Hierarchical Matrices Lib

Generated by Doxygen 1.9.4

1 hierarchical-matrices-lib

This is an incomplete C++ library for efficiently computing approximations of the most common matrix calculations via hierarchical matrices.

Note that input matrices can only be square!

To use this library in your code, download the repository and use #include "hierarchical_matrices_lib.cpp" while making sure that the corresponding file is reachable from the code where you've included it. You will at least need the Lapacke library and at most the development tools listed below to compile your code now.

The only file that should be changed is "user_settings", in which you will be able to switch between the datatypes float, double, complex floats and complex doubles. If you are only using float, there is no need to change the settings
since it is the default datatype.

Please see the included documentation file for further instructions on how to use this library.
Any mentioning of pages refer to the book "Hierarchical Matrices", see below.

Concrete examples can be found in the "example.cc". If you're using the tools as listed below, changing the paths in the Makefile should make you able to compile the code via typing "make" into a terminal/console.

1.1 Available functions:

- Constructor
- Coarsening
- Matrix-matrix addition
- Matrix-vector multiplication

1.2 Used sources/tools for development:

- "Hierarchical Matrices" by Mario Bebendorf from 2008
  (see https://link.springer.com/book/10.1007/978-3-540-77147-0)
  Specifically the first two chapters about "Low-Rank Matrices and Matrix Partitioning" and "Hierarchical Matrices"
- OS: Windows
- C++ version: C17
- Compiler: MinGW 32
- Libraries: LAPACKE 3.10 (Instructions: https://icl.utk.edu/lapack-for-windows/lapack/#lapacke)

  --> Needs C, C++ and Fortran compiler (e.g. gcc, g++ and gfortran included in MinGW)
- Documentation: Doxygen

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

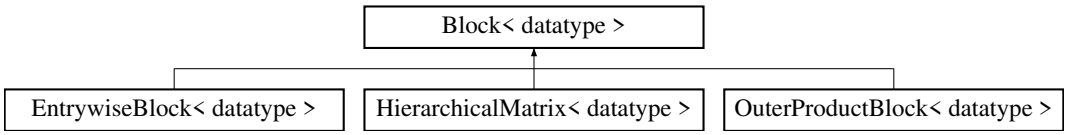
Here is a list of all documented files with brief descriptions:

5 Class Documentation

5.1 Block< datatype > Class Template Reference

Abstract class to depict all parts of the matrix.

Inheritance diagram for Block< datatype >:



Protected Member Functions

- **Block** (unsigned int m, unsigned int n)

  Automatic initialization of attributes from subclasses.
- **Block** ()

  Empty constructor for calling empty subclass constructors.
- virtual Block * coarse (double accuracy, bool checkForLeaf)=0

  Actual coarse functionality.
- virtual unsigned int getStorageOrRank (bool getStorage)=0
- virtual datatype * operator* (const datatype vector[ ])=0
- virtual Block * operator+ (Block *addedBlock)=0

  First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.
- virtual Block * operator+ (HierarchicalMatrix< datatype > *addedBlock)=0

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- virtual Block * operator+ (OuterProductBlock< datatype > *addedBlock)=0

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- virtual Block * operator+ (EntrywiseBlock< datatype > *addedBlock)=0

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- virtual ~**Block** ()

  Empty virtual destructor to polymorphically destruct matrix attribute in HierarchicalMatrix.

Private Attributes

- unsigned int **mDim**

  Number of rows from the original matrix depicted within this block.
- unsigned int **nDim**

  Number of columns from the original matrix depicted within this block.
- unsigned int **indiceRange** [2][2]

  Lower and upper bound of row and column indices from the original matrix depicted within this block, see enum InndiceOrientation.

Friends

- class **OuterProductBlock< datatype >**
- class **EntrywiseBlock< datatype >**
- class **HierarchicalMatrix< datatype >**

5.1.1 Detailed Description

template<class datatype>
class Block< datatype >

Abstract class to depict all parts of the matrix.

