

Systems of linear equations

Wygenerowano przez Doxygen 1.8.17

1 Indeks hierarchiczny	1
1.1 Hierarchia klas	1
2 Indeks klas	3
2.1 Lista klas	3
3 Indeks plików	5
3.1 Lista plików	5
4 Dokumentacja klas	7
4.1 Dokumentacja klasy Matrix	7
4.1.1 Opis szczegółowy	8
4.1.2 Dokumentacja konstruktora i destruktora	8
4.1.2.1 Matrix() [1/3]	8
4.1.2.2 Matrix() [2/3]	8
4.1.2.3 Matrix() [3/3]	8
4.1.2.4 ~Matrix()	8
4.1.3 Dokumentacja funkcji składowych	9
4.1.3.1 add_d_x()	9
4.1.3.2 add_d_y()	9
4.1.3.3 capacity_x()	9
4.1.3.4 capacity_y()	9
4.1.3.5 operator*() [1/2]	10
4.1.3.6 operator*() [2/2]	10
4.1.3.7 operator*=()	10
4.1.3.8 operator+()	11
4.1.3.9 operator+=()	11
4.1.3.10 operator=() [1/2]	11
4.1.3.11 operator=() [2/2]	12
4.1.3.12 operator[]()	12
4.1.3.13 print() [1/3]	12
4.1.3.14 print() [2/3]	13
4.1.3.15 print() [3/3]	13
4.1.3.16 set_d()	13
4.1.3.17 size_x()	13
4.1.3.18 size_y()	14
4.1.3.19 swap_x()	14
4.1.3.20 swap_y()	14
4.1.4 Dokumentacja przyjaciół i funkcji związanych	15
4.1.4.1 operator*	15
4.1.4.2 operator<<	15
4.2 Dokumentacja klasy Solver	15
4.2.1 Opis szczegółowy	16

4.2.1.1 solv()	16
4.3 Dokumentacja klasy Solver_Gauss	16
4.3.1 Opis szczegółowy	17
4.3.2 Dokumentacja funkcji składowych	17
4.3.2.1 solv()	17
4.4 Dokumentacja klasy Solver_i	17
4.4.1 Opis szczegółowy	18
4.4.1.1 norm1()	18
4.4.1.2 norm2()	18
4.4.1.3 norm3()	18
4.4.1.4 solv()	19
4.5 Dokumentacja klasy Solver_Jacobi	19
4.5.1 Opis szczegółowy	19
4.5.2 Dokumentacja funkcji składowych	19
4.5.2.1 solv()	19
4.6 Dokumentacja klasy Solver_Residuum	20
4.6.1 Opis szczegółowy	20
4.6.2 Dokumentacja funkcji składowych	20
4.6.2.1 solv()	20
4.7 Dokumentacja klasy Solver_Seidl	21
4.7.1 Opis szczegółowy	21
4.7.2 Dokumentacja funkcji składowych	21
4.7.2.1 solv()	21
5 Dokumentacja plików	23
5.1 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Matrix.cpp	23
5.1.1 Dokumentacja funkcji	23
5.1.1.1 operator*()	23
5.1.1.2 operator<<()	24
5.2 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Matrix.h	24
5.3 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Projekt.cpp	24
5.4 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/SLE.h	25
5.5 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver.cpp	25
5.6 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver.h	25
5.7 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.cpp	25
5.8 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.h	25
5.9 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.cpp	26
5.10 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.h	26
5.11 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.cpp	26
5.12 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.h	26
5.13 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.cpp	26
5.14 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.h	26

Rozdział 1

Indeks hierarchiczny

1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Matrix	7
Solver	15
Solver_Gauss	16
Solver_j	17
Solver_Jacobi	19
Solver_Residuum	20
Solver_Seidl	21

Rozdział 2

Indeks klas

2.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

Matrix	< Dynamic allocated Matrix	7
Solver	< Abstrakcyjna klasa do rozwiązywania układów równa	15
Solver_Gauss	<Odpowiada za rozwiązywanie układów równaoda Gaussa	16
Solver_i	< Abstrakcyjna klasa do iteracyjnego rozwiązywania układów równa	17
Solver_Jacobi	<Odpowiada za rozwiązywanie układów równaoda Jacobiego	19
Solver_Residuum	<Odpowiada za rozwiązywanie układów równaoda minimalizacji residuów	20
Solver_Seidl	<Odpowiada za rozwiązywanie układów równaoda Gaussa-Seidla	21

