Project 2 - FYS4150

Generated by Doxygen 1.14.0

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Topic Documentation

4.1 Standalone Functions

Various stand-alone functions appearing in the project.

Functions

- void jacobi_rotate (arma::mat &A, arma::mat &R, int k, int l)

  Performs a single Jacobi rotation.
- void jacobi_eigensolver (const arma::mat &A, double eps, arma::vec &eigenvalues, arma::mat &eigenvectors, const int maxiter, int &iterations, bool &converged)

  Computes the eigenvalues and eigenvectors of a symmetric matrix using Jacobi's rotation method.
- void analytic_solution (arma::vec &eigenvalues, arma::mat &eigenvectors, double a, double d, int N)

  Gives the analytic solution for the eigenvalues and eigenvectors of $A\vec{v} = \lambda\vec{v}$, where A is a tridiagonal matrix(a,d,a).
- arma::mat create_tridiagonal (int n, double a, double d, double e)

  Creates a symmetric tridiagonal matrix of size n x n with constant diagonal elements d, sub-diagonal elements a, and super-diagonal elements e.
- double max_offdiag_symmetric (const arma::mat &A, int &k, int &l)

  Finds the greatest off-diagonal element in the upper triangular part (in absolute value) of a given symmetric matrix.

4.1.1 Detailed Description

Various stand-alone functions appearing in the project.

4.1.2 Function Documentation

4.1.2.1 analytic_solution()

```
void analytic_solution (
            arma::vec & eigenvalues,
            arma::mat & eigenvectors,
            double a,
            double d,
            int N)
```

Gives the analytic solution for the eigenvalues and eigenvectors of $A\vec{v} = \lambda\vec{v}$, where A is a tridiagonal matrix(a,d,a).

Parameters

eigenvalues	Armadillo vector for eigenvalues.
eigenvectors	Armadillo vector for eigenvectors.
a	Upper and lower diagonal of matrix.
d	Diagonal of matrix.
N	Size of matrix.

4.1.2.2 create_tridiagonal()

```
arma::mat create_tridiagonal (
            int n,
            double a,
            double d,
            double e)
```

Creates a symmetric tridiagonal matrix of size n x n with constant diagonal elements d, sub-diagonal elements a, and super-diagonal elements e.

Parameters

n	size of the matrix
a	sub-diagonal elements
d	diagonal elements
e	super-diagonal elements

Returns

the tridiagonal matrix

4.1.2.3 jacobi_eigensolver()

```
void jacobi_eigensolver (
            const arma::mat & A,
            double eps,
            arma::vec & eigenvalues,
            arma::mat & eigenvectors,
            const int maxiter,
            int & iterations,
            bool & converged)
```

Computes the eigenvalues and eigenvectors of a symmetric matrix using Jacobi's rotation method.

Parameters

A	The symmetric matrix to be diagonalized.
eps	The convergence tolerance for the off-diagonal elements.
eigenvalues	Vector to store the computed eigenvalues (output).
eigenvectors	Matrix to store the computed eigenvectors (output).
maxiter	The maximum number of iterations allowed.
iterations	The number of iterations performed (output).
converged	Boolean flag indicating whether the method converged (output).

4.1.2.4 jacobi_rotate()

```
void jacobi_rotate (
            arma::mat & A,
            arma::mat & R,
            int k,
            int l)
```

Performs a single Jacobi rotation.

Parameters

A	The symmetric matrix to be diagonalized.
R	The matrix of eigenvectors.
k	The row index of the maximal off-diagonal element.
l	The column index of the maximal off-diagonal element.

4.1.2.5 max_offdiag_symmetric()

```
double max_offdiag_symmetric (
            const arma::mat & A,
            int & k,
            int & l)
```

Finds the greatest off-diagonal element in the upper triangular part (in absolute value) of a given symmetric matrix.

Parameters

A	Symmetric matrix.
k	Row index.
l	Column index.

Returns

Greatest off-diagonal element in the upper triangular part (in absolute value) of A.

Chapter 5

Class Documentation

5.1 Args Struct Reference

Struct to hold command-line arguments.

```
#include <arg_parser.hpp>
```

Public Attributes

- std::string outfile = "build/outfile.csv"

  Where and what to store outfile, associated to given problem (5 or 6).
- bool run_tests = false

  If true, runs the tests defined in "tests/".
- bool run_problem_5 = false

  If true, runs problem 5.
- bool run_problem_6 = false

  If true, runs problem 6.
- double tol = 1e-14

  Tolerance when running Jacobi's rotation method.
- int n_steps = 10

  Number of steps when running Jacobi's rotation method.
- int N_max = 100

  Number of different sizes for the matrix A in Jacobi's rotation method (problem 5).
- int maxiter = 10000

  Maximum number of iterations when running Jacobi's method.