5.1.2 Member Function Documentation

5.1.2.1 coarse() template<class datatype >

```
virtual Block * Block< datatype >::coarse (
            double accuracy,
            bool checkForLeaf )  [protected], [pure virtual]
```

Actual coarse functionality.

Implemented in EntrywiseBlock< datatype >, HierarchicalMatrix< datatype >, and OuterProductBlock< datatype >.

5.1.2.2 getStorageOrRank() template<class datatype >

```
virtual unsigned int Block< datatype >::getStorageOrRank (
            bool getStorage )  [protected], [pure virtual]
```

Helper function for coarse

Returns

Storage of EW or OP Block if getStorage = true, rank of OP if getStorage = false, error if called otherwise since it's not necessary

Implemented in EntrywiseBlock< datatype >, HierarchicalMatrix< datatype >, and OuterProductBlock< datatype >.

5.1.2.3 operator*() template<class datatype >

```
virtual datatype * Block< datatype >::operator* (
            const datatype vector[] )  [protected], [pure virtual]
```

Implemented in HierarchicalMatrix< datatype >.

5.1.2.4 operator+() [1/4] template<class datatype >

```
virtual Block * Block< datatype >::operator+ (
            Block< datatype > * addedBlock ) [protected], [pure virtual]
```

First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.

Implemented in EntrywiseBlock< datatype >, HierarchicalMatrix< datatype >, and OuterProductBlock< datatype >.

5.1.2.5 operator+() [2/4] template<class datatype >

```
virtual Block * Block< datatype >::operator+ (
            EntrywiseBlock< datatype > * addedBlock ) [protected], [pure virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implemented in EntrywiseBlock< datatype >, HierarchicalMatrix< datatype >, and OuterProductBlock< datatype >.

5.1.2.6 operator+() [3/4] template<class datatype >

```
virtual Block * Block< datatype >::operator+ (
            HierarchicalMatrix< datatype > * addedBlock ) [protected], [pure virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implemented in HierarchicalMatrix< datatype >, EntrywiseBlock< datatype >, and OuterProductBlock< datatype >.

5.1.2.7 operator+() [4/4] template<class datatype >

```
virtual Block * Block< datatype >::operator+ (
            OuterProductBlock< datatype > * addedBlock ) [protected], [pure virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

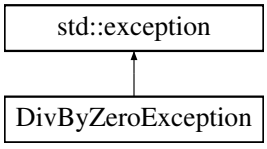
Implemented in OuterProductBlock< datatype >, EntrywiseBlock< datatype >, and HierarchicalMatrix< datatype >.

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

- hierarchical_matrices_lib_code/Block.hpp

5.2 DivByZeroException Class Reference

Inheritance diagram for DivByZeroException:



Public Member Functions

- virtual const char * **what** () const noexcept

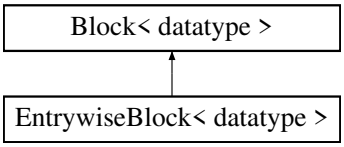
The documentation for this class was generated from the following file:

- hierarchical_matrices_lib_code/Exceptions.hpp

5.3 EntrywiseBlock< datatype > Class Template Reference

Depicts a non-admissible or full rank part of the original matrix.

Inheritance diagram for EntrywiseBlock< datatype >:



Protected Member Functions

- **EntrywiseBlock** (datatype **originalBlock, unsigned int mDim, unsigned int nDim, std::vector< unsigned int > iInd, std::vector< unsigned int > jInd)

  Copies part of the matrix to be stored in a block.
- Block< datatype > * coarse (double accuracy, bool checkForLeaf) final

  Actual coarse functionality.
- unsigned int getStorageOrRank (bool getStorage) final
- datatype * operator* (const datatype vector[ ])
- Block< datatype > * operator+ (Block< datatype > *addedBlock)

  First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.
- Block< datatype > * operator+ (HierarchicalMatrix< datatype > *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- Block< datatype > * operator+ (OuterProductBlock< datatype > *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- Block< datatype > * operator+ (EntrywiseBlock *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- ~**EntrywiseBlock** ()

  Frees all memory allocated for the block array.

Private Attributes

- datatype ** **block**

  *mDim * nDim array / Exact part of the original matrix*

Friends

- class **OuterProductBlock**< **datatype** >
- class **HierarchicalMatrix**< **datatype** >

5.3.1 Detailed Description

template<class datatype>
class EntrywiseBlock< datatype >

Depicts a non-admissible or full rank part of the original matrix.

5.3.2 Member Function Documentation

5.3.2.1 coarse() template<class datatype >