Rozdział 3

Indeks plików

3.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

C:/Users/Piotrek/source/repos/Projekt/Matrix.cpp	23
C:/Users/Piotrek/source/repos/Projekt/Matrix.h	24
C:/Users/Piotrek/source/repos/Projekt/Projekt.cpp	24
C:/Users/Piotrek/source/repos/Projekt/SLE.h	25
C:/Users/Piotrek/source/repos/Projekt/Solver.cpp	25
C:/Users/Piotrek/source/repos/Projekt/Solver.h	25
C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.cpp	25
C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.h	25
C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.cpp	26
C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.h	26
C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.cpp	26
C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.h	26
C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.cpp	26
C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.h	26

Rozdział 4

Dokumentacja klas

4.1 Dokumentacja klasy Matrix

< Dynamic allocated [Matrix](#)

```
#include <Matrix.h>
```

Metody publiczne

- [Matrix](#) ()
- [Matrix](#) (int a, int b)
- [Matrix](#) ([Matrix](#) &A)
- [~Matrix](#) ()
- double * [operator\[\]](#) (int a) const
- [Matrix](#) & [operator=](#) (const [Matrix](#) &A)
- [Matrix](#) & [operator=](#) (double **a)
- [Matrix](#) [operator+](#) (const [Matrix](#) &A) const
- [Matrix](#) & [operator+=](#) ([Matrix](#) &A)
- [Matrix](#) [operator*](#) (const [Matrix](#) &A) const
- [Matrix](#) [operator*](#) (const double a) const
- [Matrix](#) [operator*="](#) (double a)
- void [swap_x](#) (int x1, int x2)
- void [swap_y](#) (int y1, int y2)
- void [add_d_x](#) (int x)
- void [set_d](#) (int y, int x)
- void [add_d_y](#) (int y)
- const void [print](#) ()
- const void [print](#) (int precision, int width)
- const void [print](#) (int precision)
- const int [size_y](#) () const
- const int [size_x](#) () const
- const int [capacity_x](#) () const
- const int [capacity_y](#) () const

Przyjaciele

- [Matrix](#) [operator*](#) (double a, [Matrix](#) A)
- std::ostream & [operator<<](#) (std::ostream &os, [Matrix](#) &A)

4.1.1 Opis szczegółowy

< Dynamic allocated [Matrix](#)

4.1.2 Dokumentacja konstruktora i destruktora

4.1.2.1 **Matrix()** [1/3]

```
Matrix::Matrix ( )
```

Konstruktor domyslny macierzy Tworzy obiekt ale nie rezerwuje na niego zadnej pamieci

4.1.2.2 **Matrix()** [2/3]

```
Matrix::Matrix (
    int a,
    int b )
```

Konstruktor - tworzy macierz o rozmiarach AxB gdzie A to liczba wierszy, a B to liczba kolumn

Parametry

in	<i>a</i>	liczba wierszy macierzy
in	<i>b</i>	liczba kolumn macierzy

4.1.2.3 **Matrix()** [3/3]

```
Matrix::Matrix (
    Matrix & A )
```

Konstruktor kopiujacy

Parametry

in	<i>A</i>	macierz za pomoca ktorej nalezy zainicjowac dana macierz
----	----------	--

4.1.2.4 **~Matrix()**

```
Matrix::~~Matrix ( )
```

Destruktor macierzy

4.1.3 Dokumentacja funkcji składowych

4.1.3.1 add_d_x()

```
void Matrix::add_d_x (
    int x )
```

Dodaje x kolumn do macierzy

Parametry

in	x	Liczba kolumn ktore chcemy dodac
----	---	----------------------------------

4.1.3.2 add_d_y()

```
void Matrix::add_d_y (
    int y )
```

Dodaje y wierszy do macierzy

Parametry

in	y	Liczba wierszy ktore chcemy dodac
----	---	-----------------------------------

