

Combustion of Packed Pellets of Core-Shell Particle

Generated by Doxygen 1.8.17

1 combustion-packed-pellet-core-shell-particle	1
2 Namespace Index	3
2.1 Namespace List	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 QR_Factorization.py Namespace Reference	9
5.1.1 Detailed Description	9
6 Class Documentation	11
6.1 CoreShellCombustionParticle< real_t > Class Template Reference	11
6.1.1 Detailed Description	12
6.1.2 Constructor & Destructor Documentation	12
6.1.2.1 CoreShellCombustionParticle()	12
6.1.3 Member Function Documentation	13
6.1.3.1 getDensity()	13
6.1.3.2 getDiffusivity()	13
6.1.3.3 getEnthalpy()	14
6.1.3.4 getHeatCapacity()	14
6.1.3.5 getHeatConductivity()	14
6.1.3.6 printProperties()	14
6.2 GMatrix< real_t > Class Template Reference	15
6.2.1 Detailed Description	16
6.2.2 Constructor & Destructor Documentation	16
6.2.2.1 GMatrix()	16
6.2.3 Member Function Documentation	17
6.2.3.1 multiply() [1/2]	17
6.2.3.2 multiply() [2/2]	17
6.2.3.3 multiplyLastRow()	18
6.3 QMatrix< real_t > Class Template Reference	19
6.3.1 Detailed Description	19
6.3.2 Constructor & Destructor Documentation	20
6.3.2.1 QMatrix()	20
6.3.3 Member Function Documentation	20
6.3.3.1 getElement()	20
6.3.3.2 getIndex()	21
6.3.3.3 indexOfZeroElement()	21
6.3.3.4 multiply()	22

6.3.3.5 setElement()	22
6.4 QRSolver< real_t > Class Template Reference	22
6.4.1 Detailed Description	23
6.4.2 Constructor & Destructor Documentation	24
6.4.2.1 QRSolver()	24
6.4.3 Member Function Documentation	24
6.4.3.1 getSolution()	24
6.4.3.2 printMatrixEquation()	24
6.4.3.3 printQRMatrices()	25
6.4.3.4 setEquation()	25
6.4.3.5 setEquationFirstRow()	25
6.4.3.6 setEquationLastRow()	26
6.4.4 Member Data Documentation	26
6.4.4.1 R	26
6.5 RMatrix< real_t > Class Template Reference	27
6.5.1 Detailed Description	27
6.5.2 Constructor & Destructor Documentation	28
6.5.2.1 RMatrix()	28
6.5.2.2 ~RMatrix()	28
6.5.3 Member Function Documentation	28
6.5.3.1 getElement()	28
6.5.3.2 getIndex()	29
6.5.3.3 indexOfZeroElement()	29
6.5.3.4 print()	30
6.5.3.5 printMatrix()	30
6.5.3.6 setElement()	30
6.5.4 Member Data Documentation	30
6.5.4.1 array	30
6.5.4.2 N	31
6.6 Substance< real_t > Class Template Reference	31
6.6.1 Detailed Description	32
6.6.2 Constructor & Destructor Documentation	32
6.6.2.1 Substance()	32
6.6.3 Member Function Documentation	33
6.6.3.1 getDensity()	33
6.6.3.2 getEnthalpy()	33
6.6.3.3 getHeatCapacity()	34
6.6.3.4 getHeatConductivity()	34
6.6.3.5 getMolecularWeight()	35
6.6.3.6 getStandardEnthalpyOfFormation()	35
6.6.3.7 printProperties()	35

7 File Documentation	37
7.1 examples/qrsolver/G_Matrix_Example.cpp File Reference	37
7.1.1 Detailed Description	38
7.2 examples/qrsolver/Q_Matrix_Example.cpp File Reference	38
7.2.1 Detailed Description	39
7.3 examples/qrsolver/QR_Solver_Example.cpp File Reference	39
7.3.1 Detailed Description	40
7.4 examples/qrsolver/R_Matrix_Example.cpp File Reference	40
7.4.1 Detailed Description	41
7.5 examples/thermo-physical-properties/Core-Shell-Combustion-Particle_Example.cpp File Reference	41
7.5.1 Detailed Description	42
7.6 examples/thermo-physical-properties/Substance_Example.cpp File Reference	42
7.6.1 Detailed Description	43
7.7 include/qrsolver/G_Matrix.hpp File Reference	43
7.7.1 Detailed Description	44
7.8 include/qrsolver/Q_Matrix.hpp File Reference	44
7.8.1 Detailed Description	44
7.9 include/qrsolver/QR_Solver.hpp File Reference	45
7.9.1 Detailed Description	45
7.10 include/qrsolver/R_Matrix.hpp File Reference	46
7.10.1 Detailed Description	46
7.11 include/thermo-physical-properties/Core-Shell-Combustion-Particle.hpp File Reference	46
7.11.1 Detailed Description	47
7.12 include/thermo-physical-properties/Substance.hpp File Reference	48
7.12.1 Detailed Description	48
7.13 src/qrsolver/G_Matrix.cpp File Reference	49
7.13.1 Detailed Description	49
7.14 src/qrsolver/Q_Matrix.cpp File Reference	50
7.14.1 Detailed Description	50
7.15 src/qrsolver/QR_Solver.cpp File Reference	51
7.15.1 Detailed Description	51
7.16 src/qrsolver/R_Matrix.cpp File Reference	52
7.16.1 Detailed Description	52
7.17 src/thermo-physical-properties/Core-Shell-Combustion-Particle.cpp File Reference	53
7.17.1 Detailed Description	53
Index	55

Chapter 1

combustion-packed-pellet-core-shell-particle

Chapter 2

Namespace Index

2.1 Namespace List

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

QR_Factorization.py	9
---	---

Chapter 3

Class Index

3.1 Class List

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

CoreShellCombustionParticle< real_t >	
Class to represent Core-Shell Particle with functions estimate thermodynamic properties for varying composition and temperature	11
GMatrix< real_t >	
Class to implement a memory efficient Givens' Rotation Matrix that can vanish the element $a_{k+1,k}$ of a matrix A	15
QMatrix< real_t >	
Class to implement a memory efficient model of $N \times N$ Q Matrix	19
QRSolver< real_t >	
Class to implement QR factorization algorithm for solving matrix equations of the $A \cdot x = b$ where A is a $N \times N$ tridiagonal matrix and x and b are $N \times 1$ vectors	22
RMatrix< real_t >	
Class to implement a memory efficient $N \times N$ R matrix	27
Substance< real_t >	
Class to represent a pure substance	31