```
Block< datatype > * EntrywiseBlock< datatype >::coarse (
            double accuracy,
            bool checkForLeaf )  [final], [protected], [virtual]
```

Actual coarse functionality.

Implements Block< datatype >.

5.3.2.2 getStorageOrRank() template<class datatype >

```
unsigned int EntrywiseBlock< datatype >::getStorageOrRank (
            bool getStorage )  [final], [protected], [virtual]
```

Helper function for coarse

Returns

Storage of EW or OP Block if getStorage = true, rank of OP if getStorage = false, error if called otherwise since it's not necessary

Implements Block< datatype >.

5.3.2.3 operator*() template<class datatype >

```
datatype * EntrywiseBlock< datatype >::operator* (
            const datatype vector[] )  [protected], [virtual]
```

Implements Block< datatype >.

5.3.2.4 operator+() [1/4] template<class datatype >

```
Block< datatype > * EntrywiseBlock< datatype >::operator+ (
            Block< datatype > * addedBlock ) [protected], [virtual]
```

First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.

Implements Block< datatype >.

5.3.2.5 operator+() [2/4] template<class datatype >

```
Block< datatype > * EntrywiseBlock< datatype >::operator+ (
            EntrywiseBlock< datatype > * addedBlock ) [protected], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

5.3.2.6 operator+() [3/4] template<class datatype >

```
Block< datatype > * EntrywiseBlock< datatype >::operator+ (
            HierarchicalMatrix< datatype > * addedBlock ) [protected], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

5.3.2.7 operator+() [4/4] template<class datatype >

```
Block< datatype > * EntrywiseBlock< datatype >::operator+ (
            OuterProductBlock< datatype > * addedBlock ) [protected], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

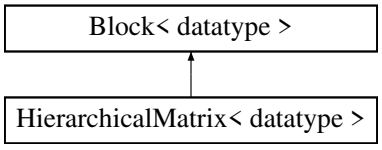
The documentation for this class was generated from the following files:

- hierarchical_matrices_lib_code/Block.hpp
- hierarchical_matrices_lib_code/EntrywiseBlock.hpp

5.4 HierarchicalMatrix< datatype > Class Template Reference

Transforms a square entrywise matrix into a hierarchical matrix to be calculated with.

Inheritance diagram for HierarchicalMatrix< datatype >:



Public Member Functions

- HierarchicalMatrix (datatype **originalMatrix, std::list< std::vector< unsigned int > > *originalIndices, unsigned int dim, double clusterParamEta=0.5)
- void coarse (double accuracy)
- HierarchicalMatrix * operator+ (const HierarchicalMatrix &addedMatrix)
- HierarchicalMatrix * operator+= (const HierarchicalMatrix &addedMatrix)
- datatype * operator* (const datatype vector[ ])
- ~**HierarchicalMatrix** ()

  Recursively frees all Blocks saved in the matrix-attribute.

