

Fluidos

2.0.0

Gerado por Doxygen 1.8.1.2

Terça, 1 de Janeiro de 2013 22:01:01

Sumário

1	Índice dos Componentes	1
1.1	Lista de Componentes	1
2	Índice dos Arquivos	3
2.1	Lista de Arquivos	3
3	Classes	5
3.1	Referência da Classe MatrixFilesIO	5
3.1.1	Construtores & Destrutores	5
3.1.1.1	MatrixFilesIO	5
3.1.1.2	~MatrixFilesIO	5
3.1.2	Métodos	5
3.1.2.1	CreateAndSaveMFile	5
3.1.2.2	OperationIO	5
3.1.2.3	OperationIO	6
3.1.2.4	ShowMatrix	6
3.2	Referência da Classe NavierStokes	6
3.2.1	Construtores & Destrutores	6
3.2.1.1	NavierStokes	6
3.3	Referência da Classe Poisson	6
3.3.1	Construtores & Destrutores	6
3.3.1.1	Poisson	6
3.3.1.2	~Poisson	6
3.3.1.3	Poisson	6
3.3.2	Métodos	7
3.3.2.1	PoissonDirichlet	7
3.3.2.2	PoissonNeumann	7
4	Arquivos	9
4.1	Referência do Arquivo Link to edp_01.cpp	9
4.1.1	Funções	9
4.1.1.1	main	9

4.2	Referência do Arquivo Link to edp_05.cpp	9
4.2.1	Funções	9
4.2.1.1	main	9
4.3	Referência do Arquivo Link to main.cpp	9
4.3.1	Funções	10
4.3.1.1	main	10
4.4	Referência do Arquivo Link to MatrixFilesIO.cpp	10
4.4.1	Funções	10
4.4.1.1	CreateAndSaveMFile	10
4.5	Referência do Arquivo Link to MatrixFilesIO.hpp	10
4.5.1	Definições e macros	10
4.5.1.1	CHECK_AND_CLEAN_POINTER_DOUBLE	10
4.5.1.2	CHECK_AND_CLOSE_OPENED_FILES	11
4.5.1.3	CHECK_READ_SYSTEM_BINARY_OR_ASCII	11
4.5.1.4	CHECK_WRITE_SYSTEM_BINARY_OR_ASCII	11
4.5.1.5	CONVERT_MATRIX_FROM_EIGEN_TO_DOUBLE_POINTER	11
4.5.1.6	ERROR_MESSAGE_WRONG_PARAMETERS	11
4.5.1.7	INITIALIZE_FILE_READ_BINARY_OR_ASCII	11
4.5.1.8	INITIALIZE_FILE_WRITE_BINARY_OR_ASCII	12
4.5.1.9	READ_DOUBLE_AND_CONVERT_TO_EIGEN_MATRIX	12
4.5.1.10	UPDATE_NUMBER_OF_MATRICES_READ_OR_WRITTEN	12
4.5.1.11	WARNING_WRITE_ASCII	12
4.5.1.12	WRONG_PARAMETERS	12
4.6	Referência do Arquivo Link to MatrixFilesIO_test.cpp	12
4.6.1	Funções	12
4.6.1.1	main	12
4.7	Referência do Arquivo Link to NavierStokes.cpp	12
4.8	Referência do Arquivo Link to NavierStokes.hpp	13
4.8.1	Definições e macros	13
4.8.1.1	VELOCITY_X_AVERAGE	13
4.8.1.2	VELOCITY_X_NO_PRESSURE	13
4.8.1.3	VELOCITY_X_SUM	14
4.8.1.4	VELOCITY_Y_AVERAGE	14
4.8.1.5	VELOCITY_Y_NO_PRESSURE	14
4.8.1.6	VELOCITY_Y_SUM	14
4.9	Referência do Arquivo Link to OwnMath.hpp	15
4.9.1	Funções	15
4.9.1.1	dReadConditions	15
4.9.1.2	fileName	15
4.9.1.3	PoissonDirichlet	15