4.1.3.3 capacity_x()

```
const int Matrix::capacity_x ( ) const
```

Zwraca wartosc x pojemnosci macierzy

Zwraca

pojemnosc x macierzy

4.1.3.4 capacity_y()

```
const int Matrix::capacity_y ( ) const
```

Zwraca wartosc y pojemnosci macierzy

Zwraca

pojemnosc y macierzy

4.1.3.5 operator*() [1/2]

```
Matrix Matrix::operator* (
    const double a ) const
```

Operator mnożenia macierzy przez skalar

Parametry

in	a	skalar przez który ma być pomnożona macierz
----	---	---

Zwraca

macierz iloczynu

4.1.3.6 operator*() [2/2]

```
Matrix Matrix::operator* (
    const Matrix & A ) const
```

Operator mnożenia macierzy

Parametry

in	A	prawostronny czynnik iloczynu macierzy
----	---	--

Zwraca

macierz iloczynu

4.1.3.7 operator*=()

```
Matrix Matrix::operator*= (
    double a )
```

Operator mnożenia macierzy przez skalar i przypisania

Parametry

in	a	skalar przez który ma być pomnożona macierz
----	---	---

Zwraca

macierz iloczynu

4.1.3.8 operator+()

```
Matrix Matrix::operator+ (
    const Matrix & A ) const
```

Operator dodawania macierzy

Parametry

in	<i>skladnik</i>	sumy
----	-----------------	------

Zwraca

macierz sumy

4.1.3.9 operator+=()

```
Matrix & Matrix::operator+= (
    Matrix & A )
```

Operator dodawania i przypisania macierzy

Parametry

in	<i>A</i>	skladnik sumy
----	----------	---------------

Zwraca

*this

4.1.3.10 operator=() [1/2]

```
Matrix & Matrix::operator= (
    const Matrix & A )
```

Operator przepisania macierzy

Parametry

in	<i>A</i>	macierz ktora chcemy przepisac
----	----------	--------------------------------

Zwraca

*this

4.1.3.11 operator=() [2/2]

```
Matrix & Matrix::operator= (
    double ** a )
```

Operator przepisania tablicy do macierzy. Zakladam, ze tablica ma rozmiar taki jak aktualnie ma macierz

Parametry

in	<i>a</i>	tablica zmiennych typu double, nalezy uwazac na rozmiar tablicy i macierzy
----	----------	--

Zwraca

*this

4.1.3.12 operator[]()

```
double * Matrix::operator[] (
    int a ) const
```

Operator pozwalajacy na dostep do konkretnych komorek macierzy. Jesli macierz ma jedna wiersz lub kolumne to nalezy odwolac sie do niej - np `A[0][n]`

Parametry

in	<i>a</i>	indekswiersza do ktorego chcemy sie odwolac
----	----------	---

4.1.3.13 print() [1/3]

```
const void Matrix::print ( )
```

Wyswietla macierz

4.1.3.14 print() [2/3]

```
const void Matrix::print (
    int precision )
```

Wyswietla macierz

Parametry

in	<i>precision</i>	ustawia precyzje wyswietlania liczb w macierzy
----	------------------	--

4.1.3.15 print() [3/3]

```
const void Matrix::print (
    int precision,
    int width )
```

Wyswietla macierz

Parametry

in	<i>precision</i>	ustawia precyzje wyswietlania liczb w macierzy
in	<i>width</i>	ustawia szerokosc wyswietlenia pojedynj komorki macierzy

4.1.3.16 set_d()

```
void Matrix::set_d (
    int y,
    int x )
```

Ustawia rozmiar macierzy na y na x

Parametry

in	<i>y</i>	Liczba wierszy ktore chcemy posiadac po operacji
in	<i>x</i>	Liczba kolumn ktore chcemy posiadac po operacji

4.1.3.17 size_x()

```
const int Matrix::size_x ( ) const
```

Zwraca wartosc x rozmiaru macierzy

Zwraca

rozmiar x macierzy

4.1.3.18 size_y()

```
const int Matrix::size_y ( ) const
```

Zwraca wartosc y rozmiaru macierzy

Zwraca

rozmiar y macierzy

4.1.3.19 swap_x()

```
void Matrix::swap_x (
    int x1,
    int x2 )
```

Zamienia ze soba kolumny macierzy

Parametry

in	x1	indeks kolumny ktora ma byc zamieniona
in	x2	indeks kolumny ktora ma byc zamieniona

