry**
- **vector\_filename\_entry**
- **length\_entry**
- **selected\_generator**

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

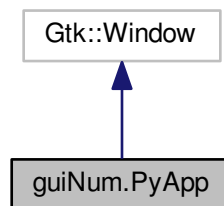
- matrix\_generator/matrix\_generator\_tab.py

## 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 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 def serial\_gaussian\_elimination.SerialGaussianElimination.partial\_pivot ( self, A, b, k )

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

##### Parameters

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

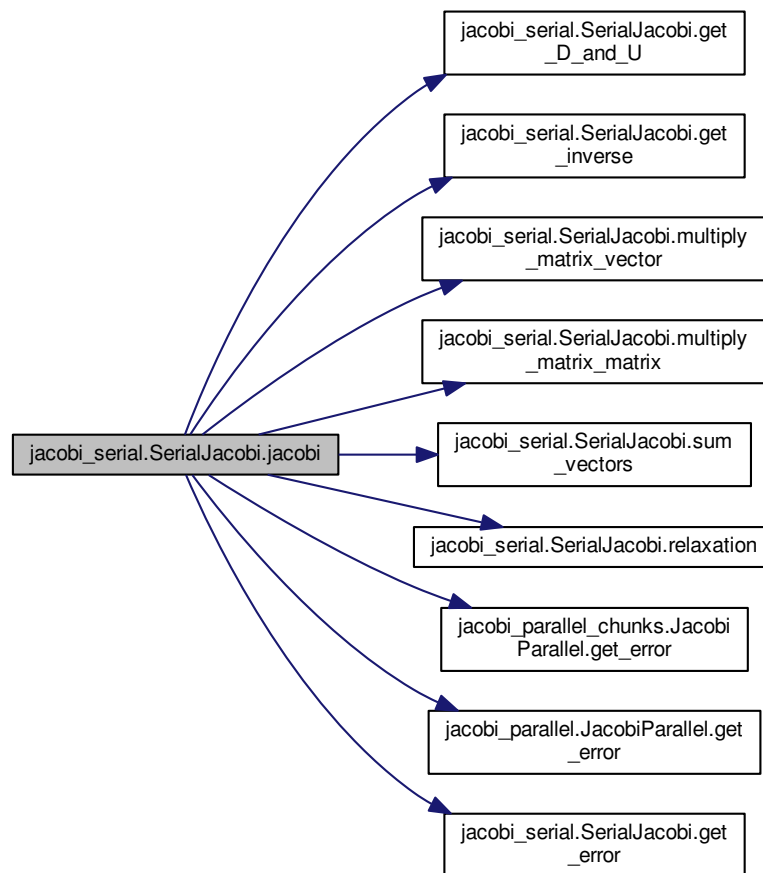
## Parameters

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

Returns the dot product between two matrices.

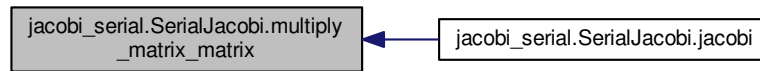
#### Parameters

<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 def jacobi\_serial.SerialJacobi.multiply\_matrix\_vector ( *self*, *A\_matrix*, *b\_vector* )

Returns the dot product between a matrix and a vector.

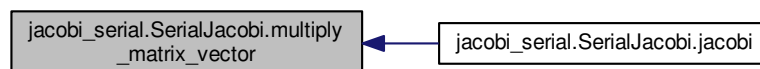
## Parameters

<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 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 `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 `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 `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 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 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 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 `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:

- `sparse_matrices/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:

- `sparse_matrices/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:

- `sparse_matrices/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:

- sparse\_matrices/sparse\_matrix\_tab.py



# Index

block\_operations\_tab.BlockTab, 9  
BlockOperations, 7

create\_sparse\_matrix  
    sparse\_matrix::SparseMatrix, 37

decomposition\_LU  
    serial\_decomposition\_LU::SerialLUDecomposition, 36

elimination  
    serial\_gauss\_jordan::GaussJordanSerial, 14  
    serial\_gaussian\_elimination::SerialGaussianElimination, 29

