GPU solvers for SLAE

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

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

Decompuses a matrix A into two matrices L and U.

4.4.1 Detailed Description

Decompuses 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.

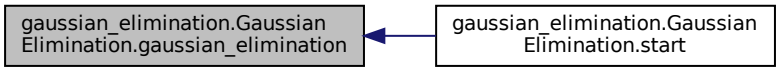
Parameters

A	Augmented matrix representing a SLAE.
size	Size of coefficiente matrix.
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.

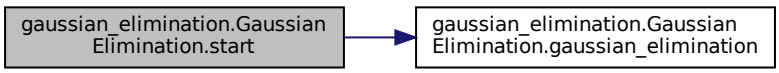
Parameters

A_matrix	Coefficient matrix of a SLAE.
b_matrix	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.

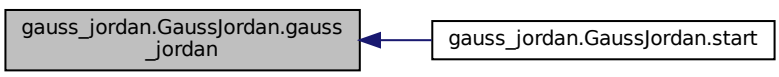
Parameters

A	Augmented matrix representing a SLAE.
size	Size of coefficiente matrix.
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.

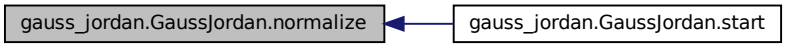
Parameters

A	Augmented matrix representing a SLAE.
size	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.

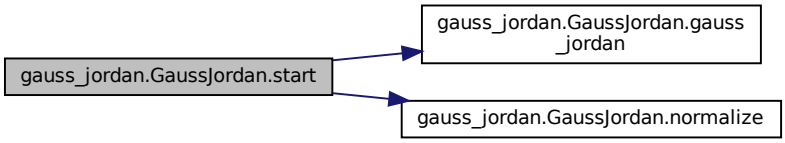
Parameters

A_matrix	Coefficient matrix of a SLAE.
b_vector	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.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 gaussian_lu_decomposition()

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

Performs Gaussian LU elimination.

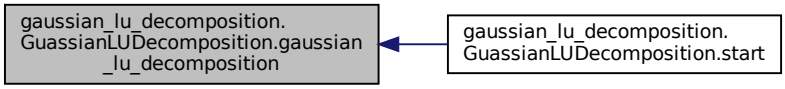
Parameters

A	Coefficient matrix A.
L	Matrix in which to store the multipliers.
size	Size of coefficiente matrix.
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.

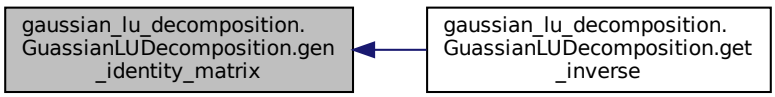
Parameters

size	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:

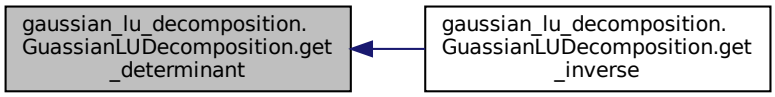
Parameters

L	The lower triangular matrix of the system.
U	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.

