IdeoSim for Boundary Element Method

Generated by Doxygen 1.8.13

Contents

Chapter 1

Implementation of Hackbusch's H-matrix operations

Operator	Functions to be performed	C++ functions
$\mathcal{T}^{\mathcal{R}}_{r\leftarrow s}$	Truncation of a rank-k matrix	RkMatrix<Number>::truncate_to_rank(size_type new_rank)
$\mathcal{T}^{\mathcal{H}}_{r\leftarrow s}$	Truncation of rank-k matrix blocks in an $\mathcal{H}$-matrix	HMatrix<spacedim, Number>::truncate_to_rank(size_type new_rank)
$\mathcal{T}^{\mathcal{R}\leftarrow\mathcal{F}}_{r}$	Convert a full matrix to rank-k matrix	• RkMatrix<Number>::RkMatrix(const size_type fixed_rank_k, LAPACKFullMatrixExt<Number> &M) • RkMatrix<Number>::RkMatrix(LAPACKFullMatrixExt<Number> &M)

Operator	Functions to be performed	C++ functions
$M\|_b$ where $M \in \mathcal{F}$	Restrict a full matrix M to the block cluster $b = \tau \times \sigma$ as a full matrix	• LAPACKFullMatrixExt(const std::vector<types::global_dof_index> &tau_index_set, const std::vector<types::global_dof_index> &sigma_index_set, const LAPACKFullMatrixExt<Number> & M) • LAPACKFullMatrixExt(const std::vector<types::global_dof_index> &tau_index_set, const std::vector<types::global_dof_index> &sigma_index_set, const LAPACKFullMatrixExt<Number> & M, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> &col_index_global_to_local_map_for_M)

Operator	Functions to be performed	C++ functions
$\mathcal{T}_r^{\mathcal{R}\leftarrow\mathcal{F}}(M\|_b)$ where $M \in \mathcal{F}$	Restrict a full matrix M to the block cluster $b = \tau \times \sigma$ and convert it to a rank-k matrix	• RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const size_type fixed_rank_k, const LAPACKFullMatrixExt<Number> & M) • RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const LAPACKFullMatrixExt<Number> & M) • RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const size_type fixed_rank_k, const LAPACKFullMatrixExt<Number> & M, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> &col_index_global_to_local_map_for_M) • RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const LAPACKFullMatrixExt<Number> & M, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> &col_index_global_to_local_map_for_M)

Operator	Functions to be performed	C++ functions
$\mathcal{T}^{\mathcal{R}}_{r\leftarrow s}(M\|_b)$ where $M \in \mathcal{R}$	Restrict a rank-k matrix M to the block cluster $b = \tau \times \sigma$ as a rank-k matrix	• RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const size_type fixed_rank_k, const RkMatrix<Number> & M) • RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const RkMatrix<Number> & M) • RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const size_type fixed_rank_k, const RkMatrix<Number> & M, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> &col_index_global_to_local_map_for_M) • RkMatrix<Number>::RkMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const RkMatrix<Number> & M, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> &col_index_global_to_local_map_for_M)

Operator	Functions to be performed	C++ functions
$\mathcal{T}^{\mathcal{F}\leftarrow\mathcal{R}}(M\|_b)$ where $M \in \mathcal{R}$	Restrict a rank-k matrix M to the block cluster $b = \tau \times \sigma$ as a full matrix	• RkMatrix<Number>::restrictToFullMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, LAPACKFullMatrixExt<Number> & matrix) const • RkMatrix<Number>::restrictToFullMatrix(const std::vector<types::global_dof_index> &tau, const std::vector<types::global_dof_index> &sigma, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_rk, const std::map<types::global_dof_index, size_t> & col_index_global_to_local_map_for_rk, LAPACKFullMatrixExt<Number> &matrix) const

Operator	Functions to be performed	C++ functions
$M = \begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix}$	Agglomeration of four full submatrices in `CrossSplitMode` into a larger full matrix	• LAPACKFullMatrixExt<Number>::LAPACKFullMatrixExt(const LAPACKFullMatrixExt &M11, const LAPACKFullMatrixExt &M12, const LAPACKFullMatrixExt &M21, const LAPACKFullMatrixExt &M22) • LAPACKFullMatrixExt<Number>::LAPACKFullMatrixExt(const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> &col_index_global_to_local_map_for_M, const LAPACKFullMatrixExt &M11, const std::vector<types::global_dof_index> &M11_tau_index_set, const std::vector<types::global_dof_index> &M11_sigma_index_set, const LAPACKFullMatrixExt & M12, const std::vector<types::global_dof_index> &M12_tau_index_set, const std::vector<types::global_dof_index> &M12_sigma_index_set, const LAPACKFullMatrixExt & M21, const std::vector<types::global_dof_index> &M21_tau_index_set, const std::vector<types::global_dof_index> &M21_sigma_index_set, const LAPACKFullMatrixExt & M22, const std::vector<types::global_dof_index> &M22_tau_index_set, const std::vector<types::global_dof_index> &M22_sigma_index_set)