Chapter 4

File Index

4.1 File List

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

examples/qrsolver/G_Matrix_Example.cpp	
Example to test out GMatrix class	37
examples/qrsolver/Q_Matrix_Example.cpp	
Example to test QMatrix class	38
examples/qrsolver/QR_Solver_Example.cpp	
Example cpp file to test out QRSolver class	39
examples/qrsolver/R_Matrix_Example.cpp	
An example cpp program to test the RMatrix class	40
examples/thermo-physical-properties/Core-Shell-Combustion-Particle_Example.cpp	41
examples/thermo-physical-properties/Substance_Example.cpp	
Example to test functions of Substance class	42
include/qrsolver/G_Matrix.hpp	
This header serves the definition of an implementation of Givens' Rotation matrix. The rotation matrix is used to solve matrix equations through QR factorization method, particularly tridiagonal matrix equation	43
include/qrsolver/Q_Matrix.hpp	
This header file defines a class for memory efficient implementation of Q matrix used for QR factorisation of tridiagonal matrix using Givens rotation matrix	44
include/qrsolver/QR_Solver.hpp	
This header file defines a class for solving 2D matrix equations of the form $A \cdot x = b$ (where A is an $N \times N$ matrix and x and b are $N \times 1$ vectors) using QR factorization technique	45
include/qrsolver/R_Matrix.hpp	
This header file defines a class for memory efficient implementation of R matrix used for QR factorisation of tridiagonal matrix	46
include/thermo-physical-properties/Core-Shell-Combustion-Particle.hpp	
This header defines a class for core shell particle	46
include/thermo-physical-properties/Substance.hpp	
This header defines a class to represent pure substances with their thermodynamic properties like density, heat capacity etc	48
scripts/qrsolver/QR_Factorization.py	??
src/qrsolver/G_Matrix.cpp	
Member function definitions for GMatrix class	49
src/qrsolver/Q_Matrix.cpp	
Member function definitions for QMatrix class	50
src/qrsolver/QR_Solver.cpp	
Member function definitions for QRSolver class	51

src/qrsolver/ R_Matrix.cpp	
Member function definitions for RMatrix class	52
src/thermo-physical-properties/ Core-Shell-Combustion-Particle.cpp	53

Chapter 5

Namespace Documentation

5.1 QR_Factorization.py Namespace Reference

5.1.1 Detailed Description

Python program to quickly check out and understand QR Factorization method for solving matrix equations

The matrices are printed at each step of factorization to understand the transformation of the matrices

Chapter 6

Class Documentation

6.1 CoreShellCombustionParticle< real_t > Class Template Reference

Class to represent Core-Shell Particle with functions estimate thermodynamic properties for varying composition and temperature.

```
#include <Core-Shell-Combustion-Particle.hpp>
```

Public Member Functions

- [CoreShellCombustionParticle](#) ([Substance](#)< real_t > core_material, [Substance](#)< real_t > shell_material, [Substance](#)< real_t > product_material, real_t diffusivity_pre_exponential_factor, real_t diffusivity_↔ activation_energy)

Construct a new Core Shell Combustion Particle object.

- real_t [getDensity](#) ()

Get the Density of the Particle.

- real_t [getHeatCapacity](#) ()

Get the Heat Capacity of the particle.

- real_t [getHeatConductivity](#) ()

Get the Heat Conductivity of the particle.

- real_t [getEnthalpy](#) (real_t temperature)

Get the Enthalpy of the particle.

- real_t [getDiffusivity](#) (real_t temperature)

Get the Diffusivity for the interdiffusion of the core and shell material.

- void [printProperties](#) (std::ostream &output_stream)

Print the properties of the substance to the given stream.

Protected Attributes

- `real_t mass_fraction_reactant_A`
Mass fraction of reactant A.
- `real_t mass_fraction_reactant_B`
Mass fraction of reactant B.
- `real_t mass_fraction_product_AB`
Mass fraction of product AB.
- `real_t pre_exponential_factor`
Pre exponential factor for Arrhenius Diffusivity model.
- `real_t activation_energy`
Activation energy for Arrhenius Diffusivity model.
- `Substance< real_t > reactant_A`
Reactant A substance initially present in the core.
- `Substance< real_t > reactant_B`
Reactant B substance initially present in the shell.
- `Substance< real_t > product_AB`
Product AB substance.

6.1.1 Detailed Description

```
template<typename real_t>
class CoreShellCombustionParticle< real_t >
```

Class to represent Core-Shell Particle with functions estimate thermodynamic properties for varying composition and temperature.

Template Parameters

<code>real_t</code>	
---------------------	--

Definition at line 28 of file Core-Shell-Combustion-Particle.hpp.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 CoreShellCombustionParticle()

```
template<typename real_t >
CoreShellCombustionParticle< real_t >::CoreShellCombustionParticle (
    Substance< real_t > core_material,
    Substance< real_t > shell_material,
    Substance< real_t > product_material,
    real_t diffusivity_pre_exponential_factor,
    real_t diffusivity_activation_energy )
```

Construct a new Core Shell Combustion Particle object.

Parameters

<i>core_material</i>	Substance that forms the core material
<i>shell_material</i>	Substance that forms the shell material
<i>product_material</i>	Substance that is produced as a result of the reaction of the core and shell material
<i>diffusivity_pre_exponential_factor</i>	Pre-exponential factor in the Arrhenius model for diffusivity
<i>diffusivity_activation_energy</i>	Activation energy in the Arrhenius model for diffusivity

Definition at line 17 of file Core-Shell-Combustion-Particle.cpp.

6.1.3 Member Function Documentation

6.1.3.1 getDensity()

```
template<typename real_t >
real_t CoreShellCombustionParticle< real_t >::getDensity
```

Get the Density of the Particle.

Returns

real_t Density of the particle

Definition at line 59 of file Core-Shell-Combustion-Particle.cpp.

6.1.3.2 getDiffusivity()

```
template<typename real_t >
real_t CoreShellCombustionParticle< real_t >::getDiffusivity (
    real_t temperature )
```

Get the Diffusivity for the interdiffusion of the core and shell material.

Parameters

<i>temperature</i>	Overall temperature of the particle
--------------------	-------------------------------------

Returns

real_t Diffusivity for interdiffusion model at the specified temperature

Definition at line 68 of file Core-Shell-Combustion-Particle.cpp.

6.1.3.3 getEnthalpy()

```
template<typename real_t >
real_t CoreShellCombustionParticle< real_t >::getEnthalpy (
    real_t temperature )
```

Get the Enthalpy of the particle.

Parameters

<i>temperature</i>	Overall temperature of the particle
--------------------	-------------------------------------

Returns

real_t Enthalpy of the particle at the specified temperature

Definition at line 92 of file Core-Shell-Combustion-Particle.cpp.

6.1.3.4 getHeatCapacity()

```
template<typename real_t >
real_t CoreShellCombustionParticle< real_t >::getHeatCapacity
```

Get the Heat Capacity of the particle.

Returns

real_t Heat capacity of the particle

Definition at line 74 of file Core-Shell-Combustion-Particle.cpp.

6.1.3.5 getHeatConductivity()

```
template<typename real_t >
real_t CoreShellCombustionParticle< real_t >::getHeatConductivity
```

Get the Heat Conductivity of the particle.

Returns

real_t Heat capacity of the particle

Definition at line 83 of file Core-Shell-Combustion-Particle.cpp.

6.1.3.6 printProperties()

```
template<typename real_t >
void CoreShellCombustionParticle< real_t >::printProperties (
    std::ostream & output_stream )
```

Print the properties of the substance to the given stream.

Parameters

<code>output_stream</code>	Stream to which the properties are printed
----------------------------	--

Definition at line 36 of file Core-Shell-Combustion-Particle.cpp.

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

- include/thermo-physical-properties/Core-Shell-Combustion-Particle.hpp
- src/thermo-physical-properties/Core-Shell-Combustion-Particle.cpp

6.2 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 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 $k + 1, k$.

6.2.1 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 matrix A .