gauss\_jordan  
    gauss\_jordan::GaussJordan, 12  
gauss\_jordan.GaussJordan, 12  
gauss\_jordan::GaussJordan  
    gauss\_jordan, 12  
    normalize, 12  
    start, 13  
gauss\_jordan\_tab.GaussJordanTab, 14  
GaussJordan, 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, 15  
    get\_determinant, 16  
    get\_inverse, 16  
    get\_solution, 17  
    start, 17  
GaussianElimination, 7  
gen\_antisymmetric\_matrix  
    matrix\_generator::MatrixGenerator, 24  
gen\_band\_matrix  
    matrix\_generator::MatrixGenerator, 24  
gen\_diagonal\_matrix  
    matrix\_generator::MatrixGenerator, 25  
gen\_dominant  
    matrix\_generator::MatrixGenerator, 25  
gen\_identity\_matrix  
    gaussian\_lu\_decomposition::GuassianLUDecomposition, 15  
    matrix\_generator::MatrixGenerator, 25  
gen\_lower\_matrix  
    matrix\_generator::MatrixGenerator, 25  
gen\_random\_matrix  
    matrix\_generator::MatrixGenerator, 26  
gen\_scalar\_matrix  
    matrix\_generator::MatrixGenerator, 26  
gen\_symmetric\_matrix  
    matrix\_generator::MatrixGenerator, 26  
gen\_upper\_matrix  
    matrix\_generator::MatrixGenerator, 26  
gen\_vector  
    matrix\_generator::MatrixGenerator, 27  
    sparse\_matrix::SparseMatrix, 37  
get\_D\_and\_U  
    jacobi\_serial::SerialJacobi, 30  
get\_determinant  
    gaussian\_lu\_decomposition::GuassianLUDecomposition, 16  
get\_error  
    jacobi\_parallel::JacobiParallel, 20  
    jacobi\_parallel\_chunks::JacobiParallel, 18  
    jacobi\_serial::SerialJacobi, 31  
get\_inverse  
    gaussian\_lu\_decomposition::GuassianLUDecomposition, 16  
    jacobi\_serial::SerialJacobi, 31  
get\_solution  
    gaussian\_lu\_decomposition::GuassianLUDecomposition, 17  
guiNum.PyApp, 28

Jacobi, 8  
jacobi  
    jacobi\_parallel::JacobiParallel, 21  
    jacobi\_parallel\_chunks::JacobiParallel, 19  
    jacobi\_serial::SerialJacobi, 32  
jacobi\_parallel.JacobiParallel, 20  
jacobi\_parallel::JacobiParallel  
    get\_error, 20  
    jacobi, 21  
    start, 21  
jacobi\_parallel\_chunks.JacobiParallel, 18  
jacobi\_parallel\_chunks::JacobiParallel  
    get\_error, 18

- jacobi, 19
  - start, 19
- jacobi\_serial.SerialJacobi, 30
- jacobi\_serial::SerialJacobi
  - get\_D\_and\_U, 30
  - get\_error, 31
  - get\_inverse, 31
  - jacobi, 32
  - multiply\_matrix\_matrix, 33
  - multiply\_matrix\_vector, 34
  - relaxation, 34
  - sum\_vectors, 35
- jacobi\_tab.JacobiTab, 22
- load\_sparse\_matrix
  - sparse\_matrix::SparseMatrix, 37
- lu\_decomposition\_tab.LUDecompositionTab, 23
- LUDecomposition, 8
- matrix\_generator.MatrixGenerator, 24
- matrix\_generator::MatrixGenerator
  - gen\_antisymmetric\_matrix, 24
  - gen\_band\_matrix, 24
  - gen\_diagonal\_matrix, 25
  - gen\_dominant, 25
  - gen\_identity\_matrix, 25
  - gen\_lower\_matrix, 25
  - gen\_random\_matrix, 26
  - gen\_scalar\_matrix, 26
  - gen\_symmetric\_matrix, 26
  - gen\_upper\_matrix, 26
  - gen\_vector, 27
- matrix\_generator\_tab.MatrixGeneratorTab, 27
- MatrixGenerator, 8
- multiply
  - sparse\_matrix::SparseMatrix, 37
- multiply\_matrix\_matrix
  - jacobi\_serial::SerialJacobi, 33
- multiply\_matrix\_vector
  - jacobi\_serial::SerialJacobi, 34
- normalize
  - gauss\_jordan::GaussJordan, 12
- partial\_pivot
  - serial\_gaussian\_elimination::SerialGaussianElimination, 29
- relaxation
  - jacobi\_serial::SerialJacobi, 34
- serial\_decomposition\_LU.SerialLUDecomposition, 35
- serial\_decomposition\_LU::SerialLUDecomposition
  - decomposition\_LU, 36
  - solve\_system, 36
- serial\_gauss\_jordan.GaussJordanSerial, 13
- serial\_gauss\_jordan::GaussJordanSerial
  - elimination, 14
- serial\_gaussian\_elimination.SerialGaussianElimination, 29
- serial\_gaussian\_elimination::SerialGaussianElimination
  - elimination, 29
  - partial\_pivot, 29
- sm\_test.SparseMatrix, 38
- sm\_testCSR.SparseMatrix, 38
- solve\_system
  - serial\_decomposition\_LU::SerialLUDecomposition, 36
- sparse\_matrix.SparseMatrix, 36
- sparse\_matrix::SparseMatrix
  - create\_sparse\_matrix, 37
  - gen\_vector, 37
  - load\_sparse\_matrix, 37
  - multiply, 37
- sparse\_matrix\_tab.SparseMatrixTab, 38
- SparseMatrices, 8
- start
  - gauss\_jordan::GaussJordan, 13
  - gaussian\_elimination::GaussianElimination, 10
  - gaussian\_lu\_decomposition::GaussianLUDecomposition, 17
  - jacobi\_parallel::JacobiParallel, 21
  - jacobi\_parallel\_chunks::JacobiParallel, 19
- sum\_vectors
  - jacobi\_serial::SerialJacobi, 35