Operator	Functions to be performed	C++ functions
$M = \begin{pmatrix} M_1 & M_2 \end{pmatrix}$ or $M = \begin{pmatrix} M_1 \\ M_2 \end{pmatrix}$	Agglomeration of two full submatrices in either `HorizontalSplitMode` or `VerticalSplitMode` into a larger full matrix	• LAPACKFullMatrixExt<Number>::LAPACKFullMatrixExt(const LAPACKFullMatrixExt &M1, const LAPACKFullMatrixExt &M2, bool is_horizontal_split) • LAPACKFullMatrixExt<Number>::LAPACKFullMatrixExt(const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> & col_index_global_to_local_map_for_M, const LAPACKFullMatrixExt &M1, const std::vector<types::global_dof_index> & M1_tau_index_set, const std::vector<types::global_dof_index> &M1_sigma_index_set, const LAPACKFullMatrixExt & M2, const std::vector<types::global_dof_index> & M2_tau_index_set, const std::vector<types::global_dof_index> &M2_sigma_index_set, bool is_horizontal_split)

Operator	Functions to be performed	C++ functions
$\mathcal{T}^{\mathcal{R}}_{r,\text{pairw}}(M) = \mathcal{T}^{\mathcal{R}}_{r,\text{pairw}}\left(\begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix}\right) = M_1\vert^b + M_2\vert^b + M_3\vert^b + M_4\vert^b$	Agglomeration of four rank-k sub-matrices into a larger rank-k matrix using pairwise formatted addition	• RkMatrix<Number>::RkMatrix(const size_type fixed_rank_k, const RkMatrix<Number> &M11, const RkMatrix<Number> &M12, const RkMatrix<Number> &M21, const RkMatrix<Number> &M22) • RkMatrix<Number>::RkMatrix(const size_type fixed_rank_k, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> &col_index_global_to_local_map_for_M, const RkMatrix<Number> &M11, const std::vector<types::global_dof_index> &M11_tau_index_set, const std::vector<types::global_dof_index> &M11_sigma_index_set, const RkMatrix<Number> & M12, const std::vector<types::global_dof_index> &M12_tau_index_set, const std::vector<types::global_dof_index> &M12_sigma_index_set, const RkMatrix<Number> & M21, const std::vector<types::global_dof_index> &M21_tau_index_set, const std::vector<types::global_dof_index> &M21_sigma_index_set, const RkMatrix<Number> & M22, const std::vector<types::global_dof_index> &M22_tau_index_set, const std::vector<types::global_dof_index> &M22_sigma_index_set)

Operator	Functions to be performed	C++ functions
$\mathcal{T}_r^{\mathcal{R}}(M) = \mathcal{T}_r^{\mathcal{R}}\left(\begin{pmatrix} M_1 & M_2 \end{pmatrix}\right) = M_1\vert^b + M_2\vert^b$ or $\mathcal{T}_r^{\mathcal{R}}(M) = \mathcal{T}_r^{\mathcal{R}}\left(\begin{pmatrix} M_1 \\ M_2 \end{pmatrix}\right) = M_1\vert^b + M_2\vert^b$	Agglomeration of two rank-k submatrices in either `HorizontalSplitMode` or `VerticalSplitMode` into a larger rank-k matrix using formatted addition	• RkMatrix<Number>::RkMatrix(const size_type fixed_rank_k, const RkMatrix<Number> &M1, const RkMatrix<Number> &M2, bool is_horizontal_split) • RkMatrix<Number>::RkMatrix(const size_type fixed_rank_k, const std::map<types::global_dof_index, size_t> &row_index_global_to_local_map_for_M, const std::map<types::global_dof_index, size_t> & col_index_global_to_local_map_for_M, const RkMatrix<Number> &M1, const std::vector<types::global_dof_index> & M1_tau_index_set, const std::vector<types::global_dof_index> &M1_sigma_index_set, const RkMatrix<Number> & M2, const std::vector<types::global_dof_index> & M2_tau_index_set, const std::vector<types::global_dof_index> &M2_sigma_index_set, bool is_horizontal_split)
$\mathcal{T}^{\mathcal{F}\leftarrow\mathcal{R}}$	Convert a rank-k matrix to a full matrix (for verification purpose only)	RkMatrix<Number>::convertToFullMatrix(LAPACKFullMatrixExt<Number> &matrix) const
$\mathcal{T}^{\mathcal{F}\leftarrow\mathcal{H}}$	Convert an $\mathcal{H}$-matrix to a full matrix (for verification purpose only)	HMatrix<spacedim, Number>::convertToFullMatrix(MatrixType &M) const
$\mathcal{T}_r^{\mathcal{R}\leftarrow\mathcal{H}}$ $(Convert\_H)$	Convert an $\mathcal{H}$-matrix to a rank-k matrix	convertHMatBlockToRkMatrix(HMatrix<spacedim, Number> * hmat_block, const unsigned int fixed_rank_k, const HMatrix<spacedim, Number> *hmat_root_block, size_t * calling_counter, const std::string &output_file_base_name)