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

$$g_{ij} = \begin{cases} 1 & i = j \neq k, k+1 \\ \cos \theta & i = j = k, k+1 \\ \sin \theta & i = k, j = k+1 \\ -\sin \theta & i = k+1, j = k \\ 0 & \text{otherwise} \end{cases} \quad (6.1)$$

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

$$C_{N \times N} = G_{N \times N} \cdot B_{N \times N} \quad (6.2)$$

$$\Rightarrow c_{ij} = \sum_{l=1}^N g_{i,l} b_{l,j} \quad (6.3)$$

$$\Rightarrow c_{ij} = \begin{cases} g_{k,k} b_{k,j} + g_{k,k+1} b_{k+1,j} & i = k \\ g_{k+1,k} b_{k,j} + g_{k+1,k+1} b_{k+1,j} & i = k+1 \\ b_{i,j} & \text{otherwise} \end{cases} \quad (6.4)$$

Template Parameters

<i>real_t</i>	float, double or long double data types to represent real numbers
---------------	---

Definition at line 58 of file G_Matrix.hpp.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 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

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

Definition at line 18 of file G_Matrix.cpp.

6.2.3 Member Function Documentation

6.2.3.1 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

$$C_{N \times N} = G_{N \times N} \cdot Q_{N \times N} \quad (6.5)$$

$$\Rightarrow c_{ij} = \sum_{l=1}^N g_{i,l} q_{l,j} \quad (6.6)$$

$$\Rightarrow c_{ij} = \begin{cases} g_{k,k} l_{k,j} + g_{k,k+1} q_{k+1,j} & i = k \\ g_{k+1,k} l_{k,j} + g_{k+1,k+1} q_{k+1,j} & i = k + 1 \\ q_{i,j} & \text{otherwise} \end{cases} \quad (6.7)$$

Definition at line 49 of file G_Matrix.cpp.

6.2.3.2 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

$$r_{i,j} = 0 \quad \text{if } j < i - 1 \quad \text{or } j > i + 1 \quad (6.8)$$

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

$$c_{ij} = \begin{cases} g_{k,k}r_{k,j} + g_{k,k+1}r_{k+1,j} & i = k \\ g_{k+1,k}r_{k,j} + g_{k+1,k+1}r_{k+1,j} & i = k + 1 \\ r_{i,j} & \text{otherwise} \end{cases} \quad (6.9)$$

$$\Rightarrow c_{ij} = \begin{cases} g_{k,k}r_{k,k-1} & i = k & j = k - 1 \\ g_{k,k}r_{k,k} + g_{k,k+1}r_{k+1,k} & i = k & j = k \\ g_{k,k}r_{k,k+1} + g_{k,k+1}r_{k+1,k+1} & i = k & j = k + 1 \\ g_{k,k+1}r_{k+1,k+2} & i = k & j = k + 2 \\ g_{k+1,k}r_{k,k-1} & i = k + 1 & j = k - 1 \\ g_{k+1,k}r_{k,k} + g_{k+1,k+1}r_{k+1,k} & i = k + 1 & j = k \\ g_{k+1,k}r_{k,k+1} + g_{k+1,k+1}r_{k+1,k+1} & i = k + 1 & j = k + 1 \\ g_{k+1,k+1}r_{k+1,k+2} & i = k + 1 & j = k + 2 \\ r_{i,j} & \text{otherwise} \end{cases} \quad (6.10)$$

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

$$\Rightarrow c_{ij} = \begin{cases} 0 & i = k & j = k - 1 \\ g_{k,k}r_{k,k} + g_{k,k+1}r_{k+1,k} & i = k & j = k \\ g_{k,k}r_{k,k+1} + g_{k,k+1}r_{k+1,k+1} & i = k & j = k + 1 \\ g_{k,k+1}r_{k+1,k+2} & i = k & j = k + 2 \\ 0 & i = k + 1 & j = k - 1 \\ 0 & i = k + 1 & j = k \\ g_{k+1,k}r_{k,k+1} + g_{k+1,k+1}r_{k+1,k+1} & i = k + 1 & j = k + 1 \\ g_{k+1,k+1}r_{k+1,k+2} & i = k + 1 & j = k + 2 \\ r_{i,j} & \text{otherwise} \end{cases} \quad (6.11)$$

Definition at line 80 of file G_Matrix.cpp.

6.2.3.3 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.

$$\Rightarrow c_{ij} = \begin{cases} 0 & i = k & j = k - 1 \\ g_{N-1,N-1}r_{N-1,N-1} + g_{N-1,N}r_{N,N-1} & i = k & j = k \\ g_{N-1,N-1}r_{N-1,N} + g_{N-1,N}r_{N,N} & i = k & j = k + 1 \\ 0 & i = k + 1 & j = k - 1 \\ 0 & i = k + 1 & j = k \\ g_{N,N-1}r_{N-1,N} + g_{N,N}r_{N,N} & i = k + 1 & j = k + 1 \\ r_{i,j} & \text{otherwise} \end{cases} \quad (6.12)$$

Parameters

<i>R</i>	R Matrix that will be converted into upper tridiagonal matrix after multiplication
----------	--

Definition at line 135 of file G_Matrix.cpp.

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

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

6.3 QMatrix< real_t > Class Template Reference

Class to implement a memory efficient model of $N \times N$ 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 identity](#) ()
Makes Q matrix an identity 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.

6.3.1 Detailed Description

```
template<typename real_t>
class QMatrix< real_t >
```

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

Template Parameters

<i>real_t</i>	float, double or long double data types to represent real numbers
---------------	---

Definition at line 25 of file Q_Matrix.hpp.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 QMatrix()

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

Construct a new Q Matrix.

Parameters

<i>n</i>	Number of rows in the $N \times N$ square matrix
----------	--

Definition at line 21 of file Q_Matrix.cpp.

6.3.3 Member Function Documentation

6.3.3.1 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

<i>row_index</i>	Row index <i>i</i>
<i>column_index</i>	Column index <i>j</i>

Returns

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

Definition at line 62 of file Q_Matrix.cpp.

6.3.3.2 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

<i>row_index</i>	Row index i
<i>column_index</i>	Column index j

Returns

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

Definition at line 41 of file Q_Matrix.cpp.

6.3.3.3 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

<i>row_index</i>	Row index i
<i>column_index</i>	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.

6.3.3.4 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

<i>b</i>	
<i>x</i>	

Definition at line 132 of file Q_Matrix.cpp.

6.3.3.5 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

<i>row_index</i>	Row index i
<i>column_index</i>	Column index j
<i>value</i>	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/qrsolver/Q_Matrix.hpp](#)
- [src/qrsolver/Q_Matrix.cpp](#)

6.4 QRSolver< real_t > Class Template Reference

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

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

Public Member Functions

- [QRSolver](#) (unsigned int **N**)
Construct a new [QRSolver](#) object.
- [~QRSolver](#) ()
Destroy the [QRSolver](#) object.
- void [setEquation](#) (unsigned int index, real_t e, real_t f, real_t g, real_t **b**)
Set up equation represented by i th row of the matrix equation $ex_{i-1} + fx_i + gx_{i+1} = b$.
- void [setEquationFirstRow](#) (real_t f, real_t g, real_t **b**)
Set up equation represented by the first row of the matrix equation $fx_i + gx_{i+1} = b$.
- void [setEquationLastRow](#) (real_t e, real_t f, real_t **b**)
Set up equation represented by the last row of the matrix equation $ex_{i-1} + fx_i = b$.
- void [printMatrixEquation](#) ()
Prints the matrix A and vector b .
- void [printQRMatrices](#) ()
Prints the factorized matrices $Q \cdot R$.
- void [getSolution](#) (real_t *x)
Finds the solution to matrix equation and saves it to array x .

Private Member Functions

- void [QRFactorize](#) ()
Factorizes the tridiagonal A matrix stored in R to $Q \cdot R$.