4.9.1.4	save	16
4.9.1.5	saveTmEqn	16
4.10	Referência do Arquivo Link to Poisson.cpp	16
4.11	Referência do Arquivo Link to Poisson.hpp	16
4.12	Referência do Arquivo Link to Poisson_test.cpp	17
4.13	Referência do Arquivo Link to SimpleTestReadAndWrite.cpp	17
4.13.1	Funções	17
4.13.1.1	main	17
4.13.1.2	read	17
4.13.1.3	write	17
4.14	Referência do Arquivo Link to stdheader.hpp	17
4.14.1	Definições e macros	18
4.14.1.1	ASCII	18
4.14.1.2	BINARY	18
4.14.1.3	CSTDLIB_H	18
4.14.1.4	CTIME_H	18
4.14.1.5	DENSE_H	18
4.14.1.6	FSTREAM_H	18
4.14.1.7	IOSTREAM_H	18
4.14.1.8	LEXICAL_CAST_H	18
4.14.1.9	PENTADIAGONAL	18
4.14.1.10	READ	18
4.14.1.11	SPARSE_H	18
4.14.1.12	STRING_H	18
4.14.1.13	SUCCESS	18
4.14.1.14	VECTOR_H	18
4.14.1.15	WRITE	18
4.14.2	Definições dos tipos	18
4.14.2.1	SpMat	18
4.14.2.2	T	18

Capítulo 1

Índice dos Componentes

1.1 Lista de Componentes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

MatrixFilesIO	5
NavierStokes	6
Poisson	6

Capítulo 2

Índice dos Arquivos

2.1 Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Link to edp_01.cpp	9
Link to edp_05.cpp	9
Link to main.cpp	9
Link to MatrixFilesIO.cpp	10
Link to MatrixFilesIO.hpp	10
Link to MatrixFilesIO_test.cpp	12
Link to NavierStokes.cpp	12
Link to NavierStokes.hpp	13
Link to OwnMath.hpp	15
Link to Poisson.cpp	16
Link to Poisson.hpp	16
Link to Poisson_test.cpp	17
Link to SimpleTestReadAndWrite.cpp	17
Link to stdheader.hpp	17

Capítulo 3

Classes

3.1 Referência da Classe MatrixFilesIO

```
#include <Link to MatrixFilesIO.hpp>
```

Métodos Públicos

- [MatrixFilesIO](#) (int nMatrixOrder, bool bReadOrWrite, bool bBinaryOrASCII, std::string strFileName)
- virtual [~MatrixFilesIO](#) ()
- void [OperationIO](#) ()
- void [OperationIO](#) (Eigen::MatrixXd dMatrixToBeWritten)
- void [CreateAndSaveMFile](#) ()
- void [ShowMatrix](#) ()

3.1.1 Construtores & Destrutores

3.1.1.1 MatrixFilesIO::MatrixFilesIO (int nMatrixOrder, bool bReadOrWrite, bool bBinaryOrASCII, std::string strFileName)

Este é o construtor da classe de IO para matrizes, [MatrixFilesIO](#). O construtor deve ser utilizado sempre. Uma vez criado um objeto, ele só poderá executar o método específico com o qual foi construído. Vamos falar dos parâmetros então.

Parâmetros

<i>nMatrixOrder</i>	é um inteiro que pede a ordem da matriz que será lida ou salva.
<i>bReadOrWrite</i>	pode tomar os valores READ ou WRITE, que são 0 e 1 respectivamente. isso limite então se o objeto servirá para ler dados ou salvá-los.
<i>bBinaryOrASCII</i>	pode tomar os valores BINARY ou ASCII que indica se será salvo ou lido arquivos em binário double ou em caracteres ASCII.
<i>strFileName</i>	é o nome do arquivo que será criado ou lido dele.

3.1.1.2 MatrixFilesIO::~MatrixFilesIO () [virtual]

3.1.2 Métodos

3.1.2.1 void MatrixFilesIO::CreateAndSaveMFile ()

3.1.2.2 void MatrixFilesIO::OperationIO ()

3.1.2.3 void MatrixFilesIO::OperationIO (Eigen::MatrixXd *dMatrixToBeWritten*)

3.1.2.4 void MatrixFilesIO::ShowMatrix ()

Esta função mostra a matriz `m_dMatrix` que é uma matriz lida ou a última matriz salva em arquivo.