Operator	Functions to be performed	C++ functions
$\mathcal{T}^{\mathcal{H}\leftarrow\mathcal{H}}_{P'\leftarrow P}$ with $T(I\times J,P')\subset T(I\times J,P)$	Coarsen an $\mathcal{H}$-matrix to a subtree $T(I\times J,P')$ or the partition P'	• HMatrix<spacedim, Number>::coarsen_to_subtree(const BlockClusterTree<spacedim, Number> &subtree, const unsigned int fixed_rank_k) • HMatrix<spacedim, Number>::coarsen_to_partition(const std::vector< typename BlockClusterTree<spacedim, Number>::node_pointer_type> &partition, const unsigned int fixed_rank_k)
$\mathcal{T}^{\mathcal{H}\leftarrow\mathcal{H}}_{P'\leftarrow P}$ with $T(I\times J,P')\supset T(I\times J,P)$	Refine an $\mathcal{H}$-matrix to an extended block cluster tree with the finer partition P'	HMatrix<spacedim, Number>::refine_to_supertree()
$\mathcal{T}^{\mathcal{H}\leftarrow\mathcal{H}}_{P'\leftarrow P}$ with $T(I\times J,P')\not\subset T(I\times J,P)$ and $T(I\times J,P')\not\supset T(I\times J,P)$	Convert an $\mathcal{H}$-matrix to another $\mathcal{H}$-matrix with a different block cluster tree	HMatrix<spacedim, Number>::convert_between_different_block_cluster_trees(BlockClusterTree<spacedim, Number> &bct1, BlockClusterTree<spacedim, Number> &bct2, const unsigned int fixed_rank_k2)
$y\|_\tau = y\|_\tau + M\|_b \cdot x\|_\sigma$	Multiplication of an $\mathcal{H}$-matrix and a block vector	HMatrix<spacedim, Number>::vmult(Vector<Number> & y, const Vector<Number> &x) const
$y\|_\sigma = y\|_\sigma + M^T\|_b \cdot x\|_\tau$	Multiplication of the transpose of an $\mathcal{H}$-matrix and a block vector	HMatrix<spacedim, Number>::Tvmult(Vector<Number> & y, const Vector<Number> &x) const
$M_1 \oplus_r M_2$	Formatted addition of two $\mathcal{H}$-matrices	HMatrix<spacedim, Number>::add(HMatrix<spacedim, Number> & C, const HMatrix<spacedim, Number> &B, const size_type fixed_rank_k) const

Chapter 2

Definition of terms

- $\mathcal{H}$-matrix node: In the current C++ implementation of $\mathcal{H}$-matrix, it is represented as a hierarchy of linked `HMatrix` objects, which are organized in a tree structure being the same as the associated block cluster tree (`BlockClusterTree`). Then an $\mathcal{H}$-matrix node is one of these `HMatrix` objects, which is a node in the tree.
- Minimum matrix dimension: Let $M \in \mathbb{R}^{m \times n}$, then the minimum matrix dimension is $\min\{m, n\}$. Then $\mathrm{rank}(M) \leq \min\{m, n\}$. This term is frequently used in the implementation of `RkMatrix` and singular value decomposition (`LAPACKFullMatrixExt::svd` and `LAPACKFullMatrixExt::reduced_svd`) of a full matrix `LAPACKFullMatrixExt`.
- Global matrix: the matrix defined on the complete index set $I \times J$.
- Local matrix: the matrix defined on a block cluster $\tau \times \sigma$, which is a subset of the complete index set $I \times J$.
- Formal rank of a rank-k matrix: it is the number of columns in A or B.
- Long matrix: the matrix has more number of rows than columns.
- Wide matrix: the matrix has more number of columns than rows.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

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

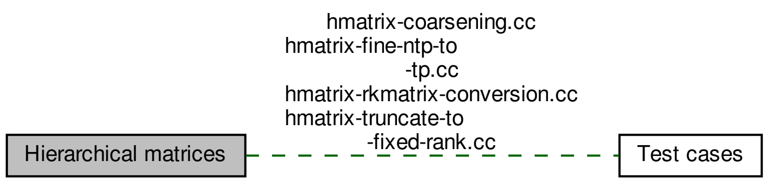
Chapter 7

Module Documentation

7.1 Hierarchical matrices

Implementation of hierarchical matrix data structure and algebraic operations.

Collaboration diagram for Hierarchical matrices:



Files

- file block_cluster.h

  Implementation of the class BlockCluster.
- file block_cluster_tree.h

  Implementation of the class BlockClusterTree.
- file cluster.h

  Implementation of the class Cluster.
- file hmatrix.h

  Definition of hierarchical matrix.
- file rkmatrix.h

  Definition of rank-k matrix.
- file tree.h

  Implementation of the classes for binary tree node, general tree node and functions for manipulating the trees constructed from these nodes.
- file block-cluster-tree.cc

  This files verifies the admissible block cluster partition.