4.1.3.20 swap_y()

```
void Matrix::swap_y (
    int y1,
    int y2 )
```

Zamienia ze soba kolumny macierzy

Parametry

in	y1	indeks wiersza ktory ma byc zamieniony
in	y2	indeks wiersza ktory ma byc zamieniony

4.1.4 Dokumentacja przyjaciół i funkcji związanych

4.1.4.1 operator*

```
Matrix operator* (
    double a,
    Matrix A ) [friend]
```

Funkcja zaprzyjzniona - operator mnożenia macierzy przez skalar

Parametry

in	a	skalar przez który ma być pomnożona macierz
----	---	---

Zwraca

macierz iloczynu

4.1.4.2 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    Matrix & A ) [friend]
```

Zaprzyjzniona funkcja operator strumieniowy

Parametry

in	os	strumietorego chcemy przeslac macierz
in	A	Macierz która przesyłamy do strumienia

Dokumentacja dla tej klasy została wygenerowana z plików:

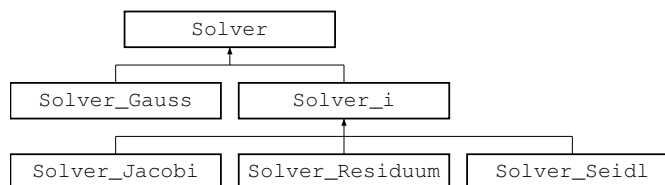
- C:/Users/Piotrek/source/repos/Projekt/[Matrix.h](#)
- C:/Users/Piotrek/source/repos/Projekt/[Matrix.cpp](#)

4.2 Dokumentacja klasy Solver

< Abstrakcyjna klasa do rozwiązywania układów równa

```
#include <Solver.h>
```

Diagram dziedziczenia dla Solver



Metody publiczne

- virtual bool `solv (Matrix A)=0`
- `Matrix results ()`

Atrybuty chronione

- `Matrix wyniki`
Macierz wynikow.

4.2.1 Opis szczegółowy

< Abstrakcyjna klasa do rozwiązywania układów równa

4.2.1.1 solv()

```
virtual bool Solver::solv (
    Matrix A ) [pure virtual]
```

wirtualna metoda rozwiązywania równa1]Parametry `in A` macierz dołączona $N \times (N+1)$ gdzie lewa część $N \times N$ jest macierza współczynników, a ostatni wiersz jest wektorem wyników równania w postaci macierzowej

Implementowany w [Solver_i](#), [Solver_Residuum](#), [Solver_Seidl](#), [Solver_Gauss](#) i [Solver_Jacobi](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

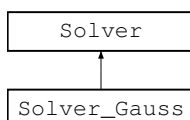
- `C:/Users/Piotrek/source/repos/Projekt/Solver.h`
- `C:/Users/Piotrek/source/repos/Projekt/Solver.cpp`

4.3 Dokumentacja klasy Solver_Gauss

<Odpowiada za rozwiązywanie układów równaoda Gaussa

```
#include <Solver_Gauss.h>
```

Diagram dziedziczenia dla Solver_Gauss



Metody publiczne

- bool `solv` (`Matrix` A)

Dodatkowe Dziedziczone Składowe

4.3.1 Opis szczegółowy

<Odpowiada za rozwiązywanie układów równań Gaussa

4.3.2 Dokumentacja funkcji składowych

4.3.2.1 solv()

```
bool Solver_Gauss::solv (
    Matrix A ) [virtual]
```

Implementacja metody rozwiązywania równań Gaussa

Parametry

in	A	macierz dołączona Nx(N+1) gdzie lewa część NxN jest macierzą współczynników, a ostatni wiersz jest wektorem wyników równania w postaci macierzowej
----	---	--

Implementuje [Solver](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

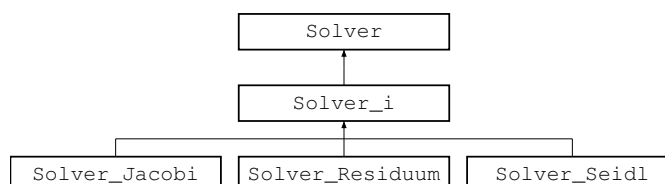
- C:/Users/Piotrek/source/repos/Projekt/[Solver_Gauss.h](#)
- C:/Users/Piotrek/source/repos/Projekt/[Solver_Gauss.cpp](#)

4.4 Dokumentacja klasy Solver_i

< Abstrakcyjna klasa do iteracyjnego rozwiązywania układów równa