Private Attributes

- [QMatrix< real_t > Q](#)
 Q matrix for QR Factorization
- [RMatrix< real_t > R](#)
 R matrix for QR Factorization
- const unsigned int **N**
Number of rows N of the matrix A .
- real_t * **b**
One dimensional array to store $N \times N$ vector b .
- unsigned int **k**
Iterator.

6.4.1 Detailed Description

```
template<typename real_t>
class QRSolver< real_t >
```

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

Template Parameters

<i>real_t</i>	
<i>_t</i>	

Definition at line 32 of file QR_Solver.hpp.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 QRSolver()

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

Construct a new [QRSolver](#) object.

Parameters

N	Size of the matrix equations
-----	------------------------------

Definition at line 20 of file QR_Solver.cpp.

6.4.3 Member Function Documentation

6.4.3.1 getSolution()

```
template<typename real_t >
void QRSolver< real_t >::getSolution (
    real_t * x )
```

Finds the solution to matrix equation and saves it to array x .

Parameters

x	Array to store the solution of the matrix equation
-----	--

Definition at line 154 of file QR_Solver.cpp.

6.4.3.2 printMatrixEquation()

```
template<typename real_t >
void QRSolver< real_t >::printMatrixEquation
```

Prints the matrix A and vector b .

To be used before QR factorization is performed in `getSolution`

Definition at line 94 of file `QR_Solver.cpp`.

6.4.3.3 printQRMatrices()

```
template<typename real_t >
void QRSolver< real_t >::printQRMatrices
```

Prints the factorized matrices $Q \cdot R$.

To be used after QR factorization is performed in `getSolution`

Definition at line 140 of file `QR_Solver.cpp`.

6.4.3.4 setEquation()

```
template<typename real_t >
void QRSolver< real_t >::setEquation (
    unsigned int index,
    real_t e,
    real_t f,
    real_t g,
    real_t b )
```

Set up equation represented by i th row of the matrix equation $ex_{i-1} + fx_i + gx_{i+1} = b$.

Parameters

<i>index</i>	<i>i</i>
<i>e</i>	Coefficient to x_{i-1}
<i>f</i>	Coefficient to x_i
<i>g</i>	Coefficient to x_{i+1}
<i>b</i>	Constant

Definition at line 39 of file `QR_Solver.cpp`.

6.4.3.5 setEquationFirstRow()

```
template<typename real_t >
void QRSolver< real_t >::setEquationFirstRow (
    real_t f,
    real_t g,
    real_t b )
```

Set up equation represented by the first row of the matrix equation $fx_i + gx_{i+1} = b$.

Parameters

f	Coefficient to x_i
g	Coefficient to x_{i+1}
b	Constant

Definition at line 60 of file QR_Solver.cpp.

6.4.3.6 setEquationLastRow()

```
template<typename real_t >
void QRSolver< real_t >::setEquationLastRow (
    real_t e,
    real_t f,
    real_t b )
```

up equation represented by the last row of the matrix equation $ex_{i-1}fx_i = b$

Parameters

e	Coefficient to x_{i-1}
f	Coefficient to x_i
b	Constant

Definition at line 77 of file QR_Solver.cpp.

6.4.4 Member Data Documentation**6.4.4.1 R**

```
template<typename real_t >
RMatrix<real_t> QRSolver< real_t >::R [private]
```

R matrix for QR Factorization

The R matrix also serves as the initial tridiagonal A matrix to save memory and more important reduce redundant memory copy operations

Definition at line 48 of file QR_Solver.hpp.

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

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

6.5 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.

6.5.1 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.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 RMatrix()

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

Construct a new R Matrix.

Parameters

<i>n</i>	Size of main diagonal of the $N \times N$ R Matrix
----------	--

Definition at line 18 of file R_Matrix.cpp.

6.5.2.2 ~RMatrix()

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

Destroy the R Matrix.

Definition at line 29 of file R_Matrix.cpp.

6.5.3 Member Function Documentation

6.5.3.1 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

<i>row_index</i>	Row index i
<i>column_index</i>	Column index j

Returns

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

Definition at line 36 of file R_Matrix.cpp.

6.5.3.2 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

<i>row_index</i>	Row index i
<i>column_index</i>	Column index j

Returns

Returns the index in the the flattened array

Definition at line 109 of file R_Matrix.cpp.

6.5.3.3 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

<i>row_index</i>	
<i>column_index</i>	

Returns

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

Definition at line 125 of file R_Matrix.cpp.

6.5.3.4 print()

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

Prints the R Matrix in form of a flattened array.

Definition at line 94 of file R_Matrix.cpp.

6.5.3.5 printMatrix()

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

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

Definition at line 75 of file R_Matrix.cpp.

6.5.3.6 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

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

Definition at line 54 of file R_Matrix.cpp.

6.5.4 Member Data Documentation

6.5.4.1 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.

6.5.4.2 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/qrsolver/R_Matrix.hpp
- src/qrsolver/R_Matrix.cpp

6.6 Substance< real_t > Class Template Reference

Class to represent a pure substance.

```
#include <Substance.hpp>
```

Public Member Functions

- [Substance](#) (real_t density, real_t heat_capacity, real_t molecular_weight, real_t heat_conductivity, real_t standard_enthalpy_of_formation=0)
Construct a new [Substance](#).
- real_t [getDensity](#) ()
Get the Density in.
- real_t [getHeatCapacity](#) ()
Get the Heat Capacity.
- real_t [getMolecularWeight](#) ()
Get the Molecular Weight.
- real_t [getHeatConductivity](#) ()
Get the Heat Conductivity.
- real_t [getStandardEnthalpyOfFormation](#) ()
Get the Standard Enthalpy Of Formation.
- real_t [getEnthalpy](#) (real_t Temperature)
Get the Enthalpy of the substance with respect to the T_{REF} .
- void [printProperties](#) (std::ostream &output_stream)
Print the properties of the substance to the given stream.

Private Attributes

- `const real_t _density`
Density of the substance.
- `const real_t _heat_capacity`
Heat capacity of the substance.
- `const real_t _molecular_weight`
Molecular Weight of the substance.
- `const real_t _heat_conductivity`
Heat Conductivity of the substance.
- `const real_t _std_enthalpy_formation`
Standard Enthalpy of Formation of the substance.

6.6.1 Detailed Description

```
template<typename real_t>
class Substance< real_t >
```

Class to represent a pure substance.

Template Parameters

<code>real_t</code>	
---------------------	--

Definition at line 27 of file Substance.hpp.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Substance()

```
template<typename real_t >
Substance< real_t >::Substance (
    real_t density,
    real_t heat_capacity,
    real_t molecular_weight,
    real_t heat_conductivity,
    real_t standard_enthalpy_of_formation = 0 ) [inline]
```

Construct a new [Substance](#).

Parameters

<code>density</code>	Density of the substance in kg/m^3
<code>heat_capacity</code>	Heat capacity of the substance in $J/kg - K$
<code>molecular_weight</code>	Molecular weight of the substance in kg/mol
<code>heat_conductivity</code>	Heat conductivity of the substance in W/m
<code>standard_enthalpy_of_formation</code>	Standard enthalpy of formation of the substance at 298 K in J/kg

Definition at line 40 of file Substance.hpp.

6.6.3 Member Function Documentation

6.6.3.1 getDensity()

```
template<typename real_t >
real_t Substance< real_t >::getDensity ( ) [inline]
```

Get the Density in.

Returns

real_t Density of the substance

Definition at line 58 of file Substance.hpp.

