Combustion of Packed Pellets of Core-Shell Particle

Generated by Doxygen 1.8.17

Chapter 1

combustion-packed-pellet-core-shell-particle

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 GMatrix< real_t > Class Template Reference

Class to implement a memory efficient Givens' Rotation Matrix that can vanish the element $a_{k+1,k}$ of a R matrix A.

`#include <G_Matrix.hpp>`

Public Member Functions

- GMatrix (RMatrix< real_t > &matrix, unsigned int index)

  Construct a new Givens Rotation Matrix to vanish the element at position $(k+1, k)$ of R matrix.
- void multiply (RMatrix< real_t > &R)

  Multiplies the Givens Rotation Matrix to the R Matrix and updates the R matrix in place.
- void multiplyLastRow (RMatrix< real_t > &R)

  Multiplies the Givens rotation matrix to the R matrix where the rotation matrix is formed to vanish the sub diagonal element of the last row $(N, N-1)$ of the $N \times N$ R matrix.
- void multiply (QMatrix< real_t > &Q)

  Multiplies the Givens Rotation Matrix to the Q Matrix and updates the Q matrix in place.

Private Attributes

- real_t g_k_k

  Element of Givens Rotation Matrix at position k, k.
- real_t g_k_kp1

  Element of Givens Rotation Matrix at position $k, k+1$.
- real_t g_kp1_k

  Element of Givens Rotation Matrix at position $k+1, k$.
- real_t g_kp1_kp1

  Element of Givens Rotation Matrix at position $k+1, k+1$.
- unsigned int k

  Index k, where we want to vanish the element at position $k+1, k$.

4.1.1 Detailed Description

template<typename real_t>
class GMatrix< real_t >

Class to implement a memory efficient Givens' Rotation Matrix that can vanish the element $a_{k+1,k}$ of a R matrix A.

A Givens rotation matrix is an orthogonal such that $G_{N\times N} = [g_{ij}]_{N\times N}$ where

$$g_{ij} = \begin{cases} 1 & i = j \neq k, k+1 \\ \cos\theta & i = j = k, k+1 \\ \sin\theta & i = k, j = k+1 \\ -\sin\theta & i = k+1, j = k \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

When the Givens Rotation Matrix is multiplied to another matrix $B_{N\times N} = [b_{i,j}]_{N\times N}$

$$C_{N\times N} = G_{N\times N} \cdot B_{N\times N} \tag{4.2}$$

$$\Rightarrow c_{ij} = \sum_{l=1}^{N} g_{i,l} b_{l,j} \tag{4.3}$$

$$\Rightarrow c_{ij} = \begin{cases} g_{k,k}b_{k,j} + g_{k,k+1}b_{k+1,j} & i = k \\ g_{k+1,k}b_{k,j} + g_{k+1,k+1}b_{k+1,j} & i = k+1 \\ b_{i,j} & \text{otherwise} \end{cases} \tag{4.4}$$

Template Parameters

real_t	float, double or long double data types to represent real numbers

Definition at line 58 of file G_Matrix.hpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 GMatrix()

```
template<typename real_t >
GMatrix< real_t >::GMatrix (
            RMatrix< real_t > & matrix,
            unsigned int index )
```

Construct a new Givens Rotation Matrix to vanish the element at position $(k+1, k)$ of R matrix.

Parameters

R	R matrix whose element at position $(k+1, k)$ needs to be vanished
index	k

Definition at line 18 of file G_Matrix.cpp.

4.1.3 Member Function Documentation

4.1.3.1 multiply() [1/2]

```
template<typename real_t >
void GMatrix< real_t >::multiply (
             QMatrix< real_t > & Q )
```

Multiplies the Givens Rotation Matrix to the Q Matrix and updates the Q matrix in place.

Parameters

Q	Q Matrix passed as reference

Q matrix may be represented as $Q_{N\times N} = [q_{i,j}]_{N\times N}$ where

$$q_{i,j} = 0, \quad j > i$$

Thus, when a rotation matrix is multiplied to Q matrix

$$C_{N\times N} = G_{N\times N} \cdot Q_{N\times N} \tag{4.5}$$

$$\Rightarrow c_{ij} = \sum_{l=1}^{N} g_{i,l} q_{l,j} \tag{4.6}$$

$$\Rightarrow c_{ij} = \begin{cases} g_{k,k} l_{k,j} + g_{k,k+1} q_{k+1,j} & i = k \\ g_{k+1,k} l_{k,j} + g_{k+1,k+1} q_{k+1,j} & i = k+1 \\ q_{i,j} & \text{otherwise} \end{cases} \tag{4.7}$$

Definition at line 40 of file G_Matrix.cpp.