Private Member Functions

- **HierarchicalMatrix** (datatype **originalMatrix, std::list< std::vector< unsigned int > > *originalIndices, unsigned int indices[2][2], double clusterParamEta, unsigned int **distances)

  Internal connstructor for all other layers except the outside constructor call.
- void **constructHierarchicalMatrix** (datatype **originalMatrix, std::list< std::vector< unsigned int > > *originalIndices, unsigned int indices[2][2], double clusterParamEta, unsigned int **distances)

  Outsourced part of the constructor that holds code to be executed in both the public and private one.
- Block< datatype > * coarse (double accuracy, bool checkForLeaf) final

  Actual coarse functionality.
- unsigned int getStorageOrRank (bool getStorage) final
- Block< datatype > * operator+ (Block< datatype > *addedBlock)

  First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.
- Block< datatype > * operator+ (HierarchicalMatrix *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- Block< datatype > * operator+ (OuterProductBlock< datatype > *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- Block< datatype > * operator+ (EntrywiseBlock< datatype > *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Private Attributes

- Block< datatype > * **matrix** [2][2]

  Hierarchical matrix, recursively divided into quadrants until one of the other Block types is reached, can also hold nullptr if a 2x1 or 1x2 division is reached.

Friends

- class **OuterProductBlock**< **datatype** >
- class **EntrywiseBlock**< **datatype** >

Additional Inherited Members

5.4.1 Detailed Description

template<**class datatype**>
class HierarchicalMatrix< **datatype** >

Transforms a square entrywise matrix into a hierarchical matrix to be calculated with.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 HierarchicalMatrix() template<class datatype >

```
HierarchicalMatrix< datatype >::HierarchicalMatrix (
            datatype ** originalMatrix,
            std::list< std::vector< unsigned int > > * originalIndices,
            unsigned int dim,
            double clusterParamEta = 0.5 )
```

Transforms an entrywise matrix with float, double or complex float entries into a hierarchical matrix Example↩ : HierarchicalMatrix<double> exampleMatrix(data, &indices, 100)

Parameters

originalMatrix	A two-dimensional array containing the square entrywise matrix to be transformed and calculated with
originalIndices	List containing the admissible partition into blocks for column and row indices of the matrix, each vector contains all the indices in ascending order, vectors are listed in ascending order so the first vector always s withh 0 and the last ends with dim-1, see page 31: Fig 1.5
dim	Number of columns/rows of the input matrix
clusterParamEta	Optional cluster parameter = (0.0, 1.0), will be defaulted to 0.5 if no value is given, see page 24: (1.13)

5.4.3 Member Function Documentation

5.4.3.1 coarse() [1/2] template<class datatype >

```
void HierarchicalMatrix< datatype >::coarse (
            double accuracy )
```

Coarsens the given hierarchical matrix until the given accuracy can no longer be held, usually keeping the same storage size while reducing the number of branches/blocks

Parameters

accuracy	Accuracy = (0.0, 1.0) to be satisfied in each coarsening step, see page 72: (2.13)

5.4.3.2 coarse() [2/2] template<class datatype >

```
Block< datatype > * HierarchicalMatrix< datatype >::coarse (
            double accuracy,
            bool checkForLeaf )  [final], [private], [virtual]
```

Actual coarse functionality.

Implements Block< datatype >.

5.4.3.3 getStorageOrRank() template<class datatype >

```
unsigned int HierarchicalMatrix< datatype >::getStorageOrRank (
            bool getStorage ) [final], [private], [virtual]
```

Helper function for coarse

Returns

Storage of EW or OP Block if getStorage = true, rank of OP if getStorage = false, error if called otherwise since it's not necessary

Implements Block< datatype >.

5.4.3.4 operator*() template<class datatype >

```
datatype * HierarchicalMatrix< datatype >::operator* (
            const datatype vector[] ) [virtual]
```

Matrix-vector multiplication

Parameters

vector	One-dimensional array with size equaling the number of columns of the input matrix

Returns

Resulting vector with size equaling the number of rows of the input matrix, nullptr if vector couldn't be calculated

Implements Block< datatype >.

5.4.3.5 operator+() [1/5] template<class datatype >

```
Block< datatype > * HierarchicalMatrix< datatype >::operator+ (
            Block< datatype > * addedBlock ) [private], [virtual]
```

First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.

Implements Block< datatype >.