A documentação para esta classe foi gerada a partir dos seguintes arquivos:

- [Link to MatrixFilesIO.hpp](#)
- [Link to MatrixFilesIO.cpp](#)

3.2 Referência da Classe NavierStokes

Métodos Públicos

- [NavierStokes](#) (double *dMi*=1.0, double *dRho*=1.0)

3.2.1 Construtores & Destrutores

3.2.1.1 NavierStokes::NavierStokes (double *dMi* = 1.0, double *dRho* = 1.0) [inline]

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- [Link to NavierStokes.cpp](#)

3.3 Referência da Classe Poisson

```
#include <Link to Poisson.hpp>
```

Métodos Públicos

- [Poisson](#) ()
- virtual [~Poisson](#) ()
- [Poisson](#) (Eigen::MatrixXd *dBoundaryConditions*, Eigen::MatrixXd *dNonHomogeneity*)
- Eigen::MatrixXd [PoissonDirichlet](#) ()
- Eigen::MatrixXd [PoissonNeumann](#) ()

3.3.1 Construtores & Destrutores

3.3.1.1 Poisson::Poisson ()

3.3.1.2 Poisson::~~Poisson () [virtual]

3.3.1.3 Poisson::Poisson (Eigen::MatrixXd *dBoundaryConditions*, Eigen::MatrixXd *dNonHomogeneity*)

Construtor da classe [Poisson](#) É obrigatória a entrada da matriz de condições de contorno, pode ter uma condição inicial na matriz, mas ela será desconsiderada. A matriz de não-homogeneidade é a matriz que segue a equação do lado direito. A equação em questão é a seguinte: $\nabla^2 u = g$

Parâmetros

<i>dNon-Homogeneity</i>	é matriz de não homogeneidade, a função g
<i>dBoundary-Conditions</i>	contém a condição inicial e de contorno

É importante que as matrizes sejam do mesmo tamanho. Caso contrário o programa será finalizado, com erro. Outro detalhe, as matrizes devem ser quadradas!

3.3.2 Métodos

3.3.2.1 `Eigen::MatrixXd Poisson::PoissonDirichlet ()`

3.3.2.2 `Eigen::MatrixXd Poisson::PoissonNeumann ()`

A documentação para esta classe foi gerada a partir dos seguintes arquivos:

- [Link to Poisson.hpp](#)
- [Link to Poisson.cpp](#)

Capítulo 4

Arquivos

4.1 Referência do Arquivo Link to edp_01.cpp

```
#include <OwnMath.hpp>
```

Funções

- int `main` ()

4.1.1 Funções

4.1.1.1 int main ()

4.2 Referência do Arquivo Link to edp_05.cpp

```
#include <stdheader.hpp>
```

Funções

- int `main` ()

4.2.1 Funções

4.2.1.1 int main ()

4.3 Referência do Arquivo Link to main.cpp

```
#include <stdheader.hpp>
```

Funções

- int `main` ()

4.3.1 Funções

4.3.1.1 int main ()

4.4 Referência do Arquivo Link to MatrixFilesIO.cpp

```
#include "../include/stdheader.hpp"
#include "../include/MatrixFilesIO.hpp"
```

Funções

- void [CreateAndSaveMFile](#) ()

4.4.1 Funções

4.4.1.1 void CreateAndSaveMFile ()

4.5 Referência do Arquivo Link to MatrixFilesIO.hpp

```
#include <stdheader.hpp>
```

Componentes

- class [MatrixFilesIO](#)

Definições e Macros

- #define [INITIALIZE_FILE_READ_BINARY_OR_ASCII](#)
- #define [INITIALIZE_FILE_WRITE_BINARY_OR_ASCII](#)
- #define [ERROR_MESSAGE_WRONG_PARAMETERS](#)
- #define [WARNING_WRITE_ASCII](#)
- #define [WRONG_PARAMETERS](#) (m_bReadOrWrite!=bReadOrWrite) || (m_bBinaryOrASCII!=bBinaryOrASCII)
- #define [CHECK_AND_CLEAN_POINTER_DOUBLE](#)
- #define [CONVERT_MATRIX_FROM_EIGEN_TO_DOUBLE_POINTER](#)
- #define [CHECK_AND_CLOSE_OPENED_FILES](#)
- #define [READ_DOUBLE_AND_CONVERT_TO_EIGEN_MATRIX](#)
- #define [CHECK_WRITE_SYSTEM_BINARY_OR_ASCII](#)
- #define [CHECK_READ_SYSTEM_BINARY_OR_ASCII](#)
- #define [UPDATE_NUMBER_OF_MATRICES_READ_OR_WRITTEN](#) m_nNumberOfMatrices++;