```
#include <Solver.h>
```

Diagram dziedziczenia dla Solver_i



Metody publiczne

- virtual bool `solv` (`Matrix` A)=0
- void `set_precision` (double `precision`)
Ustawia precyzje generowania rozwiaznia.

Metody chronione

- bool `norm1` (`Matrix` &A)
- bool `norm2` (`Matrix` &A)
- bool `norm3` (`Matrix` &A)

Atrybuty chronione

- double `precision` = 0.001
Precyzja generowania rozwiazania (warunek stopu)

4.4.1 Opis szczegółowy

< Abstrakcyjna klasa do iteracyjnego rozwiązywania układów równa

4.4.1.1 `norm1()`

```
bool Solver_i::norm1 (  
    Matrix & A ) [protected]
```

1. Norma zbieżności metody iteracyjnej

4.4.1.2 `norm2()`

```
bool Solver_i::norm2 (  
    Matrix & A ) [protected]
```

1. Norma zbieżności metody iteracyjnej

4.4.1.3 `norm3()`

```
bool Solver_i::norm3 (  
    Matrix & A ) [protected]
```

1. Norma zbieżności metody iteracyjnej

4.4.1.4 solv()

```
virtual bool Solver_i::solv (  
    Matrix A ) [pure virtual]
```

wirtualna metoda rozwiązywania równań Parametry `in A` macierz dołączona $N \times (N+1)$ gdzie lewa część $N \times N$ jest macierzą współczynników, a ostatni wiersz jest wektorem wyników równania w postaci macierzowej

Implementuje [Solver](#).

Implementowany w [Solver_Residuum](#), [Solver_Seidl](#) i [Solver_Jacobi](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

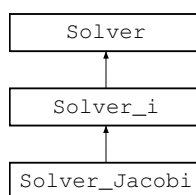
- C:/Users/Piotrek/source/repos/Projekt/[Solver.h](#)
- C:/Users/Piotrek/source/repos/Projekt/[Solver.cpp](#)

4.5 Dokumentacja klasy Solver_Jacobi

<Odpowiada za rozwiązywanie układów równań Jacobiego

```
#include <Solver_Jacobi.h>
```

Diagram dziedziczenia dla Solver_Jacobi



Metody publiczne

- bool [solv](#) ([Matrix A](#))

Dodatkowe Dziedziczone Składowe

4.5.1 Opis szczegółowy

<Odpowiada za rozwiązywanie układów równań Jacobiego

4.5.2 Dokumentacja funkcji składowych

4.5.2.1 solv()

```
bool Solver_Jacobi::solv (  
    Matrix A ) [virtual]
```

Implementacja metody rozwiązywania równań Jacobiego

Parametry

in	A	macierz dolaczona Nx(N+1) gdzie lewa czesc NxN jest macierza wspolczynnika, a ostatni wiersz jest wektorem wynikow rownania w postaci macierzowej
----	---	---

Implementuje [Solver_i](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

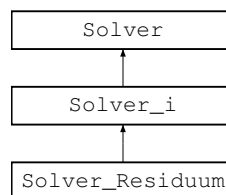
- C:/Users/Piotrek/source/repos/Projekt/[Solver_Jacobi.h](#)
- C:/Users/Piotrek/source/repos/Projekt/[Solver_Jacobi.cpp](#)

4.6 Dokumentacja klasy Solver_Residuum

<Odpowiada za rozwiązywanie układów równań minimalizacji residuów

```
#include <Solver_Residuum.h>
```

Diagram dziedziczenia dla Solver_Residuum



Metody publiczne

- bool [solv](#) ([Matrix](#) A)

Dodatkowe Dziedziczone Składowe

4.6.1 Opis szczegółowy

<Odpowiada za rozwiązywanie układów równań minimalizacji residuów

4.6.2 Dokumentacja funkcji składowych

4.6.2.1 solv()