4.1.3.2 multiply() [2/2]

```
template<typename real_t >
void GMatrix< real_t >::multiply (
             RMatrix< real_t > & R )
```

Multiplies the Givens Rotation Matrix to the R Matrix and updates the R matrix in place.

Parameters

R	R Matrix that will be converted into upper tridiagonal matrix after multiplication

R matrix can be represented as $R_{N\times N} = [r_{i,j}]_{N\times N}$ where

$$r_{i,j} = \quad 0 \quad \text{if} \quad j < i-1 \quad \text{or} \quad j > i+1 \tag{4.8}$$

Thus, upon multiplying the R matrix with the rotation matrix, we get matrix $C_{N\times N}$

$$c_{ij} = \begin{array}{ll} g_{k,k}r_{k,j} + g_{k,k+1}r_{k+1,j} & i = k \\ g_{k+1,k}r_{k,j} + g_{k+1,k+1}r_{k+1,j} & i = k+1 \\ r_{i,j} & \text{otherwise} \end{array} \tag{4.9}$$

$$\Rightarrow c_{ij} = \begin{array}{lll} g_{k,k}r_{k,k-1} & i = k & j = k-1 \\ g_{k,k}r_{k,k} + g_{k,k+1}r_{k+1,k} & i = k & j = k \\ g_{k,k}r_{k,k+1} + g_{k,k+1}r_{k+1,k+1} & i = k & j = k+1 \\ g_{k,k+1}r_{k+1,k+2} & i = k & j = k+2 \\ g_{k+1,k}r_{k,k-1} & i = k+1 & j = k-1 \\ g_{k+1,k}r_{k,k} + g_{k+1,k+1}r_{k+1,k} & i = k+1 & j = k \\ g_{k+1,k}r_{k,k+1} + g_{k+1,k+1}r_{k+1,k+1} & i = k+1 & j = k+1 \\ g_{k+1,k+1}r_{k+1,k+2} & i = k+1 & j = k+2 \\ r_{i,j} & \text{otherwise} & \end{array} \tag{4.10}$$

But as we iterate over k and multiply the corresponding rotation marix, the sub-diagonal ($r_{i,i-1}$) becomes zero and the super-super-diagonal ($r_{i,i+2}$) fills up with non-zero values.

$$\Rightarrow c_{ij} = \begin{array}{lll} 0 & i = k & j = k-1 \\ g_{k,k}r_{k,k} + g_{k,k+1}r_{k+1,k} & i = k & j = k \\ g_{k,k}r_{k,k+1} + g_{k,k+1}r_{k+1,k+1} & i = k & j = k+1 \\ g_{k,k+1}r_{k+1,k+2} & i = k & j = k+2 \\ 0 & i = k+1 & j = k-1 \\ 0 & i = k+1 & j = k \\ g_{k+1,k}r_{k,k+1} + g_{k+1,k+1}r_{k+1,k+1} & i = k+1 & j = k+1 \\ g_{k+1,k+1}r_{k+1,k+2} & i = k+1 & j = k+2 \\ r_{i,j} & \text{otherwise} & \end{array} \tag{4.11}$$

Definition at line 63 of file G_Matrix.cpp.

4.1.3.3 multiplyLastRow()

```
template<typename real_t >
void GMatrix< real_t >::multiplyLastRow (
            RMatrix< real_t > & R )
```

Multiplies the Givens rotation matrix to the R matrix where the rotation matrix is formed to vanish the sub diagonal element of the last row $(N, N-1)$ of the $N \times N$ R matrix.

$$\Rightarrow c_{ij} = \begin{array}{lll} 0 & i = k & j = k-1 \\ g_{N-1,N-1}r_{N-1,N-1} + g_{N-1,N}r_{N,N-1} & i = k & j = k \\ g_{N-1,N-1}r_{N-1,N} + g_{N-1,N}r_{N,N} & i = k & j = k+1 \\ 0 & i = k+1 & j = k-1 \\ 0 & i = k+1 & j = k \\ g_{N,N-1}r_{N-1,N} + g_{N,N}r_{N,N} & i = k+1 & j = k+1 \\ r_{i,j} & \text{otherwise} & \end{array} \tag{4.12}$$

Parameters

R	R Matrix that will be converted into upper tridiagonal matrix after multiplication

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

- include/G_Matrix.hpp
- src/G_Matrix.cpp

4.2 QMatrix< real_t > Class Template Reference

Class to implement a memory efficient model of $N\times$ Q Matrix.