5.1.1 Detailed Description

Struct to hold command-line arguments.

5.1.2 Member Data Documentation

5.1.2.1 maxiter

```
int Args::maxiter = 10000
```

Maximum number of iterations when running Jacobi's method.

5.1.2.2 N_max

```
int Args::N_max = 100
```

Number of different sizes for the matrix A in Jacobi's rotation method (problem 5).

5.1.2.3 n_steps

```
int Args::n_steps = 10
```

Number of steps when running Jacobi's rotation method.

5.1.2.4 outfile

```
std::string Args::outfile = "build/outfile.csv"
```

Where and what to store outfile, associated to given problem (5 or 6).

5.1.2.5 run_problem_5

```
bool Args::run_problem_5 = false
```

If true, runs problem 5.

5.1.2.6 run_problem_6

```
bool Args::run_problem_6 = false
```

If true, runs problem 6.

5.1.2.7 run_tests

```
bool Args::run_tests = false
```

If true, runs the tests defined in "tests/".

5.1.2.8 tol

```
double Args::tol = 1e-14
```

Tolerance when running Jacobi's rotation method.

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

- arg_parser.hpp

5.2 TriDag Class Reference

Class representing a tridiagonal matrix, i.e. an matrix of the form.

```
#include <triDag.hpp>
```

Public Member Functions

- TriDag (double h, int N)

  Creates a new tridiagonal object.
- void compute_eigenvalues ()

  Computes the eigenvalues of the tridiagonal matrix using arma::eig_sym.
- void print (int max=25)

  Prints the elements of the tridiagonal matrix.

Public Attributes

- arma::mat A
- arma::vec eigenvalues
- arma::mat eigenvectors

5.2.1 Detailed Description

Class representing a tridiagonal matrix, i.e. an matrix of the form.

$$A = \begin{pmatrix} a & c & 0 & \cdots & 0 \\ b & a & c & \cdots & 0 \\ 0 & b & a & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & c \\ 0 & \cdots & 0 & b & a \end{pmatrix},$$

where $b = c = -1/h^2$ and $a = 2/h^2$.

Parameters

A	Armadillo matrix.
eigenvalues	Armadillo vector for the eigenvalues of the matrix A.
eigenvectors	Armadillo vector for the eigenvectors of the matrix A.

See also

TriDag

5.2.2 Constructor & Destructor Documentation

5.2.2.1 TriDag()

```
TriDag::TriDag (
             double h,
             int N)
```

Creates a new tridiagonal object.

Parameters

h	Stepsize, defining the diagonals of the tridiagonal matrix, see general description.
N	Size of matrix.

5.2.3 Member Function Documentation

5.2.3.1 compute_eigenvalues()

```
void TriDag::compute_eigenvalues ()
```

Computes the eigenvalues of the tridiagonal matrix using arma::eig_sym.

5.2.3.2 print()

```
void TriDag::print (
             int max = 25)
```

Prints the elements of the tridiagonal matrix.

Parameters

max	Only prints matrix if the size of the matrix is less than or equal to max.

5.2.4 Member Data Documentation

5.2.4.1 A

```
arma::mat TriDag::A
```

5.2.4.2 eigenvalues

```
arma::vec TriDag::eigenvalues
```

5.2.4.3 eigenvectors

```
arma::mat TriDag::eigenvectors
```

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

- triDag.hpp

Chapter 6

File Documentation

6.1 arg_parser.hpp File Reference

```
#include <string>
#include <iostream>
```

Include dependency graph for arg_parser.hpp:

6.2 arg_parser.hpp

Go to the documentation of this file.

```
#ifndef ARG_PARSER_CPP
#define ARG_PARSER_CPP

#include <string>
#include <iostream>

struct Args
{
    std::string outfile = "build/outfile.csv";
    bool run_tests = false;
    bool run_problem_5 = false;
    bool run_problem_6 = false;
    double tol = 1e-14;
    int n_steps = 10;
    int N_max = 100;
    int maxiter = 10000;
};


Args parse_args(int argc, char *argv[]);

#endif
```

6.3 jacobi_eigensolver.hpp File Reference

```
#include <armadillo>
#include "utils.hpp"
```

Include dependency graph for jacobi_eigensolver.hpp:

Functions

- void jacobi_rotate (arma::mat &A, arma::mat &R, int k, int l)

  Performs a single Jacobi rotation.
- void jacobi_eigensolver (const arma::mat &A, double eps, arma::vec &eigenvalues, arma::mat &eigenvectors, const int maxiter, int &iterations, bool &converged)

  Computes the eigenvalues and eigenvectors of a symmetric matrix using Jacobi's rotation method.