```
bool Solver_Residuum::solv (
    Matrix A ) [virtual]
```

Implementacja metody rozwiązywania równań residuów

Parametry

in	A	macierz dolaczona Nx(N+1) gdzie lewa czesc NxN jest macierza wspolczynnkow, a ostatni wiersz jest wektorem wynikow rownania w postaci macierzowej
----	---	---

Implementuje [Solver_i](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

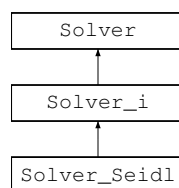
- C:/Users/Piotrek/source/repos/Projekt/[Solver_Residuum.h](#)
- C:/Users/Piotrek/source/repos/Projekt/[Solver_Residuum.cpp](#)

4.7 Dokumentacja klasy Solver_Seidl

<Odpowiada za rozwiązywanie układów równań Gaussa-Seidla

```
#include <Solver_Seidl.h>
```

Diagram dziedziczenia dla Solver_Seidl



Metody publiczne

- bool [solv](#) ([Matrix](#) A)

Dodatkowe Dziedziczone Składowe

4.7.1 Opis szczegółowy

<Odpowiada za rozwiązywanie układów równań Gaussa-Seidla

4.7.2 Dokumentacja funkcji składowych

4.7.2.1 solv()

```
bool Solver_Seidl::solv (
    Matrix A ) [virtual]
```

Implementacja metody rozwiązywania równań Gaussa-Seidla

Parametry

<code>in</code>	<code>A</code>	macierz dolaczona $N \times (N+1)$ gdzie lewa czesc $N \times N$ jest macierza wspolczynnkow, a ostatni wiersz jest wektorem wynikow rownania w postaci macierzowej
-----------------	----------------	---

Implementuje [Solver_i](#).

Dokumentacja dla tej klasy została wygenerowana z plików:

- [C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.h](#)
- [C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.cpp](#)

Rozdział 5

Dokumentacja plików

5.1 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Matrix.cpp

```
#include "Matrix.h"
#include <iostream>
#include <algorithm>
#include <iomanip>
```

Funkcje

- `Matrix operator*` (double a, `Matrix` A)
- `std::ostream & operator<<` (`std::ostream` &os, `Matrix` &A)

5.1.1 Dokumentacja funkcji

5.1.1.1 `operator*()`

```
Matrix operator* (
    double a,
    Matrix A )
```

Funkcja zaprzyjzgniona - operator mnozenia macierzy przez skalar

Parametry

in	a	skalar przez ktory ma byc pomnozona macierz
----	---	---

Zwraca

macierz iloczynu

5.1.1.2 operator<<()

```
std::ostream& operator<< (
    std::ostream & os,
    Matrix & A )
```

Zaprzyjżniona funkcja operator strumieniowy

Parametry

in	os	strumietorego chcemy przeslac macierz
in	A	Macierz ktora przesyłamy do strumienia

5.2 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Matrix.h

```
#include <iostream>
```

Komponenty

- class [Matrix](#)
< Dynamic allocated [Matrix](#)

5.3 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Projekt.cpp

```
#include <iostream>
#include <iomanip>
#include "SLE.h"
```

Funkcje

- int **main** ()

5.4 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/SLE.h

```
#include "Matrix.h"
#include "Solver.h"
#include "Solver_Gauss.h"
#include "Solver_Jacobi.h"
#include "Solver_Residuuum.h"
#include "Solver_Seidl.h"
```

5.5 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver.cpp

```
#include "Solver.h"
#include <algorithm>
```

5.6 Dokumentacja pliku C:/Users/Piotrek/source/repos/Projekt/Solver.h

```
#include "Matrix.h"
```

Komponenty

- class [Solver](#)
 - < Abstrakcyjna klasa do rozwiązywania układów równa [class Solver_i](#)
 - < Abstrakcyjna klasa do iteracyjnego rozwiązywania układów równa [DoxyCompactItemize](#)

5.7 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.cpp

```
#include "Solver_Gauss.h"
```

5.8 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.h

```
#include "Solver.h"
#include "Matrix.h"
```

Komponenty

- class [Solver_Gauss](#)
 - < Odpowiada za rozwiązywanie układów równa [oda Gaussa](#)

5.9 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.cpp