- file cluster-diameter.cc

  This files verifies the cluster diameter and pair-wise distance calculation using a 3x3 grid in a square.
- file cluster-tree-hp.cc

  Test the construction of a $\mathcal{H}^p$ cluster tree.
- file hmatrix-add-formatted.cc

  Verify formatted addition of two h-matrices.
- file hmatrix-bct-struct-with-rank.cc

  Visualize the block cluster tree structure of an H-matrix with displayed rank.
- file hmatrix-coarsening.cc

  Coarsen a H-matrix to its subtree.
- file hmatrix-fine-ntp-to-tp.cc

  Verify the conversion of an $\mathcal{H}$-matrix from fine non-tensor product structure to tensor product structure.
- file hmatrix-rkmatrix-conversion.cc

  Verify the conversion from an H-matrix to a rank-k matrix.
- file hmatrix-truncate-to-fixed-rank.cc

  Verify the truncation of an HMatrix to an RkMatrix.
- file hp-matrix.cc

  Test the H^p matrix.
- file rkmatrix-agglomeration.cc

  Verify the agglomeration of four rank-k submatrices into a larger rank-k matrix.

Classes

- class BlockCluster< spacedim, Number >

  Class for block cluster.
- class BlockClusterTree< spacedim, Number >

  Class for block cluster tree.
- class BinaryTreeNode< T >

  Class for binary tree node.

- template<int dim, int spacedim, typename Number = double>
  void map_dofs_to_average_cell_size (const DoFHandler< dim, spacedim > &dof_handler, std::vector< Number > &dof_average_cell_size)
- template<typename DoFHandlerType , typename Number = double>
  void map_dofs_to_max_cell_size (const DoFHandlerType &dof_handler, std::vector< Number > &dof_↩
  max_cell_size)
- template<typename DoFHandlerType , typename Number = double>
  void map_dofs_to_min_cell_size (const DoFHandlerType &dof_handler, std::vector< Number > &dof_min↩
  _cell_size)
- template<int spacedim, typename Number >
  std::ostream & operator<< (std::ostream &out, const Cluster< spacedim, Number > &cluster)
- template<int spacedim, typename Number = double>
  Number calc_cluster_distance (const Cluster< spacedim, Number > &cluster1, const Cluster< spacedim, Number > &cluster2, const std::vector< Point< spacedim, Number >> &all_support_points)
- template<int spacedim, typename Number = double>
  Number calc_cluster_distance (const Cluster< spacedim, Number > &cluster1, const Cluster< spacedim, Number > &cluster2, const std::vector< Point< spacedim, Number >> &all_support_points, const std↩
  ::vector< Number > &cell_size_at_dofs)
- template<int spacedim, typename Number >
  bool **operator==** (const Cluster< spacedim, Number > &cluster1, const Cluster< spacedim, Number > &cluster2)

7.1.1 Detailed Description

Implementation of hierarchical matrix data structure and algebraic operations.

7.1.2 Function Documentation

7.1.2.1 calc_cluster_distance() [1/2]

```
template<int spacedim, typename Number = double>
Number calc_cluster_distance (
            const Cluster< spacedim, Number > & cluster1,
            const Cluster< spacedim, Number > & cluster2,
            const std::vector< Point< spacedim, Number >> & all_support_points )
```

Calculate the minimum distance between two clusters. This calculation has no mesh size correction.

The calculation is based on measuring the distance between each pair of support points contained in the clusters, which prevents the distance calculation between two support sets.

Parameters

cluster1	
cluster2	
all_support_points	A list of support point coordinates which are ordered by DoF indices.

Returns

References Cluster< spacedim, Number >::get_index_set().

7.1.2.2 calc_cluster_distance() [2/2]

```
template<int spacedim, typename Number = double>
Number calc_cluster_distance (
            const Cluster< spacedim, Number > & cluster1,
            const Cluster< spacedim, Number > & cluster2,
            const std::vector< Point< spacedim, Number >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs )
```

Calculate the minimum distance between two clusters. This calculation has the mesh size correction.

The calculation is based on measuring the distance between each pair of support points contained in the clusters, which prevents the distance calculation between two support sets.

Parameters

cluster1	
cluster2	
all_support_points	A list of support point coordinates which are ordered by DoF indices.
cell_size_at_dofs	The list of estimated cell size values at DoF support points.

Returns

Calculate the uncorrected cluster distance.

Get the maximum diameter of the support sets for different DoFs, which is 2 times of the cell size associated with the corresponding DoF.

Ensure the positivity of the returned cluster distance.

References Cluster< spacedim, Number >::get_index_set().

7.1.2.3 map_dofs_to_average_cell_size()

```
template<int dim, int spacedim, typename Number = double>
void map_dofs_to_average_cell_size (
            const DoFHandler< dim, spacedim > & dof_handler,
            std::vector< Number > & dof_average_cell_size )
```

Calculate the average cell sizes associated with those DoFs handled by the given DoF handler object.