`#include <Q_Matrix.hpp>`

Public Member Functions

- QMatrix (unsigned int n)
  Construct a new Q Matrix.
- ~QMatrix ()
  Destroy the Q Matrix object.
- real_t getElement (unsigned int row_index, unsigned int column_index)
  Get the i,j th element of the Q Matrix.
- void setElement (unsigned int row_index, unsigned int column_index, real_t value)
  Set the value of the i,j th element of the Q Matrix.
- void printMatrix ()
  Prints the Q Matrix in form of a 2D array.
- void fillZeroes ()
  Fills zeros in all the position of the Q Matrix.
- void multiply (real_t *b, real_t *x)
  Multiplies the Q matrix with the column vector b and stores the result in the column vector x.

Private Member Functions

- unsigned int getIndex (unsigned int row_index, unsigned int column_index)
  Get the index of i,j th element of Q matrix in the flattened array.
- bool indexOfZeroElement (unsigned int row_index, unsigned int column_index)
  Checks if the row index i and the column index j belong to a zero element of the Q matrix.

Private Attributes

- real_t * array
  One dimensional array to store only non zero elements of the lower triangular matrix.
- const unsigned int N
  Number of rows in a $N \times N$ square matrix.

4.2.1 Detailed Description

template<typename real_t>
class QMatrix< real_t >

Class to implement a memory efficient model of $N\times$ Q Matrix.

Template Parameters

real_t	float, double or long double data types to represent real numbers

Definition at line 25 of file Q_Matrix.hpp.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 QMatrix()

```
template<typename real_t >
QMatrix< real_t >::QMatrix (
            unsigned int n )
```

Construct a new Q Matrix.

Parameters

n	Number of rows in the $N \times N$ square matrix

Definition at line 21 of file Q_Matrix.cpp.

4.2.3 Member Function Documentation

4.2.3.1 getElement()

```
template<typename real_t >
real_t QMatrix< real_t >::getElement (
            unsigned int row_index,
            unsigned int column_index )
```

Get the i,j th element of the Q Matrix.

Parameters

row_index	Row index i
column_index	Column index j

Returns

Value of the i,j th element of the Q Matrix

Definition at line 62 of file Q_Matrix.cpp.

4.2.3.2 getIndex()

```
template<typename real_t >
unsigned int QMatrix< real_t >::getIndex (
            unsigned int row_index,
            unsigned int column_index )  [private]
```

Get the index of i,j th element of Q matrix in the flattened array.

Parameters

row_index	Row index i
column_index	Column index j

Returns

Index of the i,j th element in the flattened array

Definition at line 41 of file Q_Matrix.cpp.

4.2.3.3 indexOfZeroElement()

```
template<typename real_t >
bool QMatrix< real_t >::indexOfZeroElement (
            unsigned int row_index,
            unsigned int column_index )  [private]
```

Checks if the row index i and the column index j belong to a zero element of the Q matrix.

Parameters

row_index	Row index i
column_index	Column index j

Returns

true if i,j are indices of zero elements in a Q matrix

false if i,j are indices of non zero elements in a Q matrix

Definition at line 52 of file Q_Matrix.cpp.

4.2.3.4 multiply()

```
template<typename real_t >
void QMatrix< real_t >::multiply (
            real_t * b,
            real_t * x )
```

Multiplies the Q matrix with the column vector b and stores the result in the column vector x.

Parameters

b	
x	

Definition at line 127 of file Q_Matrix.cpp.

4.2.3.5 setElement()

```
template<typename real_t >
void QMatrix< real_t >::setElement (
            unsigned int row_index,
            unsigned int column_index,
            real_t value )
```

Set the value of the i,j th element of the Q Matrix.