6.6.3.2 getEnthalpy()

```
template<typename real_t >
real_t Substance< real_t >::getEnthalpy (
    real_t Temperature ) [inline]
```

Get the Enthalpy of the substance with respect to the T_REF.

Parameters

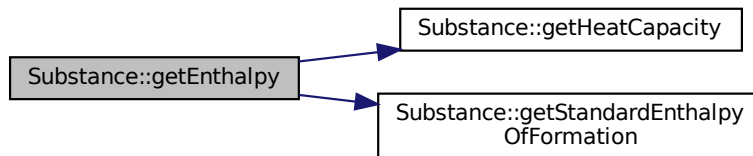
<i>Temperature</i>	Temperature at which enthalpy of the object needs to be evaluated
--------------------	---

Returns

real_t Enthalpy of the substance in J / kg

Definition at line 94 of file Substance.hpp.

Here is the call graph for this function:

**6.6.3.3 getHeatCapacity()**

```
template<typename real_t >
real_t Substance< real_t >::getHeatCapacity ( ) [inline]
```

Get the Heat Capacity.

Returns

real_t Heat Capacity of the substance

Definition at line 65 of file Substance.hpp.

6.6.3.4 getHeatConductivity()

```
template<typename real_t >
real_t Substance< real_t >::getHeatConductivity ( ) [inline]
```

Get the Heat Conductivity.

Returns

real_t Heat Conductivity of the substance

Definition at line 79 of file Substance.hpp.

6.6.3.5 getMolecularWeight()

```
template<typename real_t >
real_t Substance< real_t >::getMolecularWeight ( ) [inline]
```

Get the Molecular Weight.

Returns

real_t Molecular Weight of the substance

Definition at line 72 of file Substance.hpp.

6.6.3.6 getStandardEnthalpyOfFormation()

```
template<typename real_t >
real_t Substance< real_t >::getStandardEnthalpyOfFormation ( ) [inline]
```

Get the Standard Enthalpy Of Formation.

Returns

real_t Standard Enthalpy of Formation of the substance

Definition at line 86 of file Substance.hpp.

6.6.3.7 printProperties()

```
template<typename real_t >
void Substance< real_t >::printProperties (
    std::ostream & output_stream ) [inline]
```

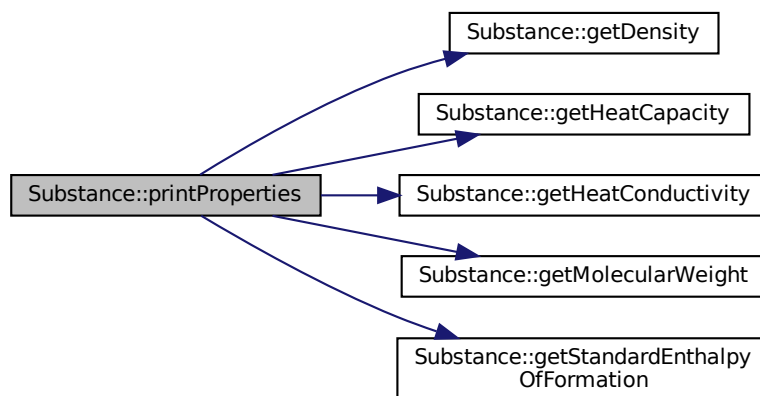
Print the properties of the substance to the given stream.

Parameters

<i>output_stream</i>	Stream to which the properties are printed
----------------------	--

Definition at line 104 of file Substance.hpp.

Here is the call graph for this function:



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

- `include/thermo-physical-properties/Substance.hpp`

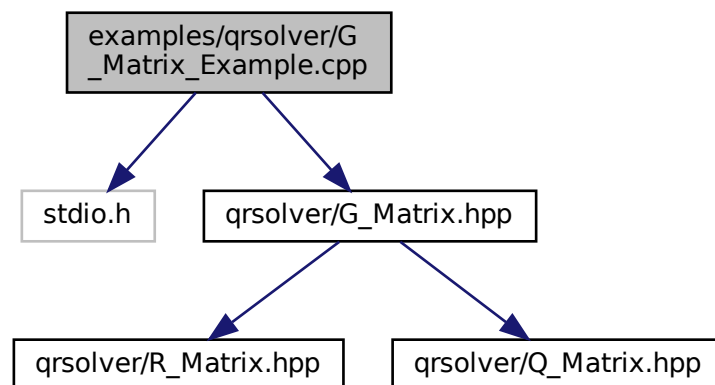
Chapter 7

File Documentation

7.1 examples/qrsolver/G_Matrix_Example.cpp File Reference

Example to test out [GMatrix](#) class.

```
#include <stdio.h>
#include "qrsolver/G_Matrix.hpp"
Include dependency graph for G_Matrix_Example.cpp:
```



Functions

- `int main (int argc, char const *argv[])`

7.1.1 Detailed Description

Example to test out [GMatrix](#) class.

Author

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

Version

0.1

Date

2021-07-01

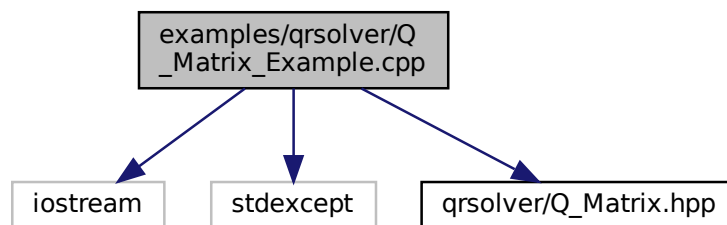
Copyright

Copyright (c) 2021

7.2 examples/qrsolver/Q_Matrix_Example.cpp File Reference

Example to test [QMatrix](#) class.

```
#include <iostream>
#include <stdexcept>
#include "qrsolver/Q_Matrix.hpp"
Include dependency graph for Q_Matrix_Example.cpp:
```



Functions

- `int main (int argc, char const *argv[])`

7.2.1 Detailed Description

Example to test [QMatrix](#) class.

Author

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

Version

0.1

Date

2021-06-27

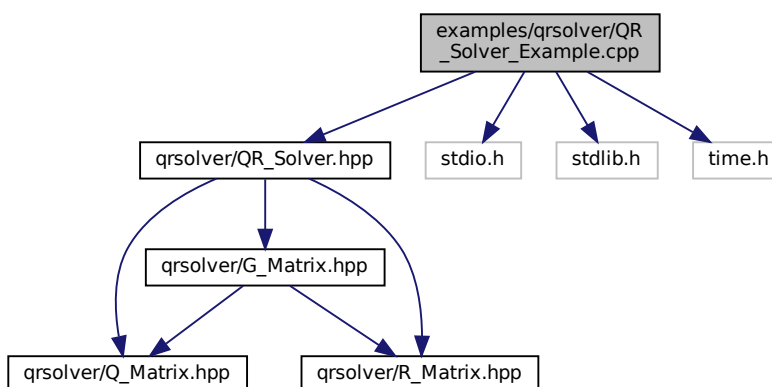
Copyright

Copyright (c) 2021

7.3 examples/qrsolver/QR_Solver_Example.cpp File Reference

Example cpp file to test out [QRSolver](#) class.

```
#include "qrsolver/QR_Solver.hpp"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
Include dependency graph for QR_Solver_Example.cpp:
```



Functions

- `int main (int argc, char const *argv[])`

7.3.1 Detailed Description

Example cpp file to test out [QRSolver](#) class.