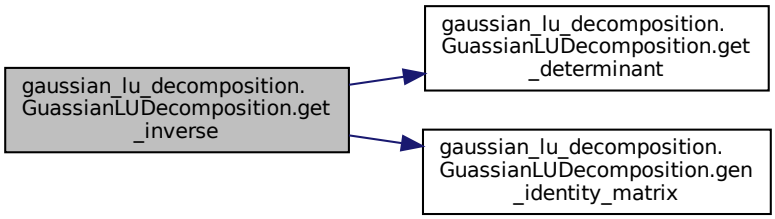
Parameters

L	The lower triangular matrix of the system.
U	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.GuassianLUDecomposition.get_solution (
              self,
              L,
              U,
              b )
```

Solves a LU system.

Parameters

L	The lower triangular matrix of the system.
U	The upper triangular matrix of the system.
b	Linearly independent vector.

Returns

float64[:]

5.7.1.6 start()

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

Decomposes A_matrix into two matrices L and U.

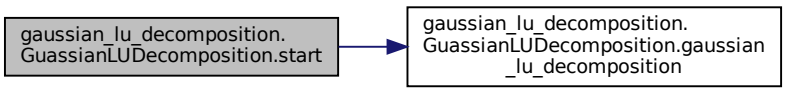
Parameters

A_matrix	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**

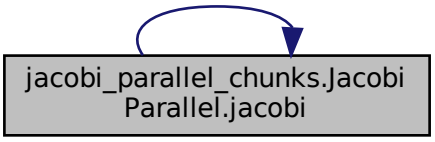
5.8.1 Member Function Documentation

5.8.1.1 jacobi()

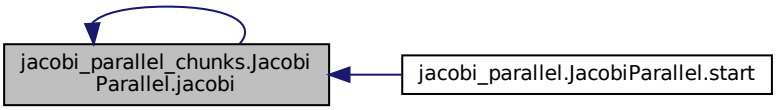
```
def jacobi_parallel_chunks.JacobiParallel.jacobi (
              A,
              b,
              x_current,
              x_next,
              rows,
              cols,
              first_row_block,
              rel )
```

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.

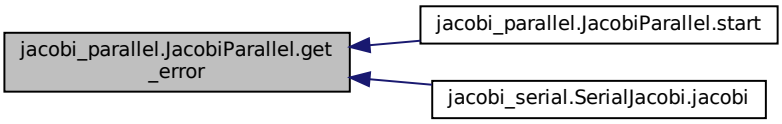
Parameters

x_current	Pointer to list representing current approximation for vector x in a system Ax = b.
x_next	Pointer to list representing new approximation for vector x in a system Ax = b.
x_error	Pointer to list in which an error for each approximation will be stored.
rows	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

A	Coefficient matrix.
b	Linearly independent vector.
x_current	Current answer's approximation.
x_next	vector in which to store new answer.
n	Coefficient matrix' size.
rel	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.

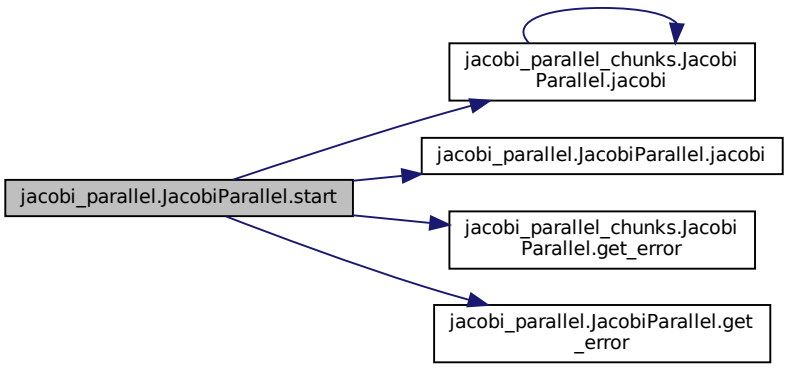
Parameters

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

size	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

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

size	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

size	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

size	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

size	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

size	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

size	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

size	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

size	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

size	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

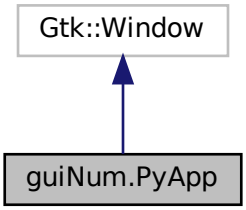
- **matrix_filename_entry**
- **vector_filename_entry**