4.5.1 Definições e macros

4.5.1.1 #define CHECK_AND_CLEAN_POINTER_DOUBLE

Valor:

```
if (m_pdConvertedMatrix!=NULL) delete m_pdConvertedMatrix;\n    m_pdConvertedMatrix = NULL;
```


4.5.1.2 #define CHECK_AND_CLOSE_OPENED_FILES

Valor:

```
if (m_ReadFromFile.is_open()) m_ReadFromFile.close();\
    else if (m_WriteToFile.is_open()) m_WriteToFile.close();
```

4.5.1.3 #define CHECK_READ_SYSTEM_BINARY_OR_ASCII

Valor:

```
if (!m_bBinaryOrASCII) ReadBinary();\
    else ReadASCII();
```

4.5.1.4 #define CHECK_WRITE_SYSTEM_BINARY_OR_ASCII

Valor:

```
if (!m_bBinaryOrASCII) WriteBinary(dMatrixToBeWritten);\
    else WriteASCII(dMatrixToBeWritten);
```

4.5.1.5 #define CONVERT_MATRIX_FROM_EIGEN_TO_DOUBLE_POINTER

Valor:

```
m_pdConvertedMatrix = new double[m_nMatrixOrder*m_nMatrixOrder];\
    for (int i=0; i<m_nMatrixOrder; i++){\
        for (int j=0; j<m_nMatrixOrder; j++){\
            m_pdConvertedMatrix[i*m_nMatrixOrder+j]=\
                dMatrixToBeConverted(i, j);\
        }\
    }
```

4.5.1.6 #define ERROR_MESSAGE_WRONG_PARAMETERS

Valor:

```
std::cerr << "Wrong parameters given, method can only execute ";\
    if (bReadOrWrite) std::cerr << "WRITE and ";\
    else std::cerr << "READ and ";\
    \
    if (bBinaryOrASCII) std::cerr << "ASCII operations.\n";\
    else std::cerr << "Binary operations.\n";
```

4.5.1.7 #define INITIALIZE_FILE_READ_BINARY_OR_ASCII

Valor:

```
switch (bBinaryOrASCII){\
    case (BINARY):\
        m_ReadFromFile.open(strFileName.c_str(), std::ios::\
            binary);\
        break;\
    case (ASCII):\
        m_ReadFromFile.open(strFileName.c_str());\
        break;\
}
```

4.5.1.8 #define INITIALIZE_FILE_WRITE_BINARY_OR_ASCII

Valor:

```
switch (bBinaryOrASCII) {\n
    case (BINARY): \n
        m_WriteToFile.open(strFileName.c_str(), std::ios::\n
            binary); \n
        break; \n
    case (ASCII): \n
        m_WriteToFile.open(strFileName.c_str()); \n
        break; \n
}
```

4.5.1.9 #define READ_DOUBLE_AND_CONVERT_TO_EIGEN_MATRIX

Valor:

```
m_pdConvertedMatrix = new double[m_nMatrixOrder*m_nMatrixOrder]; \n
    m_ReadFromFile.read((char*)m_pdConvertedMatrix, sizeof(double)*\n
        m_nMatrixOrder*m_nMatrixOrder); \n
    m_dMatrix = Eigen::Map<Eigen::MatrixXd> (m_pdConvertedMatrix,\n
        m_nMatrixOrder,m_nMatrixOrder);
```

4.5.1.10 #define UPDATE_NUMBER_OF_MATRICES_READ_OR_WRITTEN m_nNumberOfMatrices++;

4.5.1.11 #define WARNING_WRITE_ASCII

Valor:

```
std::cerr << "Warning, given matrix order is different than real.\n"; \n
    std::cerr << "This can cause problems to reading, but none for\n
        writing in an ASCII file.\n";
```