Parameters

row_index	Row index i
column_index	Column index j
value	Value to be set at the i,j th position

Definition at line 80 of file Q_Matrix.cpp.

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

- include/Q_Matrix.hpp
- src/Q_Matrix.cpp

4.3 QRSolver< real_t > Class Template Reference

Class to implement QR factorization algorithm for solving matrix equations of the A.x = b where A is a n x n tridiagonal matrix and x and b are n x 1 vectors.

```
#include <QR_Solver.hpp>
```

Public Member Functions

- QRSolver (unsigned int N)

  Construct a new QRSolver object.
- ~QRSolver ()

  Destroy the QRSolver object.
- void **QRfactorize** ()
- void **initQ** ()

Private Member Functions

- void loadR ()

  Loads the values of R matrix that will change during multiplication with Givens' rotation matrix to temporary variables.
- void setupGivensRotationMatrix ()

  Setup Givens' rotation matrix.
- void multiplyGivensMatrixWithR ()

  Multiplies the Givens rotation matrix with R matrix and updates its value.
- void multiplyGivensMatrixWithQ ()

  Multiplies the Givens rotation matrix with R matrix and updates its value.
- unsigned int getIndex (unsigned int row_index, unsigned int column_index)

  Get the index in a flattened linear array representation of a 2D matrix.

Private Attributes

- real_t * **Q**
- real_t **G_k_k**
- real_t **G_k_kp1**
- real_t **G_kp1_k**
- real_t **G_kp1_kp1**
- real_t **R_k_k**
- real_t **R_k_kp1**
- real_t **R_kp1_k**
- real_t **R_kp1_kp1**
- real_t **R_kp1_kp2**
- unsigned int **k**

4.3.1 Detailed Description

template<typename real_t>
class QRSolver< real_t >

Class to implement QR factorization algorithm for solving matrix equations of the A.x = b where A is a n x n tridiagonal matrix and x and b are n x 1 vectors.

Definition at line 30 of file QR_Solver.hpp.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 QRSolver()

```
template<typename real_t >
QRSolver< real_t >::QRSolver (
            unsigned int N )
```

Construct a new QRSolver object.

Parameters

N	

Definition at line 5 of file QR_Solver.cpp.

4.3.2.2 ~QRSolver()

```
template<typename real_t >
QRSolver< real_t >::~QRSolver ( )
```

Destroy the QRSolver object.

Definition at line 14 of file QR_Solver.cpp.

4.3.3 Member Function Documentation

4.3.3.1 getIndex()

```
template<typename real_t >
unsigned int QRSolver< real_t >::getIndex (
            unsigned int row_index,
            unsigned int column_index )  [inline], [private]
```

Get the index in a flattened linear array representation of a 2D matrix.

Parameters

row_index	Row index i of the desired element
column_index	Column index j of the desired element

Returns

Index in a linear array of the i,j element of a 2D matrix implemented using the linear array

Definition at line 90 of file QR_Solver.hpp.

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

- include/QR_Solver.hpp
- src/QR_Solver.cpp

4.4 RMatrix< real_t > Class Template Reference

Class to implement a memory efficient $N \times N$ R matrix.

`#include <R_Matrix.hpp>`

Public Member Functions

- RMatrix (unsigned int n)
  Construct a new R Matrix.
- ~RMatrix ()
  Destroy the R Matrix.
- real_t getElement (unsigned int row_index, unsigned int column_index)
  Get the i,j th element of R Matrix.
- void setElement (unsigned int row_index, unsigned int column_index, real_t value)
  Set the value of the i,j th element of R Matrix.
- void printMatrix ()
  Prints the R Matrix in form of a 2D array.
- void print ()
  Prints the R Matrix in form of a flattened array.

Private Member Functions

- unsigned int getIndex (unsigned int row_index, unsigned int column_index)
  Get the index of the i,j th element of R Matrix in the flattened array.
- bool indexOfZeroElement (unsigned int row_index, unsigned int column_index)
  Checks if the row index i and the column index j belong to a zero element of the R matrix.

Private Attributes

- real_t * array
  *Flattened array of size 4 * N to represent R matrix of size $N \times N$.*
- const unsigned int N
  Size of main diagonal of the $N \times N$ R Matrix.

4.4.1 Detailed Description

template<typename real_t>
class RMatrix< real_t >

Class to implement a memory efficient $N \times N$ R matrix.