Author

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

Version

0.1

Date

2021-06-24

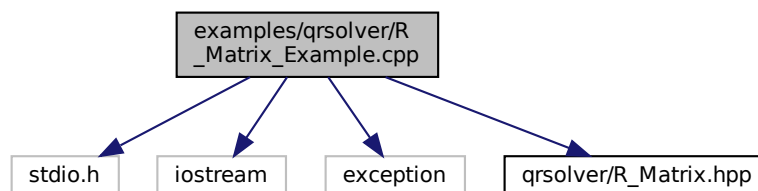
Copyright

Copyright (c) 2021

7.4 examples/qrsolver/R_Matrix_Example.cpp File Reference

An example cpp program to test the [RMatrix](#) class.

```
#include <stdio.h>
#include <iostream>
#include <exception>
#include "qrsolver/R_Matrix.hpp"
Include dependency graph for R_Matrix_Example.cpp:
```



Functions

- `int main (int argc, char const *argv[])`

7.4.1 Detailed Description

An example cpp program to test the [RMatrix](#) class.

Author

Souritra Garai (you@domain.com)

Version

0.1

Date

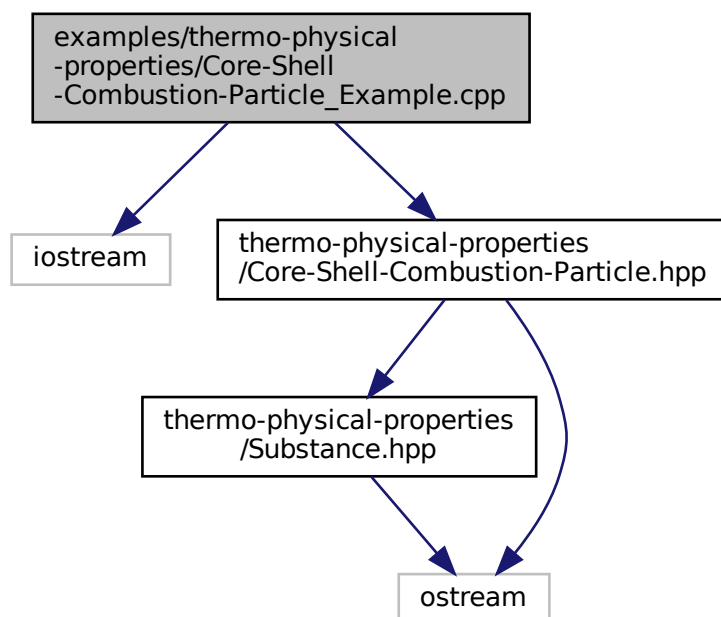
2021-06-24

Copyright

Copyright (c) 2021

7.5 examples/thermo-physical-properties/Core-Shell-Combustion-Particle_Example.cpp File Reference

```
#include <iostream>
#include "thermo-physical-properties/Core-Shell-Combustion-Particle.hpp"
Include dependency graph for Core-Shell-Combustion-Particle_Example.cpp:
```



Functions

- `int main (int argc, char const *argv[])`

Variables

- `Substance< float > A (10, 10, 10, 10)`
- `Substance< float > B (10, 10, 10, 10)`
- `Substance< float > AB (10, 10, 10, 10, -120)`
- `CoreShellCombustionParticle< float > AB_particle (A, B, AB, 10, 10)`

7.5.1 Detailed Description

Author

your name (`you@domain.com`)

Version

0.1

Date

2021-07-09

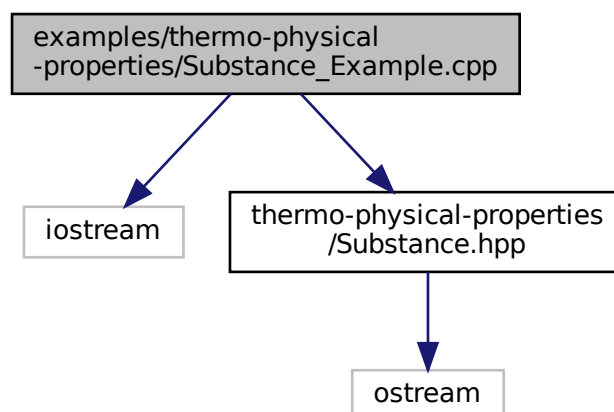
Copyright

Copyright (c) 2021

7.6 examples/thermo-physical-properties/Substance_Example.cpp File Reference

Example to test functions of `Substance` class.

```
#include <iostream>
#include "thermo-physical-properties/Substance.hpp"
Include dependency graph for Substance_Example.cpp:
```



Functions

- `int main (int argc, char const *argv[])`

Variables

- `Substance` < float > `Water` (1000, 4180, 18E-3, 10)

7.6.1 Detailed Description

Example to test functions of `Substance` class.

Author

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

Version

0.1

Date

2021-07-06

Copyright

Copyright (c) 2021

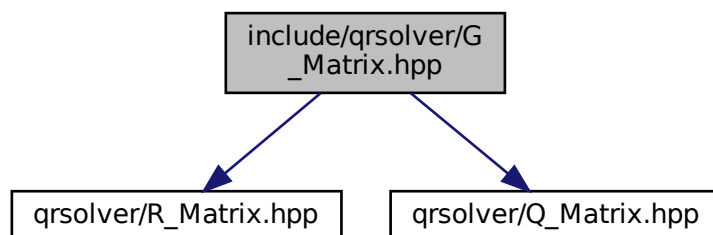
7.7 include/qrsolver/G_Matrix.hpp File Reference

This header serves the definition of an implementation of Givens' Rotation matrix. The rotation matrix is used to solve matrix equations through QR factorization method, particularly tridiagonal matrix equation.

```
#include "qrsolver/R_Matrix.hpp"
```

```
#include "qrsolver/Q_Matrix.hpp"
```

Include dependency graph for G_Matrix.hpp:



Classes

- class `GMatrix< real_t >`

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

7.7.1 Detailed Description

This header serves the definition of an implementation of Givens' Rotation matrix. The rotation matrix is used to solve matrix equations through QR factorization method, particularly tridiagonal matrix equation.

Author

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

Version

0.1

Date

2021-06-28

Copyright

Copyright (c) 2021

7.8 include/qrsolver/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 $N \times N$ Q Matrix.