6.4 jacobi_eigensolver.hpp

Go to the documentation of this file.

```
#ifndef JACOBI_EIGENSOLVER_HPP
#define JACOBI_EIGENSOLVER_HPP

#include <armadillo>
#include "utils.hpp"


void jacobi_rotate(arma::mat &A, arma::mat &R, int k, int l);

void jacobi_eigensolver(const arma::mat &A, double eps, arma::vec &eigenvalues, arma::mat
      &eigenvectors, const int maxiter, int &iterations, bool &converged);

#endif

```

6.5 problems.hpp File Reference

`#include <string>`

Include dependency graph for problems.hpp:

Functions

- void problem_5 (double N_max, double tol, int maxiter, const std::string &outfile)

  Iterates from $N = 5$ to $\texttt{N\_max}$, and for each iteration creating a symmetric $N \times N$ matrix using triDag::create_tridiaginal and computing the eigenvalues using Jacobi's rotation method implemented in jacobi_eigensolver::jacobi_eigensolver. Writes the result to `outfile`.
- void problem_6 (int n_steps, double tol, int maxiter, const std::string &outfile)

  Creates a tridiagonal matrix using triDag::create_tridiagonal, computes its eigenvalues and eigenvectors using Jacobi's rotation method implemented in jacobi_eigensolver::jacobi_eigensolver. Writes these eigenvalues and eigenvectors to outfile.

6.5.1 Function Documentation

6.5.1.1 problem_5()

```
void problem_5 (
            double N_max,
            double tol,
            int maxiter,
            const std::string & outfile)
```

Iterates from $N = 5$ to $\texttt{N\_max}$, and for each iteration creating a symmetric $N \times N$ matrix using triDag::create_tridiaginal and computing the eigenvalues using Jacobi's rotation method implemented in jacobi_eigensolver::jacobi_eigensolver. Writes the result to `outfile`.

Parameters

N_max	Final size of matrix.
tol	Tolerance passed to jacobi_eigensolver::jacobi_eigensolver.
maxiter	Maximum number of iterations.
outfile	File to write results to.

6.5.1.2 problem_6()

```
void problem_6 (
            int n_steps,
            double tol,
            int maxiter,
            const std::string & outfile)
```

Creates a tridiagonal matrix using triDag::create_tridiagonal, computes its eigenvalues and eigenvectors using Jacobi's rotation method implemented in jacobi_eigensolver::jacobi_eigensolver. Writes these eigenvalues and eigenvectors to outfile.

Parameters

n_steps	Number if steps for Jacobi's rotation method (1 - size of tridiagonal matrix).
tol	Tolerance passed to jacobi_eigensolver::jacobi_eigensolver.
maxiter	Maximum number of iterations.
outfile	File to write results to.

6.6 problems.hpp

Go to the documentation of this file.

```
#ifndef PROBLEMS
#define PROBLEMS
#include <string>

void problem_5(double N_max, double tol, int maxiter, const std::string &outfile);


void problem_6(int n_steps, double tol, int maxiter, const std::string &outfile);

#endif
```

6.7 triDag.hpp File Reference

`#include <armadillo>`
Include dependency graph for triDag.hpp:

Classes

- class TriDag

  Class representing a tridiagonal matrix, i.e. an matrix of the form.

Functions

- void analytic_solution (arma::vec &eigenvalues, arma::mat &eigenvectors, double a, double d, int N)

  Gives the analytic solution for the eigenvalues and eigenvectors of $A\vec{v} = \lambda\vec{v}$, where A is a tridiagonal matrix(a,d,a).

6.8 triDag.hpp

Go to the documentation of this file.

```
#ifndef TRI_DAG
#define TRI_DAG
#include <armadillo>


class TriDag{
private:
    double a_fill;
    double d_fill;
    int N;

    void create_matrix();  // Creates tridiagonal matrix

public:
    arma::mat A;
    arma::vec eigenvalues;
    arma::mat eigenvectors;

    TriDag(double h, int N);


    void compute_eigenvalues();

    void print(int max=25);
};

void analytic_solution(arma::vec &eigenvalues, arma::mat &eigenvectors, double a, double d, int N);

#endif
```

6.9 utils.hpp File Reference

`#include <armadillo>`

Include dependency graph for utils.hpp: This graph shows which files directly or indirectly include this file:

Functions

- arma::mat create_tridiagonal (int n, double a, double d, double e)

  Creates a symmetric tridiagonal matrix of size n x n with constant diagonal elements d, sub-diagonal elements a, and super-diagonal elements e.

- double max_offdiag_symmetric (const arma::mat &A, int &k, int &l)

  Finds the greatest off-diagonal element in the upper triangular part (in absolute value) of a given symmetric matrix.

6.10 utils.hpp

Go to the documentation of this file.

```
#ifndef UTILS
#define UTILS
#include <armadillo>





arma::mat create_tridiagonal(int n, double a, double d, double e);


double max_offdiag_symmetric(const arma::mat &A, int &k, int &l);

#endif
```

Index