4.5.1.12 #define WRONG_PARAMETERS (m_bReadOrWrite!=bReadOrWrite) || (m_bBinaryOrASCII!=bBinaryOrASCII)

4.6 Referência do Arquivo Link to MatrixFilesIO_test.cpp

```
#include <MatrixFilesIO.hpp>
```

Funções

- int [main](#) ()

4.6.1 Funções

4.6.1.1 int main ()

4.7 Referência do Arquivo Link to NavierStokes.cpp

```
#include <stdheader.hpp>\n
#include <NavierStokes.hpp>
```

Componentes

- class [NavierStokes](#)

4.8 Referência do Arquivo Link to NavierStokes.hpp

Definições e Macros

- #define [VELOCITY_X_SUM](#)
- #define [VELOCITY_Y_SUM](#)
- #define [VELOCITY_X_AVERAGE](#)
- #define [VELOCITY_Y_AVERAGE](#)
- #define [VELOCITY_X_NO_PRESSURE](#)
- #define [VELOCITY_Y_NO_PRESSURE](#)

4.8.1 Definições e macros

4.8.1.1 #define VELOCITY_X_AVERAGE

Valor:

```
0.25*(dVelocityX(i,j)[nTime]+dVelocityX(i-1,j)[nTime]\
+dVelocityX(i,j+1)[nTime]+dVelocityX\
(i-1,j-1)[nTime])
```

This part stands for the average of velocities in the X axis.

$$\mathbf{v}^t = (u^t, v^t)$$

The 't' stands for nothing special

$$u_{ij}^t = \frac{1}{4}(u_{ij} + u_{i-1j} + u_{i-1j-1} + u_{ij+1})$$

4.8.1.2 #define VELOCITY_X_NO_PRESSURE

Valor:

```
(m_dNu*(dVelocityXSum-4*dVelocityX(i,j)[nTime])/(dDeltaX*dDeltaX)+
dExternalForceX(i,j)/m_dRho\
-dVelocityX(i,j)*(dVelocityX(i+1,j)[
nTime]-dVelocityX(i-1,j)[nTime])/(2*dDeltaX)\
-dVelocityYAverage*(dVelocityX(i,j+1)[
nTime]-dVelocityX(i,j-1)[nTime])/(2*dDeltaX))*dDeltaT\
+dVelocityX(i,j)[nTime]
```

This part stand for the velocity obtained by Navier Stokes equation without considering the pressure

$$u_{ij}^* = \left[\frac{\mu}{\rho} \left(\frac{u_{ij}^s - 4u_{ij}}{\Delta x^2} \right) + \frac{f_{x,ij}}{\rho} - u_{ij} \frac{u_{i+1j} - u_{i-1j}}{2\Delta x} - v_{ij}^t \frac{u_{ij+1} - u_{ij-1}}{2\Delta x} \right] \Delta t + u_{ij}$$

4.8.1.3 #define VELOCITY_X_SUM

Valor:

```
dVelocityX(i+1,j)[nTime]+dVelocityX(i-1,j)[nTime]\
+dVelocityX(i,j+1)[nTime]+dVelocityX(i,j
-1)[nTime]
```

This part stands for the sum of velocities in the X axis.

$$\mathbf{v}^s = (u^s, v^s)$$

The 's' stands for 'sum'

$$u_{ij}^s = u_{i+1j} + u_{i-1j} + u_{ij+1} + u_{ij-1}$$

4.8.1.4 #define VELOCITY_Y_AVERAGE

Valor:

```
0.25*(dVelocityY(i,j)[nTime]+dVelocityY(i-1,j)[nTime]\
+dVelocityY(i,j+1)[nTime]+dVelocityY
(i-1,j-1)[nTime])
```

This part stands for the average of velocities in the X axis.

$$\mathbf{v}^t = (u^t, v^t)$$

The 't' stands for nothing special

$$v_{ij}^t = \frac{1}{4}(v_{ij} + v_{i-1j} + v_{i-1j-1} + v_{ij+1})$$

4.8.1.5 #define VELOCITY_Y_NO_PRESSURE

Valor:

```
(m_dNu*(dVelocityYSum-4*dVelocityY(i,j)[nTime])/(dDeltaX*dDeltaX)+
dExternalForceY(i,j)/m_dRho\
-dVelocityYAverage*(dVelocityY(i+1,j)[
nTime]-dVelocityY(i-1,j)[nTime])/(2*dDeltaX)\
-dVelocityY(i,j)[nTime]*(dVelocityY(i,
j+1)[nTime]-dVelocityY(i,j-1)[nTime])/(2*dDeltaX))*dDeltaT\
+dVelocityY(i,j)[nTime]
```