5.4.3.6 operator+() [2/5] template<class datatype >

```
HierarchicalMatrix * HierarchicalMatrix< datatype >::operator+ (
            const HierarchicalMatrix< datatype > & addedMatrix )
```

Rounded addition of two Hierarchical Matrices of the same size

Parameters

addedMatrix	Second matrix to be added with the same number of rows and columns as the first matrix

Returns

Sum of the two matrices with the same number of rows and columns as the input matrices, nullptr if matrices couldn't be added

5.4.3.7 operator+() [3/5] template<class datatype >

```
Block< datatype > * HierarchicalMatrix< datatype >::operator+ (
            EntrywiseBlock< datatype > * addedBlock ) [private], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

5.4.3.8 operator+() [4/5] template<class datatype >

```
Block< datatype > * HierarchicalMatrix< datatype >::operator+ (
            HierarchicalMatrix< datatype > * addedBlock ) [private], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

5.4.3.9 operator+() [5/5] template<class datatype >

```
Block< datatype > * HierarchicalMatrix< datatype >::operator+ (
            OuterProductBlock< datatype > * addedBlock ) [private], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

5.4.3.10 operator+=() template<class datatype >

```
HierarchicalMatrix * HierarchicalMatrix< datatype >::operator+= (
            const HierarchicalMatrix< datatype > & addedMatrix )
```

Rounded addition of two Hierarchical Matrices of the same size

Parameters

addedMatrix	Second matrix to be added with the same number of rows and columns as the first matrix

Returns

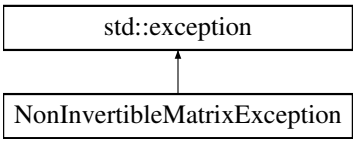
Sum of the two matrices with the same number of rows and columns as the input matrices, nullptr if matrices couldn't be added

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

- hierarchical_matrices_lib_code/Block.hpp
- hierarchical_matrices_lib_code/HierarchicalMatrix.hpp

5.5 NonInvertibleMatrixException Class Reference

Inheritance diagram for NonInvertibleMatrixException:



Public Member Functions

- virtual const char ∗ **what** () noexcept

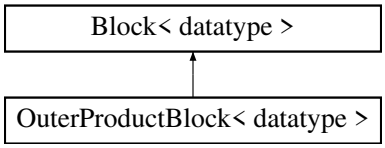
The documentation for this class was generated from the following file:

- hierarchical_matrices_lib_code/Exceptions.hpp

5.6 OuterProductBlock< datatype > Class Template Reference

Depicts an admissible or low-rank part of the original matrix.

Inheritance diagram for OuterProductBlock< datatype >:



Protected Member Functions

- **OuterProductBlock** (datatype **originalBlock, unsigned int mDim, unsigned int nDim, std::vector< unsigned int > iInd, std::vector< unsigned int > jInd, unsigned int rank)

  Transforms an entrywise part of the matrix into its outer product form.
- Block< datatype > * coarse (double accuracy, bool checkForLeaf) final

  Actual coarse functionality.
- unsigned int getStorageOrRank (bool getStorage) final
- datatype * operator* (const datatype vector[ ])
- Block< datatype > * operator+ (Block< datatype > *addedBlock)

  First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.
- Block< datatype > * operator+ (HierarchicalMatrix< datatype > *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- Block< datatype > * operator+ (OuterProductBlock *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- Block< datatype > * operator+ (EntrywiseBlock< datatype > *addedBlock)

  Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.
- ~**OuterProductBlock** ()

  Frees all memory allocated for the u, x and v arrays.

Private Attributes

- datatype ** **u**

  *mDim * k array / U*
- datatype ** **x**

  *k * k array / X, similar to S*
- datatype ** **v**

  *nDim * k array / V^H with complex entries, V^T with real entries*
- unsigned int **k**

  Rank of original matrix part.

Friends

- class **EntrywiseBlock**< **datatype** >
- class **HierarchicalMatrix**< **datatype** >

5.6.1 Detailed Description

template<**class datatype**>
class OuterProductBlock< **datatype** >

Depicts an admissible or low-rank part of the original matrix.

5.6.2 Member Function Documentation

5.6.2.1 coarse() template<class datatype >