The value doubled is used as an estimate for the diameter of the support set of each DoF.

Parameters

dof_average_cell_size	The returned list of average cell sizes. The memory for this vector should be preallocated and initialized to zero before calling this function.

Create the vector which stores the number of cells that share a common DoF for each DoF.

Get the diameter of the current cell.

Get DoF indices local to this cell.

Referenced by main().

7.1.2.4 map_dofs_to_max_cell_size()

```
template<typename DoFHandlerType , typename Number = double>
void map_dofs_to_max_cell_size (
            const DoFHandlerType & dof_handler,
            std::vector< Number > & dof_max_cell_size )
```

Calculate the maximum cell sizes associated with those DoFs handled by the given DoF handler object.

The value doubled is used as an estimate for the diameter of the support set of each DoF.

Parameters

dof_max_cell_size	The returned list of maximum cell sizes. The memory for this vector should be preallocated and initialized to zero before calling this function.

Get the diameter of the current cell.

Get DoF indices local to this cell.

Referenced by main().

7.1.2.5 map_dofs_to_min_cell_size()

```
template<typename DoFHandlerType , typename Number = double>
void map_dofs_to_min_cell_size (
            const DoFHandlerType & dof_handler,
            std::vector< Number > & dof_min_cell_size )
```

Calculate the minimum cell sizes associated with those DoFs handled by the given DoF handler object.

The value doubled is used as an estimate for the diameter of the support set of each DoF.

Parameters

dof_min_cell_size	The returned list of average cell sizes. The memory for this vector should be preallocated and initialized to zero before calling this function.

Get the diameter of the current cell.

Get DoF indices local to this cell.

Referenced by main().

7.1.2.6 operator<<()

```
template<int spacedim, typename Number >
std::ostream& operator<< (
```

```
std::ostream & out,
const Cluster< spacedim, Number > & cluster )
```

Print out the cluster data.

Parameters

out	
cluster	

Returns

7.2 Sauter quadrature

Implementation of Stefan Sauter's method about 4D numerical quadrature which aims to handle singularity.

Files

- file laplace_bem.h

  Implementation of BEM involving kernel functions and singular numerical quadratures.

7.2.1 Detailed Description

Implementation of Stefan Sauter's method about 4D numerical quadrature which aims to handle singularity.

7.3 Toolbox

This is my toolbox for assisting program debugging, function verification and information visualization.

Files

- file debug_tools.h

    This file includes a bunch of helper functions for printing out and visualizing information about grid, DoFs, map, etc.

7.3.1 Detailed Description

This is my toolbox for assisting program debugging, function verification and information visualization.

7.4 Linear algebra

Linear algebra computation.

Files

- file lapack_helpers.h

  Exposes LAPACK helper functions defined in lapack_full_matrix.cc and define new ones by following them as examples.
- file lapack-matrix-agglomeration.cc

  Verify the agglomeration of four full submatrices into a larger full matrix.
- file delete-rows-and-columns.cc

  Test deleting rows and columns as well as keeping the first n rows or columns from a LAPACKFullMatrixExt. .
- file lapack-matrix-fill.cc

  Verify filling a LAPACKFullMatrixExt from a source matrix.
- file hstack-vstack.cc

  Verify horizontal and vertical stacking of two LAPACKFullMatrixExt objects.
- file scale-rows-and-columns.cc

  Test scaling rows and columns of a LAPACKFullMatrixExt, which is actually left and right multiplication with a diagonal matrix.
- file svd.cc

  Test singular value decomposition (SVD) and reduced SVD.
- file svd-degenerate-cases.cc

  Test SVD and RSVD for degenerate cases, such as the matrix is a scalar, row vector or column vector.
- file transpose.cc

  Test in-place transpose of a LAPACKFullMatrixExt.
- file rkmatrix.cc

  Test RkMatrix class.
- file rkmatrix-add-formatted-using-qr.cc

  Verify the formatted addition of two rank-k matrices by using the QR decomposition. This requires that the component matrices of rank-k matrices should wide matrix, i.e. having more rows than columns.

7.4.1 Detailed Description

Linear algebra computation.

7.5 Programming techniques

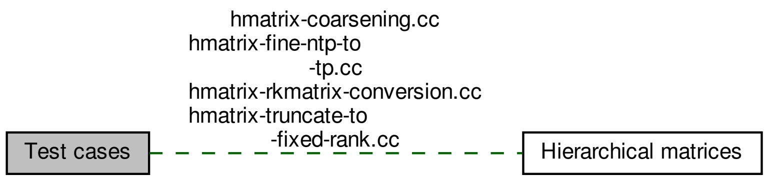
Programming techniques for C++ and software engineering.

Programming techniques for C++ and software engineering.

7.6 Test cases

Test cases for verifying code design and algorithms.