This part stand for the velocity obtained by Navier Stokes equation without considering the pressure

$$v_{ij}^* = \left[\frac{\mu}{\rho} \left(\frac{v_{ij}^s - 4v_{ij}}{\Delta x^2} \right) + \frac{f_{y,ij}}{\rho} - u_{ij}^t \frac{v_{i+1j} - v_{i-1j}}{2\Delta x} - v_{ij} \frac{v_{ij+1} - v_{ij-1}}{2\Delta x} \right] \Delta t + v_{ij}$$

4.8.1.6 #define VELOCITY_Y_SUM

Valor:

```
dVelocityY(i+1,j)[nTime]+dVelocityY(i-1,j)[nTime]\
+dVelocityY(i,j+1)[nTime]+dVelocityY(i,
j-1)[nTime]
```

This part stands for the sum of velocities in the Y axis.

$$\mathbf{v}^s = (u^s, v^s)$$

The 's' stands for 'sum'

$$v_{ij}^s = v_{i+1j} + v_{i-1j} + v_{ij+1} + v_{ij-1}$$

4.9 Referência do Arquivo Link to OwnMath.hpp

```
#include <stdheader.hpp>
```

Funções

- dMatrix [dReadConditions](#) (char strFile[])
- dMatrix [PoissonDirichlet](#) (dMatrix dNonHomogeneity, dMatrix dBoundaryConditions)
- void [fileName](#) (char *file)
- void [save](#) (char file[], clock_t tStart, time_t *inicio, MatrixXd U, string equation)
- void [saveTmEqn](#) (char file[], clock_t tStart, time_t *inicio, string equation, MatrixXd *U, int N)

4.9.1 Funções

4.9.1.1 dMatrix dReadConditions (char strFile[])

The method is made to read a matrix of doubles from an ASCII file It requires the number of columns and lines It will check if the read matrix is of right size

Parâmetros

<i>strFile[]</i>	is the name of text file
------------------	--------------------------

Retorna

dMatrixRead is returned, with the contents of the file

4.9.1.2 void fileName (char * file)

Method for creating a file name according to matrix size

Matrix size is not a parameter of function, it is set as a global variable

Parâmetros

<i>strFile</i>	is a char pointer where the string will be saved
----------------	--

Retorna

No return value.

4.9.1.3 dMatrix PoissonDirichlet (dMatrix dNonHomogeneity, dMatrix dBoundaryConditions)

Method to solve the poisson equation by a sparse method

Parâmetros

<i>dNon-Homogeneity</i>	is the non homogeneity in the right-hand side of poisson equation
<i>dBoundary-Conditions</i>	contains the boundary and initial conditions

Retorna

dPoissonSolution is returned

4.9.1.4 void save (char file[], clock_t tStart, time_t * inicio, MatrixXd U, string equation)

Method for saving a matrix in a file .m After save, it is possible to open easily with octave and print the graph for file

Parâmetros

<i>file[]</i>	name of file where matrix will be save
<i>tStart</i>	has the value of cpu time in the start, in this method the "tFinal" is calculated and subtracted from tStart
<i>*inicio</i>	is a pointer that indicates the time of start, and date
<i>U</i>	is the matrix that will be saved
<i>equation</i>	is a string that receives the equation being solved, just to show in the .m file. Ex: Laplace Equation - $u_{xx} + u_{yy} = f(x,y)$

Retorna

No return value.

4.9.1.5 void saveTmEqn (char file[], clock_t tStart, time_t * inicio, string equation, MatrixXd * U, int N)

Method for saving various matrices in a file .m After save, it is possible to open easily with octave and make a video of the matrices

Not implemented yet

Parâmetros

<i>file[]</i>	name of file where matrix will be save
<i>tStart</i>	has the value of cpu time in the start, in this method the "tFinal" is calculated and subtracted from tStart
<i>*inicio</i>	is a pointer that indicates the time of start, and date
<i>equation</i>	is a string that receives the equation being solved, just to show in the .m file. Ex: Laplace Equation - $u_{xx} + u_{yy} = f(x,y)$
<i>*U</i>	is a pointer to the 3d matrix U, that will be saved
<i>N</i>	indicates how many two-dimensional matrices there are in matrix U

Retorna

No return value.