The class is specifically built for implementation in a QR factorization algorithm for solving matrix equations of the form $A \cdot x = b$. The QR algo converts a normal tridiagonal matrix (a matrix with non zero entries only at indices $(i, i-1)$, (i, i) and $(i, i+1)$) to an upper tridiagonal matrix (a matrix with non zero entries only at indices (i, i), $(i, i+1)$ and $(i, i+2)$). Thus only indices $(i, i-1)$, (i, i), $(i, i+1)$ and $(i, i+2)$ are stored in memory for this matrix

Definition at line 30 of file R_Matrix.hpp.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 RMatrix()

```
template<typename real_t >
RMatrix< real_t >::RMatrix (
            unsigned int n )
```

Construct a new R Matrix.

Parameters

n	Size of main diagonal of the $N \times N$ R Matrix

Definition at line 19 of file R_Matrix.cpp.

4.4.2.2 ~RMatrix()

```
template<typename real_t >
RMatrix< real_t >::~RMatrix
```

Destroy the R Matrix.

Definition at line 30 of file R_Matrix.cpp.

4.4.3 Member Function Documentation

4.4.3.1 getElement()

```
template<typename real_t >
real_t RMatrix< real_t >::getElement (
            unsigned int row_index,
            unsigned int column_index )
```

Get the i,j th element of R Matrix.

Parameters

row_index	Row index i
column_index	Column index j

Returns

Value of the i,j th element of a R Matrix

Definition at line 37 of file R_Matrix.cpp.

4.4.3.2 getIndex()

```
template<typename real_t >
unsigned int RMatrix< real_t >::getIndex (
            unsigned int row_index,
            unsigned int column_index )  [private]
```

Get the index of the i,j th element of R Matrix in the flattened array.

Parameters

row_index	Row index i
column_index	Column index j

Returns

Returns the index in the the flattened array

Definition at line 110 of file R_Matrix.cpp.

4.4.3.3 indexOfZeroElement()

```
template<typename real_t >
bool RMatrix< real_t >::indexOfZeroElement (
            unsigned int row_index,
            unsigned int column_index )  [private]
```

Checks if the row index i and the column index j belong to a zero element of the R matrix.

Parameters

row_index	
column_index	

Returns

true if i,j are indices of zero elements in a R matrix

false if i,j are indicess of non-zero elements in a R matrix

Definition at line 126 of file R_Matrix.cpp.

4.4.3.4 print()

```
template<typename real_t >
void RMatrix< real_t >::print
```

Prints the R Matrix in form of a flattened array.

Definition at line 95 of file R_Matrix.cpp.

4.4.3.5 printMatrix()

```
template<typename real_t >
void RMatrix< real_t >::printMatrix
```

Prints the R Matrix in form of a 2D array.

Definition at line 76 of file R_Matrix.cpp.

4.4.3.6 setElement()

```
template<typename real_t >
void RMatrix< real_t >::setElement (
            unsigned int row_index,
            unsigned int column_index,
            real_t value )
```

Set the value of the i,j th element of R Matrix.

Parameters

row_index	Row index i
column_index	Column index j
value	Value to be set at the i,j th element

Definition at line 55 of file R_Matrix.cpp.

4.4.4 Member Data Documentation

4.4.4.1 array

```
template<typename real_t >
real_t* RMatrix< real_t >::array  [private]
```

Flattened array of size 4 $*$ N to represent R matrix of size $N \times N$.

Definition at line 93 of file R_Matrix.hpp.

4.4.4.2 N

```
template<typename real_t >
const unsigned int RMatrix< real_t >::N  [private]
```

Size of main diagonal of the $N \times N$ R Matrix.

Definition at line 100 of file R_Matrix.hpp.