7.8.1 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

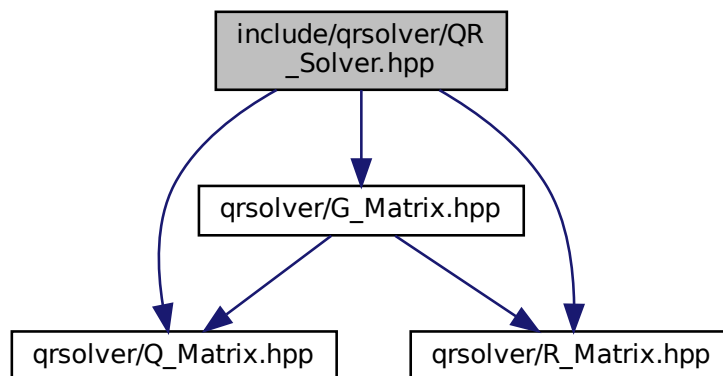
Copyright

Copyright (c) 2021

7.9 include/qrsolver/QR_Solver.hpp File Reference

This header file defines a class for solving 2D matrix equations of the form $A \cdot x = b$ (where A is an $N \times N$ matrix and x and b are $N \times 1$ vectors) using QR factorization technique.

```
#include "qrsolver/Q_Matrix.hpp"
#include "qrsolver/R_Matrix.hpp"
#include "qrsolver/G_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 $A \cdot x = b$ where A is a $N \times N$ tridiagonal matrix and x and b are $N \times 1$ vectors.

7.9.1 Detailed Description

This header file defines a class for solving 2D matrix equations of the form $A \cdot x = b$ (where A is an $N \times N$ matrix and x and b are $N \times 1$ vectors) using QR factorization technique.

Author

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

Version

0.1

Date

2021-06-23

Copyright

Copyright (c) 2021

7.10 include/qrsolver/R_Matrix.hpp File Reference

This header file defines a class for memory efficient implementation of R matrix used for QR factorisation of tridiagonal matrix.

Classes

- class `RMatrix< real_t >`

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

7.10.1 Detailed Description

This header file defines a class for memory efficient implementation of R matrix used for QR factorisation of tridiagonal matrix.

Author

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

Version

0.1

Date

2021-06-24

Copyright

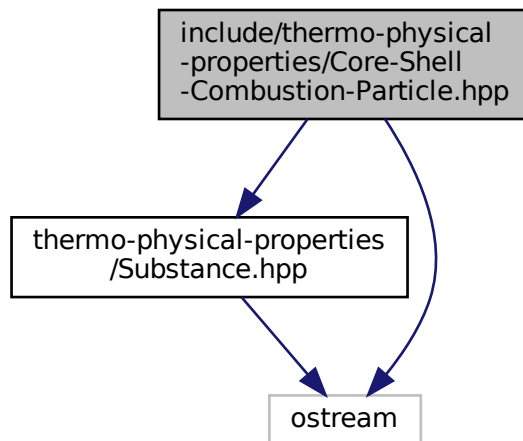
Copyright (c) 2021

7.11 include/thermo-physical-properties/Core-Shell-Combustion-Particle.hpp File Reference

This header defines a class for core shell particle.

```
#include "thermo-physical-properties/Substance.hpp"
#include <ostream>
```

Include dependency graph for Core-Shell-Combustion-Particle.hpp:



Classes

- class [CoreShellCombustionParticle< real_t >](#)

Class to represent Core-Shell Particle with functions estimate thermodynamic properties for varying composition and temperature.

7.11.1 Detailed Description

This header defines a class for core shell particle.

Author

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

Version

0.1

Date

2021-07-08

Copyright

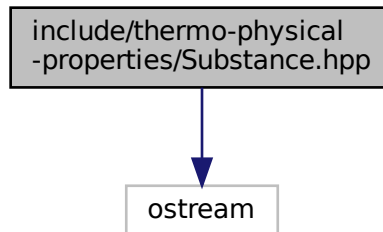
Copyright (c) 2021

7.12 include/thermo-physical-properties/Substance.hpp File Reference

This header defines a class to represent pure substances with their thermodynamic properties like density, heat capacity etc.

```
#include <ostream>
```

Include dependency graph for Substance.hpp:



Classes

- class [Substance< real_t >](#)
Class to represent a pure substance.

Macros

- `#define T_REF 298.15`

7.12.1 Detailed Description

This header defines a class to represent pure substances with their thermodynamic properties like density, heat capacity etc.

Author

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

Version

0.1

Date

2021-07-06

Copyright

Copyright (c) 2021

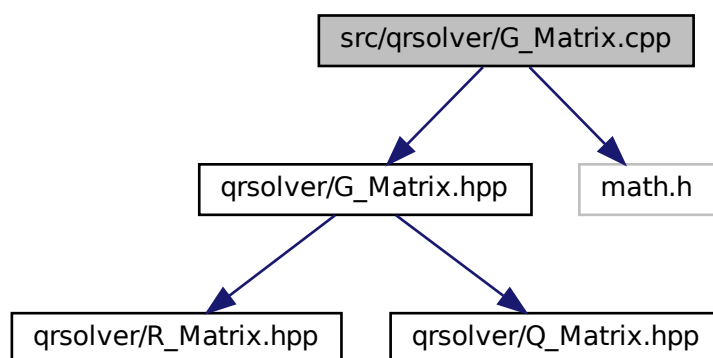
7.13 src/qrsolver/G_Matrix.cpp File Reference

Member function definitions for [GMatrix](#) class.

```
#include "qrsolver/G_Matrix.hpp"
```

```
#include <math.h>
```

Include dependency graph for G_Matrix.cpp:



7.13.1 Detailed Description

Member function definitions for [GMatrix](#) class.

Author

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

Version

0.1

Date

2021-07-01

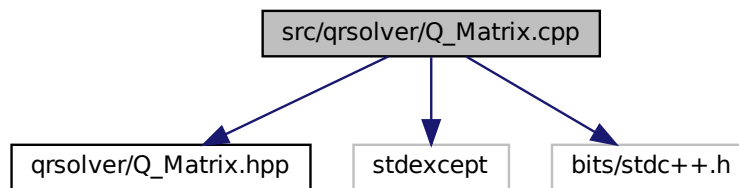
Copyright

Copyright (c) 2021

7.14 src/qrsolver/Q_Matrix.cpp File Reference

Member function definitions for [QMatrix](#) class.

```
#include "qrsolver/Q_Matrix.hpp"  
#include <stdexcept>  
#include <bits/stdc++.h>  
Include dependency graph for Q_Matrix.cpp:
```



7.14.1 Detailed Description

Member function definitions for [QMatrix](#) class.

Author

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

Version

0.1

Date

2021-06-25

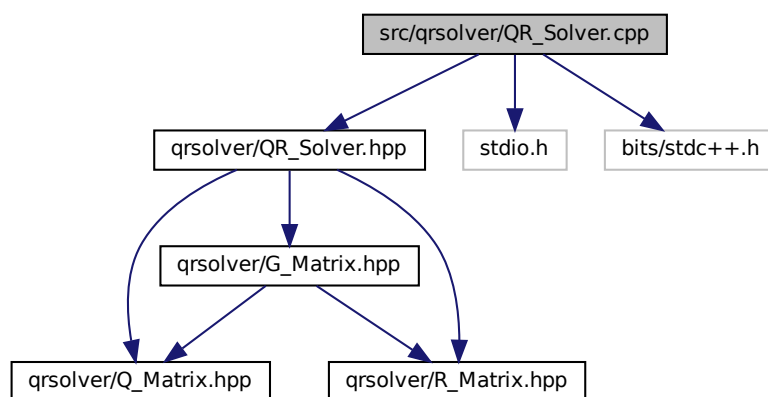
Copyright

Copyright (c) 2021

7.15 src/qrsolver/QR_Solver.cpp File Reference

Member function definitions for [QRSolver](#) class.

```
#include "qrsolver/QR_Solver.hpp"
#include <stdio.h>
#include <bits/stdc++.h>
Include dependency graph for QR_Solver.cpp:
```



7.15.1 Detailed Description

Member function definitions for [QRSolver](#) class.

Author

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

Version

0.1

Date

2021-07-01

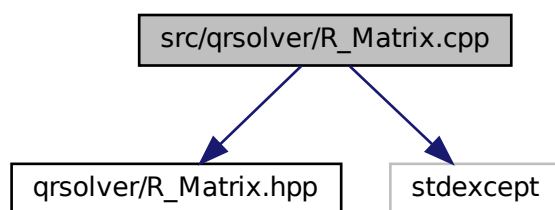
Copyright

Copyright (c) 2021

7.16 src/qrsolver/R_Matrix.cpp File Reference

Member function definitions for [RMatrix](#) class.