```
Block< datatype > * OuterProductBlock< datatype >::coarse (
            double accuracy,
            bool checkForLeaf ) [final], [protected], [virtual]
```

Actual coarse functionality.

Implements Block< datatype >.

5.6.2.2 getStorageOrRank() template<class datatype >

```
unsigned int OuterProductBlock< datatype >::getStorageOrRank (
            bool getStorage ) [final], [protected], [virtual]
```

Helper function for coarse

Returns

Storage of EW or OP Block if getStorage = true, rank of OP if getStorage = false, error if called otherwise since it's not necessary

Implements Block< datatype >.

5.6.2.3 operator*() template<class datatype >

```
datatype * OuterProductBlock< datatype >::operator* (
            const datatype vector[] ) [protected], [virtual]
```

Implements Block< datatype >.

5.6.2.4 operator+() [1/4] template<class datatype >

```
Block< datatype > * OuterProductBlock< datatype >::operator+ (
            Block< datatype > * addedBlock ) [protected], [virtual]
```

First layer of polymorphic addition, detects the first operand and calls swapped addition to detect the second operand through the second layer.

Implements Block< datatype >.

5.6.2.5 operator+() [2/4] template<class datatype >

```
Block< datatype > * OuterProductBlock< datatype >::operator+ (
            EntrywiseBlock< datatype > * addedBlock ) [protected], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

5.6.2.6 operator+() [3/4] template<class datatype >

```
Block< datatype > * OuterProductBlock< datatype >::operator+ (
            HierarchicalMatrix< datatype > * addedBlock )  [protected], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

5.6.2.7 operator+() [4/4] template<class datatype >

```
Block< datatype > * OuterProductBlock< datatype >::operator+ (
            OuterProductBlock< datatype > * addedBlock )  [protected], [virtual]
```

Second layer of polymorphic addition, detects the second operand and actually calculates the correxponding sum.

Implements Block< datatype >.

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

- hierarchical_matrices_lib_code/Block.hpp
- hierarchical_matrices_lib_code/OuterProductBlock.hpp

6 File Documentation

6.1 Block.hpp

```
1 #ifndef HIERARCHICAL_MATRICES_BLOCK_H
2 #define HIERARCHICAL_MATRICES_BLOCK_H
3
4 template <class datatype> class HierarchicalMatrix;
5 template <class datatype> class OuterProductBlock;
6 template <class datatype> class EntrywiseBlock;
7
9 enum IndiceOrientation {kRangeI=0, kRangeJ=1, kBottom=0, kTop=1};
10
11
13 template <class datatype>
14 class Block {
15
16 friend class OuterProductBlock<datatype>;
17 friend class EntrywiseBlock<datatype>;
18 friend class HierarchicalMatrix<datatype>;
19
20 private:
21
22       unsigned int mDim;
23       unsigned int nDim;
24       unsigned int indiceRange[2][2];
25
26 protected:
28       Block(unsigned int m, unsigned int n): mDim(m), nDim(n) {}
30       Block(){}
31
33       virtual Block* coarse( double accuracy, bool checkForLeaf ) =0;
36       virtual unsigned int getStorageOrRank( bool getStorage ) =0;
37
38       virtual datatype* operator*( const datatype vector[] ) =0;
39
41       virtual Block* operator+( Block* addedBlock ) =0;
43       virtual Block* operator+( HierarchicalMatrix<datatype>* addedBlock ) =0;
45       virtual Block* operator+( OuterProductBlock<datatype>* addedBlock ) =0;
47       virtual Block* operator+( EntrywiseBlock<datatype>* addedBlock ) =0;
48
50       virtual ~Block(){};
51 };
52
53 #endif // HIERARCHICAL_MATRICES_BLOCK_H
```

6.2 EntrywiseBlock.hpp