- **length_entry**
- **selected_generator**

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

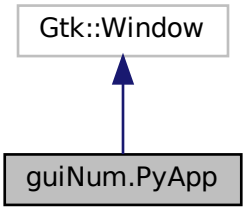
- matrixGenerator/matrix_generator_tab.py

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.

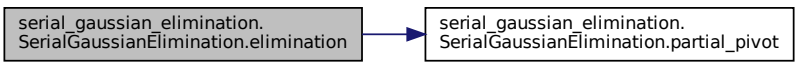
Parameters

A	The coefficient matrix of the system.
b	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.

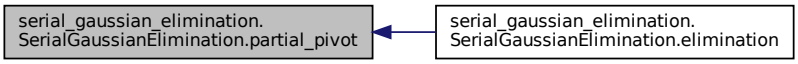
Parameters

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)

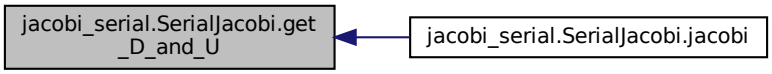
Parameters

matrix	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.

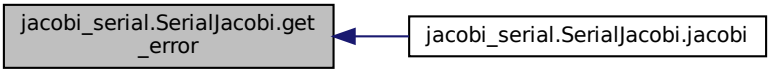
Parameters

x_vector	The vector of the current stage of the method.
xant_vector	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.

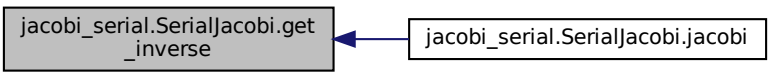
Parameters

matrixD	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.

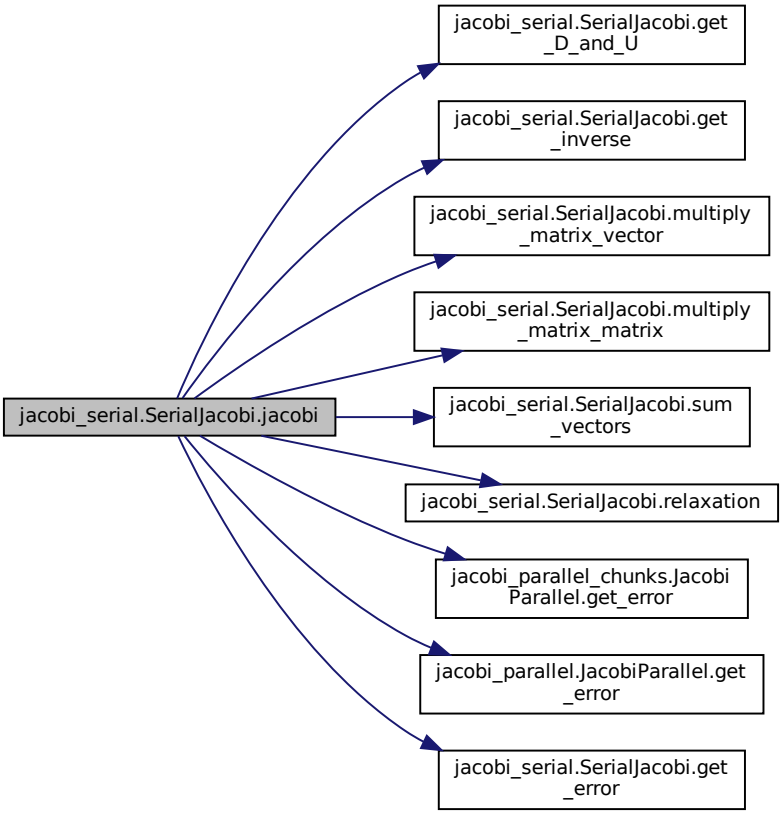
Parameters

A_matrix	The coefficient matrix of the system.
b_vector	The linearly independent vector.
max_iterations	Maximum number of iterations of the method.
tolerance	The tolerance of the method
relaxation	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.

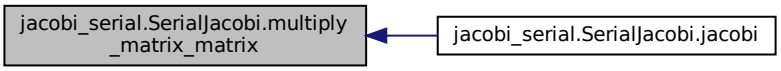
Parameters

matrix1	The first matrix to be multiplied.
matrix2	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.

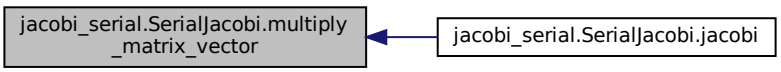
Parameters

A_matrix	The matrix to be multiplied.
b_vector	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.

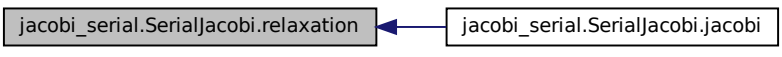
Parameters

x_vector	The vector of the current stage of the method.
xant_vector	The vector of the previous stage of the method.
relaxation	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.

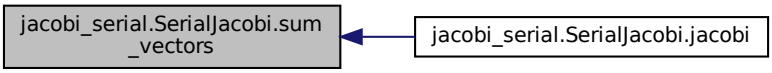
Parameters

vector1	The first vector to be added.
vector2	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

A	The coefficient matrix to be splited.

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

L	The lower triangular matrix of the system.
U	The upper triangular matrix of the system.
b	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

filename	The file name where will be stored the final result.
matrix_length	The length of the matrix.
density	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

size	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

filename	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

filename_matrix	The filename where the CSR matrix is located.
vector	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