4.10 Referência do Arquivo Link to Poisson.cpp

```
#include "../include/Poisson.hpp"
```

4.11 Referência do Arquivo Link to Poisson.hpp

```
#include <stdheader.hpp>
```

Componentes

- class [Poisson](#)

4.12 Referência do Arquivo Link to Poisson_test.cpp

```
#include "../include/Poisson.hpp"
```

4.13 Referência do Arquivo Link to SimpleTestReadAndWrite.cpp

```
#include <stdheader.hpp>
```

Funções

- void [write](#) (Matrix3d a, Matrix3d c)
- void [read](#) (Matrix3d *b, Matrix3d *d)
- int [main](#) ()

4.13.1 Funções

4.13.1.1 int main ()

4.13.1.2 void read (Matrix3d * b, Matrix3d * d)

4.13.1.3 void write (Matrix3d a, Matrix3d c)

4.14 Referência do Arquivo Link to stdheader.hpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <vector>
#include <boost/lexical_cast.hpp>
#include <eigen3/Eigen/Dense>
#include <eigen3/Eigen/Sparse>
```

Definições e Macros

- #define [IOSTREAM_H](#)
- #define [STRING_H](#)
- #define [CSTDLIB_H](#)
- #define [CTIME_H](#)
- #define [FSTREAM_H](#)
- #define [VECTOR_H](#)
- #define [LEXICAL_CAST_H](#)
- #define [DENSE_H](#)

- `#define SPARSE_H`
- `#define PENTADIAGONAL 5`
- `#define READ 0`
- `#define WRITE 1`
- `#define BINARY 0`
- `#define ASCII 1`
- `#define SUCCESS 1`

Definições de Tipos

- `typedef Eigen::SparseMatrix`
 `< double > SpMat`
- `typedef Eigen::Triplet< double > T`

4.14.1 Definições e macros

4.14.1.1 `#define ASCII 1`

4.14.1.2 `#define BINARY 0`

4.14.1.3 `#define CSTDLIB_H`

4.14.1.4 `#define CTIME_H`

4.14.1.5 `#define DENSE_H`

4.14.1.6 `#define FSTREAM_H`

4.14.1.7 `#define IOSTREAM_H`

4.14.1.8 `#define LEXICAL_CAST_H`

4.14.1.9 `#define PENTADIAGONAL 5`

4.14.1.10 `#define READ 0`

4.14.1.11 `#define SPARSE_H`

4.14.1.12 `#define STRING_H`

4.14.1.13 `#define SUCCESS 1`

4.14.1.14 `#define VECTOR_H`

4.14.1.15 `#define WRITE 1`

4.14.2 Definições dos tipos

4.14.2.1 `typedef Eigen::SparseMatrix<double> SpMat`

Os seguintes typedef's foram criados com o intuito de serem usados na seção de matrizes esparsas.

4.14.2.2 `typedef Eigen::Triplet<double> T`

Índice Remissivo

~MatrixFilesIO
 MatrixFilesIO, [5](#)
~Poisson
 Poisson, [6](#)

ASCII
 Link to stdheader.hpp, [18](#)

BINARY
 Link to stdheader.hpp, [18](#)

CSTDLIB_H
 Link to stdheader.hpp, [18](#)

CTIME_H
 Link to stdheader.hpp, [18](#)

CreateAndSaveMFile
 Link to MatrixFilesIO.cpp, [10](#)
 MatrixFilesIO, [5](#)

DENSE_H
 Link to stdheader.hpp, [18](#)

dReadConditions
 Link to OwnMath.hpp, [15](#)

FSTREAM_H
 Link to stdheader.hpp, [18](#)

fileName
 Link to OwnMath.hpp, [15](#)

IOSTREAM_H
 Link to stdheader.hpp, [18](#)

LEXICAL_CAST_H