```
1 #ifndef HIERARCHICAL_MATRICES_ENTRYWISEBLOCK_H
2 #define HIERARCHICAL_MATRICES_ENTRYWISEBLOCK_H
3
4 #include "Block.hpp"
5
6 #include <list>
7 #include <vector>
8
9
11 template <class datatype>
12 class EntrywiseBlock: public Block<datatype>{
13
14 friend class OuterProductBlock<datatype>;
15 friend class HierarchicalMatrix<datatype>;
16
17 private:
18       datatype ** block;
19
20 protected:
21       EntrywiseBlock(){}
23       EntrywiseBlock(datatype ** originalBlock, unsigned int mDim, unsigned int nDim,
      std::vector<unsigned int> iInd, std::vector<unsigned int> jInd);
24
25       Block<datatype>* coarse( double accuracy, bool checkForLeaf ) final;
26       unsigned int getStorageOrRank( bool getStorage ) final;
27
28       datatype* operator*( const datatype vector[] );
29
30       Block<datatype>* operator+( Block<datatype>* addedBlock );
31       Block<datatype>* operator+( HierarchicalMatrix<datatype>* addedBlock );
32       Block<datatype>* operator+( OuterProductBlock<datatype>* addedBlock );
33       Block<datatype>* operator+( EntrywiseBlock* addedBlock );
34
36       ~EntrywiseBlock();
37 };
38
39 #endif // HIERARCHICAL_MATRICES_ENTRYWISEBLOCK_H
```

6.3 Exceptions.hpp

```
1 #ifndef HIERARCHICAL_MATRICES_EXCEPTIONS_H
2 #define HIERARCHICAL_MATRICES_EXCEPTIONS_H
3
4 #include <stdexcept>
5 #include <exception> //enthält what()
6
7 //fkt()noexcept{} ==in fkt darf/wird keine Exception geworfen
8
9 // try{
10 //       if(1==1) {
11 //             throw( DivByZeroException() );
12 //       }
13 // } catch( DivByZeroException exc ){
14 //       std::cerr « exc.what();
15 // }
16
17
18 // Exception for divisions by zero
19 class DivByZeroException: public std::exception{
20 public:
21       virtual const char* what() const noexcept{
22             return "Division by zero\n";
23       }
24 };
25
26
27 // Exception for when the matrix inversion fails due to unchecked invertibility
28 class NonInvertibleMatrixException: public std::exception{
29 public:
30       virtual const char* what() noexcept{
31             return "Given matrix cannot be inverted\n";
32       }
33 };
34
35 #endif // HIERARCHICAL_MATRICES_EXCEPTIONS_H
```

6.4 HierarchicalMatrix.hpp