```
#include "Solver_Jacobi.h"
#include <algorithm>
#include <iostream>
```

5.10 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.h

```
#include "Solver.h"
#include "Matrix.h"
```

Komponenty

- class [Solver_Jacobi](#)
<Odpowiada za rozwiązywanie układów równań Jacobiego>

5.11 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.cpp

```
#include "Solver_Residuum.h"
#include <algorithm>
#include <iostream>
```

5.12 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.h

```
#include "Solver.h"
#include "Matrix.h"
```

Komponenty

- class [Solver_Residuum](#)
<Odpowiada za rozwiązywanie układów równań minimalizacji reszduów>

5.13 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.cpp

```
#include "Solver_Seidl.h"
#include <algorithm>
#include <iostream>
```

5.14 Dokumentacja pliku

C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.h

```
#include "Solver.h"
#include "Matrix.h"
```


Komponenty

- class `Solver_Seidl`
 <Odpowiada za rozwiązywanie układów równań Gaussa-Seidla

Indeks

- ~Matrix
 - Matrix, 8
- add_d_x
 - Matrix, 9
- add_d_y
 - Matrix, 9
- C:/Users/Piotrek/source/repos/Projekt/Matrix.cpp, 23
- C:/Users/Piotrek/source/repos/Projekt/Matrix.h, 24
- C:/Users/Piotrek/source/repos/Projekt/Projekt.cpp, 24
- C:/Users/Piotrek/source/repos/Projekt/SLE.h, 25
- C:/Users/Piotrek/source/repos/Projekt/Solver.cpp, 25
- C:/Users/Piotrek/source/repos/Projekt/Solver.h, 25
- C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.cpp, 25
- C:/Users/Piotrek/source/repos/Projekt/Solver_Gauss.h, 25
- C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.cpp, 26
- C:/Users/Piotrek/source/repos/Projekt/Solver_Jacobi.h, 26
- C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.cpp, 26
- C:/Users/Piotrek/source/repos/Projekt/Solver_Residuum.h, 26
- C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.cpp, 26
- C:/Users/Piotrek/source/repos/Projekt/Solver_Seidl.h, 26
- capacity_x
 - Matrix, 9
- capacity_y
 - Matrix, 9
- Matrix, 7
 - ~Matrix, 8
 - add_d_x, 9
 - add_d_y, 9
 - capacity_x, 9
 - capacity_y, 9
 - Matrix, 8
 - operator<<, 15
 - operator*, 9, 10, 15
 - operator*=: 10
 - operator+, 11
 - operator+=, 11
 - operator=, 11, 12
 - operator[], 12
 - print, 12, 13
 - set_d, 13
 - size_x, 13
 - size_y, 14
 - swap_x, 14
 - swap_y, 14
- Matrix.cpp
 - operator<<, 24
 - operator*, 23
- norm1
 - Solver_i, 18
- norm2
 - Solver_i, 18
- norm3
 - Solver_i, 18
- operator<<
 - Matrix, 15
 - Matrix.cpp, 24
- operator*
 - Matrix, 9, 10, 15
 - Matrix.cpp, 23
- operator*=
 - Matrix, 10
- operator+
 - Matrix, 11
- operator+=
 - Matrix, 11
- operator=
 - Matrix, 11, 12
- operator[]
 - Matrix, 12
- print
 - Matrix, 12, 13
- set_d
 - Matrix, 13
- size_x
 - Matrix, 13
- size_y
 - Matrix, 14
- solv
 - Solver, 16
 - Solver_Gauss, 17
 - Solver_i, 18
 - Solver_Jacobi, 19
 - Solver_Residuum, 20
 - Solver_Seidl, 21
- Solver, 15
 - solv, 16
- Solver_Gauss, 16
 - solv, 17
- Solver_i, 17
 - norm1, 18
 - norm2, 18
 - norm3, 18
 - solv, 18
- Solver_Jacobi, 19
 - solv, 19
- Solver_Residuum, 20
 - solv, 20
- Solver_Seidl, 21
 - solv, 21
- swap_x
 - Matrix, 14

swap_y
Matrix, [14](#)