Collaboration diagram for Test cases:



Files

- file bct-extend-to-finer-partition.cc

  Verify extend a block cluster tree to a given finer partition.
- file bct-overloaded-assignment.cc

  Verify the deep and shallow overloaded assignment operators for `BlockClusterTree`.
- file fullmatrix-hmatrix-mmult.cc

  Verify the full matrix/H-matrix multiplication.
- file hmatrix-coarsening.cc

  Coarsen a H-matrix to its subtree.
- file hmatrix-deep-copy-constructor.cc

  Verify the deep copy constructor of $\mathcal{H}$-matrix.
- file hmatrix-fine-ntp-to-tp.cc

  Verify the conversion of an $\mathcal{H}$-matrix from fine non-tensor product structure to tensor product structure.
- file hmatrix-fullmatrix-mmult.cc

  Verify the H-matrix/full matrix multiplication.
- file hmatrix-hmatrix-mmult-all-coarse-ntp.cc

  Verify the multiplication of two $\mathcal{H}$-matrices. Both operands and the result matrices have the coarse non-tensor product partitions.
- file hmatrix-hmatrix-mmult-all-fine-ntp.cc

  Verify the multiplication of two $\mathcal{H}$-matrices. Both operands and the result matrices have the fine non-tensor product partitions.
- file hmatrix-hmatrix-mmult-coarse-coarse-fine-ntp.cc

  Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have coarse-coarse-fine non-tensor product partitions.
- file hmatrix-hmatrix-mmult-coarse-fine-coarse-ntp.cc

  Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have coarse-fine-coarse non-tensor product partitions.
- file hmatrix-hmatrix-mmult-fine-coarse-fine-ntp.cc

  Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have fine-coarse-fine non-tensor product partitions.
- file hmatrix-hmatrix-mmult-fine-fine-coarse-ntp.cc

  Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have fine-fine-coarse non-tensor product partitions.
- file hmatrix-overloaded-deep-assignment.cc

Verify the overloaded deep assignment operator of $\mathcal{H}$-matrix.

- file hmatrix-overloaded-shallow-assignment.cc

  Verify the overloaded shallow assignment operator of $\mathcal{H}$-matrix.

- file hmatrix-refinement.cc

  Verify the refinement of an $\mathcal{H}$-matrix hierarchy with respect to its extended block cluster tree.

- file hmatrix-rkmatrix-conversion.cc

  Verify the conversion from an H-matrix to a rank-k matrix.

- file hmatrix-rkmatrix-mmult.cc

  Verify the H-matrix/rank-k matrix multiplication.

- file hmatrix-shallow-copy-constructor.cc

  Verify the deep copy constructor of $\mathcal{H}$-matrix.

- file hmatrix-truncate-to-fixed-rank.cc

  Verify the truncation of an HMatrix to an RkMatrix.

- file lapack-matrix-agglomeration-interwoven-indices.cc

  Verify the agglomeration of four full submatrices into a larger full matrix when the index sets of several child clusters are interwoven together into the index set of the parent cluster.

- file lapack-matrix-agglomeration-of-two-submatrices.cc

  Verify the agglomeration of two full submatrices which have been obtained from horizontal or vertical splitting.

- file lapack-matrix-agglomeration-of-two-submatrices-interwoven-indices.cc

  Verify the agglomeration of two full submatrices which have been obtained from horizontal or vertical splitting. The index sets of several child clusters are interwoven together into the index set of the parent cluster.

- file lapack-matrix-mmult.cc

  Verify the multiplication of two `LAPACKFullMatrixExt`.

- file rkmatrix-agglomeration-interwoven-indices.cc

  Verify the agglomeration of four rank-k submatrices into a larger rank-k matrix when the index sets of several child clusters are interwoven together into the index set of the parent cluster.

- file rkmatrix-agglomeration-of-two-submatrices.cc

  Verify the agglomeration of two rank-k submatrices which have been obtained from horizontal or vertical splitting.

- file rkmatrix-agglomeration-of-two-submatrices-interwoven-indices.cc

  Verify the agglomeration of two rank-k submatrices which have been obtained from horizontal or vertical splitting. The index sets of several child clusters are interwoven together into the index set of the parent cluster.

- file rkmatrix-hmatrix-mmult.cc

  Verify the rank-k/H-matrix matrix multiplication.

7.6.1 Detailed Description

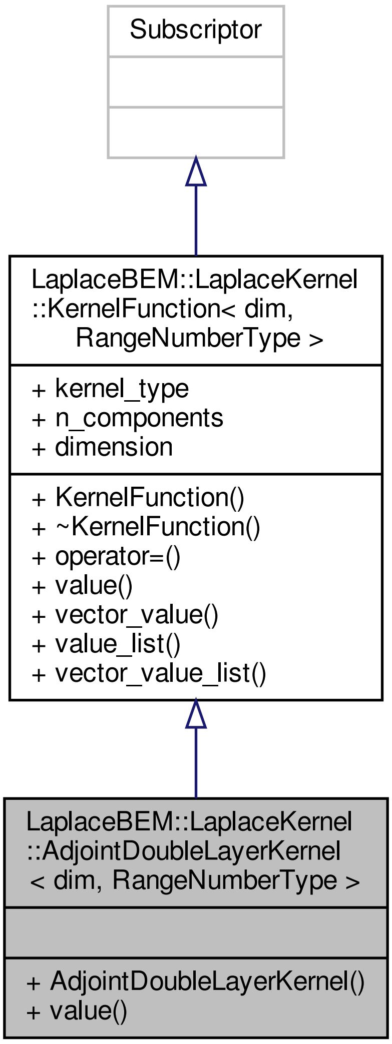
Test cases for verifying code design and algorithms.