 Link to stdheader.hpp, [18](#)

Link to edp_01.cpp, [9](#)
 main, [9](#)

Link to edp_05.cpp, [9](#)
 main, [9](#)

Link to main.cpp, [9](#)
 main, [10](#)

Link to MatrixFilesIO.cpp, [10](#)
 CreateAndSaveMFile, [10](#)

Link to MatrixFilesIO.hpp, [10](#)
 WRONG_PARAMETERS, [12](#)

Link to MatrixFilesIO_test.cpp, [12](#)
 main, [12](#)

Link to NavierStokes.cpp, [12](#)

Link to NavierStokes.hpp, [13](#)

 VELOCITY_X_SUM, [13](#)

 VELOCITY_Y_SUM, [14](#)

Link to OwnMath.hpp, [15](#)
 dReadConditions, [15](#)
 fileName, [15](#)
 PoissonDirichlet, [15](#)
 save, [16](#)
 saveTmEqn, [16](#)

Link to Poisson.cpp, [16](#)

Link to Poisson.hpp, [16](#)

Link to Poisson_test.cpp, [17](#)

Link to SimpleTestReadAndWrite.cpp, [17](#)
 main, [17](#)
 read, [17](#)
 write, [17](#)

Link to stdheader.hpp, [17](#)
 ASCII, [18](#)
 BINARY, [18](#)
 CSTDLIB_H, [18](#)
 CTIME_H, [18](#)
 DENSE_H, [18](#)
 FSTREAM_H, [18](#)
 IOSTREAM_H, [18](#)
 LEXICAL_CAST_H, [18](#)
 PENTADIAGONAL, [18](#)
 READ, [18](#)
 SPARSE_H, [18](#)
 STRING_H, [18](#)
 SUCCESS, [18](#)
 SpMat, [18](#)
 T, [18](#)
 VECTOR_H, [18](#)
 WRITE, [18](#)

main

 Link to edp_01.cpp, [9](#)

 Link to edp_05.cpp, [9](#)

 Link to main.cpp, [10](#)

 Link to MatrixFilesIO_test.cpp, [12](#)

 Link to SimpleTestReadAndWrite.cpp, [17](#)

MatrixFilesIO, [5](#)

 ~MatrixFilesIO, [5](#)

 CreateAndSaveMFile, [5](#)

 MatrixFilesIO, [5](#)

 MatrixFilesIO, [5](#)

 OperationIO, [5](#)

 ShowMatrix, [6](#)

NavierStokes, [6](#)

 NavierStokes, [6](#)

 NavierStokes, [6](#)

OperationIO

MatrixFilesIO, [5](#)

PENTADIAGONAL

Link to stdheader.hpp, [18](#)

Poisson, [6](#)

~Poisson, [6](#)

Poisson, [6](#)

PoissonDirichlet, [7](#)

PoissonNeumann, [7](#)

PoissonDirichlet

Link to OwnMath.hpp, [15](#)

Poisson, [7](#)

PoissonNeumann

Poisson, [7](#)

READ

Link to stdheader.hpp, [18](#)

read

Link to SimpleTestReadAndWrite.cpp, [17](#)

SPARSE_H

Link to stdheader.hpp, [18](#)

STRING_H

Link to stdheader.hpp, [18](#)

SUCCESS

Link to stdheader.hpp, [18](#)

save

Link to OwnMath.hpp, [16](#)

saveTmEqn

Link to OwnMath.hpp, [16](#)

ShowMatrix

MatrixFilesIO, [6](#)

SpMat

Link to stdheader.hpp, [18](#)

T

Link to stdheader.hpp, [18](#)

VECTOR_H

Link to stdheader.hpp, [18](#)

VELOCITY_X_AVERAGE

Link to NavierStokes.hpp, [13](#)

VELOCITY_X_SUM

Link to NavierStokes.hpp, [13](#)

VELOCITY_Y_AVERAGE

Link to NavierStokes.hpp, [14](#)

VELOCITY_Y_SUM

Link to NavierStokes.hpp, [14](#)

WRITE

Link to stdheader.hpp, [18](#)

WRONG_PARAMETERS

Link to MatrixFilesIO.hpp, [12](#)

write

Link to SimpleTestReadAndWrite.cpp, [17](#)