```
1 #ifndef HIERARCHICAL_MATRICES_HIERARCHICALMATRIX_H
```

```
2 #define HIERARCHICAL_MATRICES_HIERARCHICALMATRIX_H
3
4 #include "Block.hpp"
5 #include "EntrywiseBlock.hpp"
6 #include "OuterProductBlock.hpp"
7
8 #include <array>
9 #include <list>
10 #include <vector>
11
12
14 template <class datatype>
15 class HierarchicalMatrix: public Block<datatype> {
16
17 friend class OuterProductBlock<datatype>;
18 friend class EntrywiseBlock<datatype>;
19
20 private:
21       // [0][0] = top left, [0][1] = top right / [1][0] = bottom left, [1][1] = bottom right
22       Block<datatype>* matrix[2][2];
23
24 public:
34       HierarchicalMatrix(datatype ** originalMatrix, std::list<std::vector<unsigned int>*
     originalIndices, unsigned int dim, double clusterParamEta =0.5);
35
40       void coarse( double accuracy );
41
46       HierarchicalMatrix* operator+( const HierarchicalMatrix& addedMatrix );
51       HierarchicalMatrix* operator+=( const HierarchicalMatrix& addedMatrix );
52
53
58       datatype* operator*( const datatype vector[] );
59
60       // Matrix-matrix multiplication
61       // HierarchicalMatrix* operator*( const HierarchicalMatrix& multMatrix );
62       // HierarchicalMatrix* operator*=( const HierarchicalMatrix& multMatrix );
63
64       // Inversion
65       // HierarchicalMatrix* invert();
66
67       // LU-decomposition
68       // std::array<HierarchicalMatrix*,2> luDecomposition();
69
71       ~HierarchicalMatrix();
72
73 private:
74       HierarchicalMatrix(){}
76       HierarchicalMatrix(datatype ** originalMatrix, std::list<std::vector<unsigned int>*
     originalIndices, unsigned int indices[2][2], double clusterParamEta, unsigned int ** distances);
78       void constructHierarchicalMatrix(datatype ** originalMatrix, std::list<std::vector<unsigned int>*
     originalIndices, unsigned int indices[2][2], double clusterParamEta, unsigned int ** distances);
79
80       Block<datatype>* coarse( double accuracy, bool checkForLeaf ) final;
81       unsigned int getStorageOrRank( bool getStorage ) final;
82
83       Block<datatype>* operator+( Block<datatype>* addedBlock );
84       Block<datatype>* operator+( HierarchicalMatrix* addedBlock );
85       Block<datatype>* operator+( OuterProductBlock<datatype>* addedBlock );
86       Block<datatype>* operator+( EntrywiseBlock<datatype>* addedBlock );
87 };
88
89 #endif // HIERARCHICAL_MATRICES_HIERARCHICALMATRIX_H
```

6.5 OuterProductBlock.hpp

```
1 #ifndef HIERARCHICAL_MATRICES_OUTERPRODUCTBLOCK_H
2 #define HIERARCHICAL_MATRICES_OUTERPRODUCTBLOCK_H
3
4 #include "Block.hpp"
5 #include <list>
6 #include <vector>
7
8
10 template <class datatype>
11 class OuterProductBlock: public Block<datatype> {
12
13 friend class EntrywiseBlock<datatype>;
14 friend class HierarchicalMatrix<datatype>;
15
16 private:
17       datatype ** u;
18       datatype ** x;
19       datatype ** v;
20
```

```
21       unsigned int k;
22
23 protected:
24       OuterProductBlock(){}
26       OuterProductBlock(datatype ** originalBlock, unsigned int mDim, unsigned int nDim,
       std::vector<unsigned int> iInd, std::vector<unsigned int> jInd, unsigned int rank);
27
28       Block<datatype>* coarse( double accuracy, bool checkForLeaf ) final;
29       unsigned int getStorageOrRank( bool getStorage ) final;
30
31       datatype* operator*( const datatype vector[] );
32
33       Block<datatype>* operator+( Block<datatype>* addedBlock );
34       Block<datatype>* operator+( HierarchicalMatrix<datatype>* addedBlock );
35       Block<datatype>* operator+( OuterProductBlock* addedBlock );
36       Block<datatype>* operator+( EntrywiseBlock<datatype>* addedBlock );
37
39       ~OuterProductBlock();
40 };
41
42 #endif // HIERARCHICAL_MATRICES_OUTERPRODUCTBLOCK_H
```

6.6 user_settings.hpp

```
1 // This file includes necessary settings for you, the user
2 // ONLY CHANGE THE PARTS THAT ARE EXPLAINED IN THE COMMENTS
3 // Otherwise, being able to run the code cannot be guaranteed
4
5 // Change the number according to the format of the numbers in your matrices
6 // The default is float
7 // 0 = float
8 // 1 = double
9 // 2 = complex float
10 // 3 = complex double
11 #define USED_DATATYPE 1
12
13
14
15
16
17 // DO NOT CHANGE ANYTHING BEYOND THIS POINT
18
19 #if USED_DATATYPE == 1
20       #define CALC_SVD LAPACKE_dgesvd_work
21 #elif USED_DATATYPE == 2
22       #define CALC_SVD LAPACKE_cgesvd_work
23 #elif USED_DATATYPE == 3
24       #define CALC_SVD LAPACKE_zgesvd_work
25 #else
26       #define CALC_SVD LAPACKE_sgesvd_work
27 #endif
```

Index