Chapter 8

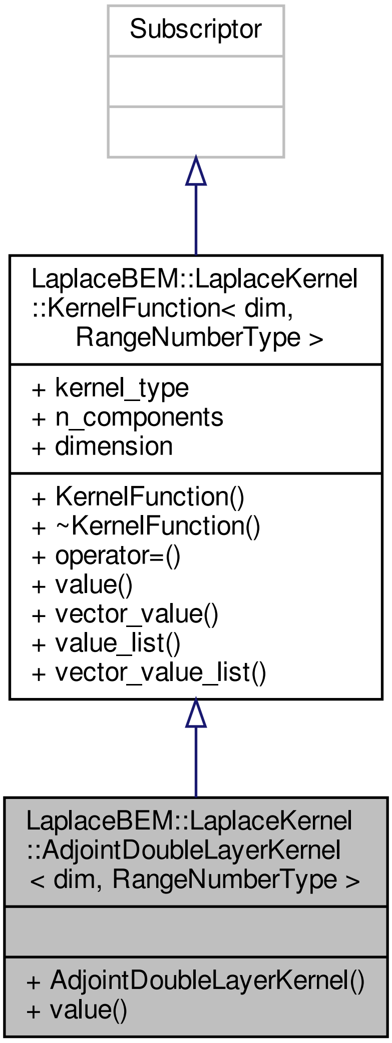
Class Documentation

8.1 LaplaceBEM::LaplaceKernel::AdjointDoubleLayerKernel< dim, RangeNumberType > Class Template Reference

Inheritance diagram for LaplaceBEM::LaplaceKernel::AdjointDoubleLayerKernel< dim, RangeNumberType >:



Collaboration diagram for LaplaceBEM::LaplaceKernel::AdjointDoubleLayerKernel< dim, RangeNumberType >:



Public Member Functions

- virtual RangeNumberType **value** (const Point< dim > &x, const Point< dim > &y, const Tensor< 1, dim > &nx, const Tensor< 1, dim > &ny, const unsigned int component=0) const override

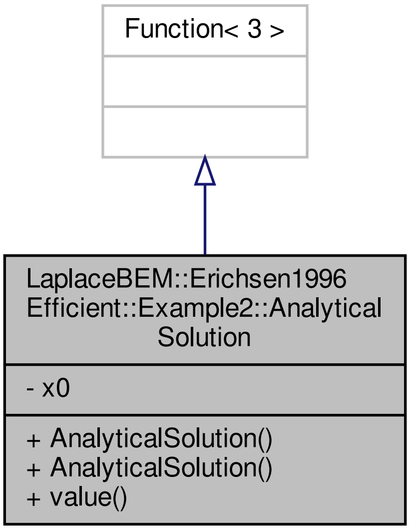
Additional Inherited Members

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

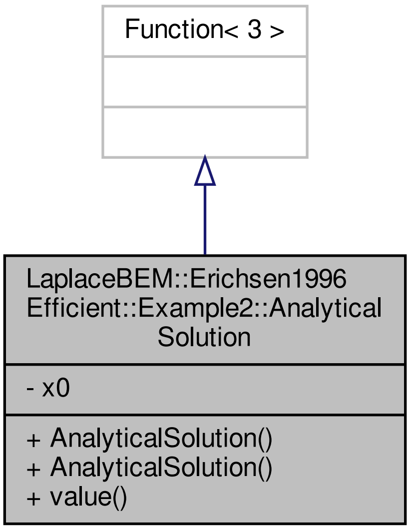
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.2 LaplaceBEM::Erichsen1996Efficient::Example2::AnalyticalSolution Class Reference

Inheritance diagram for LaplaceBEM::Erichsen1996Efficient::Example2::AnalyticalSolution:



Collaboration diagram for LaplaceBEM::Erichsen1996Efficient::Example2::AnalyticalSolution:



Public Member Functions

- **AnalyticalSolution** (const Point< 3 > &x0)
- double **value** (const Point< 3 > &p, const unsigned int component=0) const

Private Attributes

- Point< 3 > x0

8.2.1 Member Data Documentation

8.2.1.1 x0

```
Point<3> LaplaceBEM::Erichsen1996Efficient::Example2::AnalyticalSolution::x0 [private]
```