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

- include/R_Matrix.hpp
- src/R_Matrix.cpp

Chapter 5

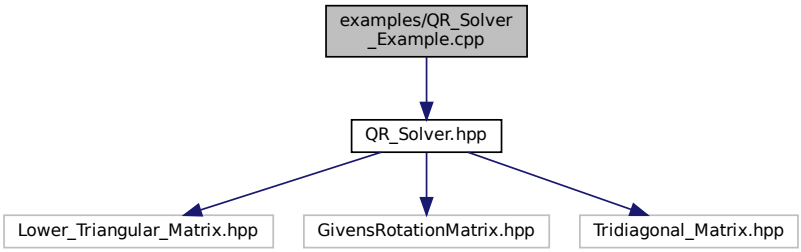
File Documentation

5.1 examples/QR_Solver_Example.cpp File Reference

Example cpp file to test out QR_Solver functions.

```
#include "QR_Solver.hpp"
```

Include dependency graph for QR_Solver_Example.cpp:



Functions

- QR_Solver **my_solver** (5)
- int **main** (int argc, char const ∗argv[ ])

Variables

- real_t **A** [3 ∗5]

5.1.1 Detailed Description

Example cpp file to test out QR_Solver functions.

Author

Souritra Garai (souritra.garai@iitgn.ac.in)

Version

0.1

Date

2021-06-24

Copyright

5.2 include/Q_Matrix.hpp File Reference

This header file defines a class for memory efficient implementation of Q matrix used for QR factorisation of tridiagonal matrix using Givens rotation matrix.

Classes

- class QMatrix< real_t >

  Class to implement a memory efficient model of $N\times$ Q Matrix.

5.2.1 Detailed Description

This header file defines a class for memory efficient implementation of Q matrix used for QR factorisation of tridiagonal matrix using Givens rotation matrix.

Author

Souritra Garai (souritra.garai@iitgn.ac.in)

Version

0.1

Date

2021-06-25

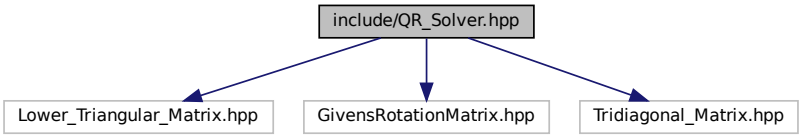
Copyright

5.3 include/QR_Solver.hpp File Reference

This header file defines a class for solving 2D matrix equations of the form A.x = b (where A is an n x n matrix and x and b are n x 1 vectors) using QR factorization technique. Also the class is implemented in such a way that it may be parallelized easily using openmp constructs.

```
#include "Lower_Triangular_Matrix.hpp"
#include "GivensRotationMatrix.hpp"
#include "Tridiagonal_Matrix.hpp"
```

Include dependency graph for QR_Solver.hpp:



Classes

- class QRSolver< real_t >

  Class to implement QR factorization algorithm for solving matrix equations of the A.x = b where A is a n x n tridiagonal matrix and x and b are n x 1 vectors.

5.3.1 Detailed Description

This header file defines a class for solving 2D matrix equations of the form A.x = b (where A is an n x n matrix and x and b are n x 1 vectors) using QR factorization technique. Also the class is implemented in such a way that it may be parallelized easily using openmp constructs.

Author

Souritra Garai (souritra.garai@iitgn.ac.in)

Version

0.1

Date

2021-06-23

Copyright

5.4 include/R_Matrix.hpp File Reference

This header file serves the definition of an implementation for a R Matrix that is used for QR Factorization of a tridiagonal matrix.

Classes

- class RMatrix< real_t >

  Class to implement a memory efficient $N \times N$ R matrix.

5.4.1 Detailed Description

This header file serves the definition of an implementation for a R Matrix that is used for QR Factorization of a tridiagonal matrix.

Author

Souritra Gari (souritra.garai@iitgn.ac.in)

Version

0.1

Date

2021-06-24

Copyright

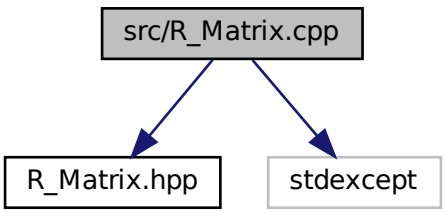
Copyright (c) 2021

5.5 src/R_Matrix.cpp File Reference

Implementation of R Matrix for QR Factorization of tridiagonal matrices using Givens' rotation matrices.

```
#include "R_Matrix.hpp"
#include <stdexcept>
```

Include dependency graph for R_Matrix.cpp:



5.5.1 Detailed Description

Implementation of R Matrix for QR Factorization of tridiagonal matrices using Givens' rotation matrices.

Author

Souritra Gari (souritra.garai@iitgn.ac.in)

Version

0.1

Date

2021-06-24

Copyright

Index