```
#include "qrsolver/R_Matrix.hpp"  
#include <stdexcept>  
Include dependency graph for R_Matrix.cpp:
```



7.16.1 Detailed Description

Member function definitions for [RMatrix](#) class.

Author

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

Version

0.1

Date

2021-06-24

Copyright

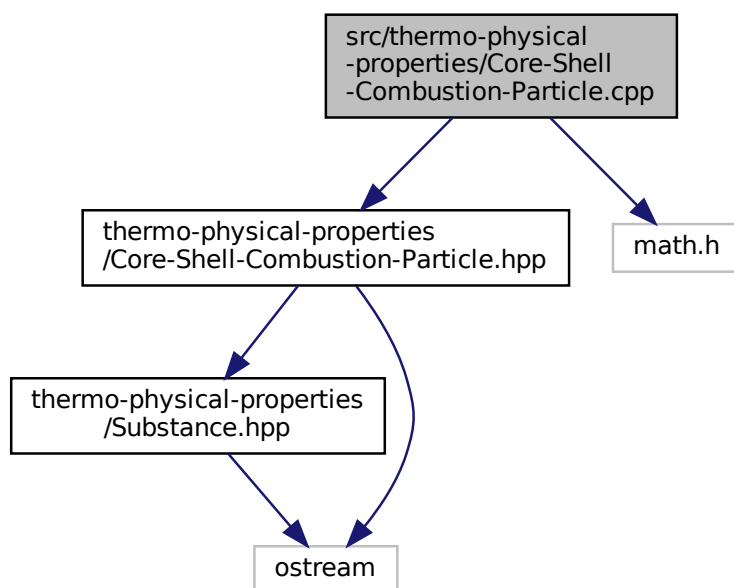
Copyright (c) 2021

7.17 src/thermo-physical-properties/Core-Shell-Combustion-Particle.cpp File Reference

```
#include "thermo-physical-properties/Core-Shell-Combustion-Particle.hpp"
```

```
#include <math.h>
```

Include dependency graph for Core-Shell-Combustion-Particle.cpp:



7.17.1 Detailed Description

Author

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

Version

0.1

Date

2021-07-08

Copyright

Copyright (c) 2021

Index

- ~RMatrix
 - RMatrix< real_t >, [28](#)
- array
 - RMatrix< real_t >, [30](#)
- CoreShellCombustionParticle
 - CoreShellCombustionParticle< real_t >, [12](#)
- CoreShellCombustionParticle< real_t >, [11](#)
 - CoreShellCombustionParticle, [12](#)
 - getDensity, [13](#)
 - getDiffusivity, [13](#)
 - getEnthalpy, [13](#)
 - getHeatCapacity, [14](#)
 - getHeatConductivity, [14](#)
 - printProperties, [14](#)
- examples/qrsolver/G_Matrix_Example.cpp, [37](#)
- examples/qrsolver/Q_Matrix_Example.cpp, [38](#)
- examples/qrsolver/QR_Solver_Example.cpp, [39](#)
- examples/qrsolver/R_Matrix_Example.cpp, [40](#)
- examples/thermo-physical-properties/Core-Shell-Combustion-Particle_Example.cpp, [41](#)
- examples/thermo-physical-properties/Substance_Example.cpp, [42](#)
- getDensity
 - CoreShellCombustionParticle< real_t >, [13](#)
 - Substance< real_t >, [33](#)
- getDiffusivity
 - CoreShellCombustionParticle< real_t >, [13](#)
- getElement
 - QMatrix< real_t >, [20](#)
 - RMatrix< real_t >, [28](#)
- getEnthalpy
 - CoreShellCombustionParticle< real_t >, [13](#)
 - Substance< real_t >, [33](#)
- getHeatCapacity
 - CoreShellCombustionParticle< real_t >, [14](#)
 - Substance< real_t >, [34](#)
- getHeatConductivity
 - CoreShellCombustionParticle< real_t >, [14](#)
 - Substance< real_t >, [34](#)
- getIndex
 - QMatrix< real_t >, [21](#)
 - RMatrix< real_t >, [29](#)
- getMolecularWeight
 - Substance< real_t >, [34](#)
- getSolution
 - QRSolver< real_t >, [24](#)
- getStandardEnthalpyOfFormation
 - Substance< real_t >, [35](#)
- GMatrix
 - GMatrix< real_t >, [16](#)
- GMatrix< real_t >, [15](#)
 - GMatrix, [16](#)
 - multiply, [17](#)
 - multiplyLastRow, [18](#)
- include/qrsolver/G_Matrix.hpp, [43](#)
- include/qrsolver/Q_Matrix.hpp, [44](#)
- include/qrsolver/QR_Solver.hpp, [45](#)
- include/qrsolver/R_Matrix.hpp, [46](#)
- include/thermo-physical-properties/Core-Shell-Combustion-Particle.hpp, [46](#)
- include/thermo-physical-properties/Substance.hpp, [48](#)
- indexOfZeroElement
 - QMatrix< real_t >, [21](#)
 - RMatrix< real_t >, [29](#)
- multiply
 - GMatrix< real_t >, [17](#)
 - QMatrix< real_t >, [21](#)
- multiplyLastRow
 - GMatrix< real_t >, [18](#)
- N
 - RMatrix< real_t >, [31](#)
- print
 - RMatrix< real_t >, [29](#)
- printMatrix
 - RMatrix< real_t >, [30](#)
- printMatrixEquation
 - QRSolver< real_t >, [24](#)
- printProperties
 - CoreShellCombustionParticle< real_t >, [14](#)
 - Substance< real_t >, [35](#)
- printQRMatrices
 - QRSolver< real_t >, [25](#)
- QMatrix
 - QMatrix< real_t >, [20](#)
- QMatrix< real_t >, [19](#)
 - getElement, [20](#)
 - getIndex, [21](#)
 - indexOfZeroElement, [21](#)
 - multiply, [21](#)
 - QMatrix, [20](#)
 - setElement, [22](#)
- QR_Factorization.py, [9](#)

QRSolver

QRSolver< real_t >, 24

QRSolver< real_t >, 22

getSolution, 24

printMatrixEquation, 24

printQRMatrices, 25

QRSolver, 24

R, 26

setEquation, 25

setEquationFirstRow, 25

setEquationLastRow, 26

R

QRSolver< real_t >, 26

RMatrix

RMatrix< real_t >, 28

RMatrix< real_t >, 27

~RMatrix, 28

array, 30

getElement, 28

getIndex, 29

indexOfZeroElement, 29

N, 31

print, 29

printMatrix, 30

RMatrix, 28

setElement, 30

setElement

QMatrix< real_t >, 22

RMatrix< real_t >, 30

setEquation

QRSolver< real_t >, 25

setEquationFirstRow

QRSolver< real_t >, 25

setEquationLastRow

QRSolver< real_t >, 26

src/qrsolver/G_Matrix.cpp, 49

src/qrsolver/Q_Matrix.cpp, 50

src/qrsolver/QR_Solver.cpp, 51

src/qrsolver/R_Matrix.cpp, 52

src/thermo-physical-properties/Core-Shell-Combustion-Particle.cpp, 53

Substance

Substance< real_t >, 32

Substance< real_t >, 31

getDensity, 33

getEnthalpy, 33

getHeatCapacity, 34

getHeatConductivity, 34

getMolecularWeight, 34

getStandardEnthalpyOfFormation, 35

printProperties, 35

Substance, 32