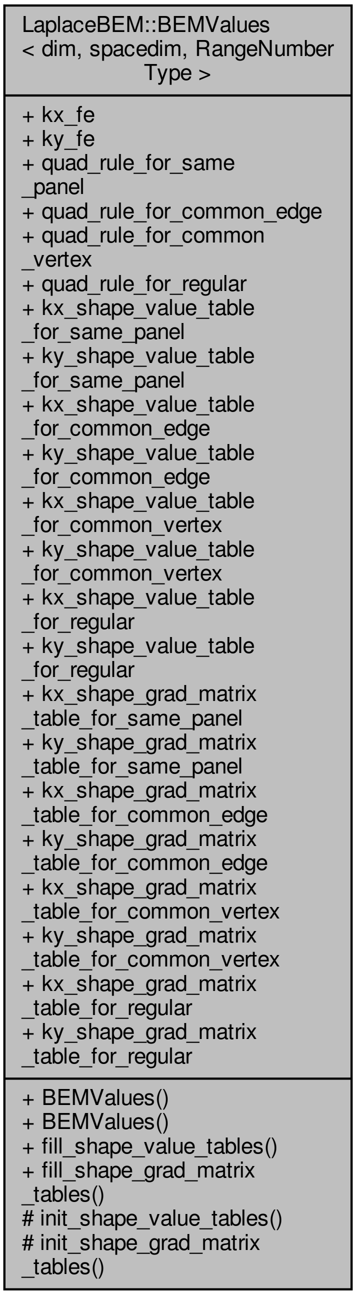
Location of the point source.

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

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/erichsen1996efficient_↩ example2.h

8.3 LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType > Class Template Reference

Collaboration diagram for LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >:



Public Member Functions

- **BEMValues** (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &quad_rule_for_same_panel, const QGauss< 4 > &quad_rule_for_common_edge, const QGauss< 4 > &quad_rule_for_common_vertex, const QGauss< 4 > &quad_rule_for_regular)

- BEMValues (const BEMValues< dim, spacedim, RangeNumberType > &bem_values)
- void **fill_shape_value_tables** ()
- void **fill_shape_grad_matrix_tables** ()

Public Attributes

- const FiniteElement< dim, spacedim > & **kx_fe**
- const FiniteElement< dim, spacedim > & **ky_fe**
- const QGauss< 4 > & **quad_rule_for_same_panel**
- const QGauss< 4 > & **quad_rule_for_common_edge**
- const QGauss< 4 > & **quad_rule_for_common_vertex**
- const QGauss< 4 > & **quad_rule_for_regular**
- Table< 3, RangeNumberType > **kx_shape_value_table_for_same_panel**
- Table< 3, RangeNumberType > **ky_shape_value_table_for_same_panel**
- Table< 3, RangeNumberType > **kx_shape_value_table_for_common_edge**
- Table< 3, RangeNumberType > **ky_shape_value_table_for_common_edge**
- Table< 3, RangeNumberType > **kx_shape_value_table_for_common_vertex**
- Table< 3, RangeNumberType > **ky_shape_value_table_for_common_vertex**
- Table< 3, RangeNumberType > **kx_shape_value_table_for_regular**
- Table< 3, RangeNumberType > **ky_shape_value_table_for_regular**
- Table< 2, FullMatrix< RangeNumberType > > **kx_shape_grad_matrix_table_for_same_panel**
- Table< 2, FullMatrix< RangeNumberType > > **ky_shape_grad_matrix_table_for_same_panel**
- Table< 2, FullMatrix< RangeNumberType > > **kx_shape_grad_matrix_table_for_common_edge**
- Table< 2, FullMatrix< RangeNumberType > > **ky_shape_grad_matrix_table_for_common_edge**
- Table< 2, FullMatrix< RangeNumberType > > **kx_shape_grad_matrix_table_for_common_vertex**
- Table< 2, FullMatrix< RangeNumberType > > **ky_shape_grad_matrix_table_for_common_vertex**
- Table< 2, FullMatrix< RangeNumberType > > **kx_shape_grad_matrix_table_for_regular**
- Table< 2, FullMatrix< RangeNumberType > > **ky_shape_grad_matrix_table_for_regular**

Protected Member Functions

- void **init_shape_value_tables** ()
- void **init_shape_grad_matrix_tables** ()

8.3.1 Constructor & Destructor Documentation

8.3.1.1 BEMValues()

```
template<int dim, int spacedim, typename RangeNumberType >
LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues (
            const BEMValues< dim, spacedim, RangeNumberType > & bem_values )
```

Copy constructor for class `BEMValues`.

Parameters

bem_values	

References LaplaceBEM::bem_shape_grad_matrices_common_edge(), LaplaceBEM::bem_shape_grad_↩ matrices_common_vertex(), LaplaceBEM::bem_shape_grad_matrices_regular(), LaplaceBEM::bem_shape_↩ grad_matrices_same_panel(), LaplaceBEM::bem_shape_values_common_edge(), LaplaceBEM::bem_shape_↩ values_common_vertex(), LaplaceBEM::bem_shape_values_regular(), and LaplaceBEM::bem_shape_values_↩ same_panel().

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

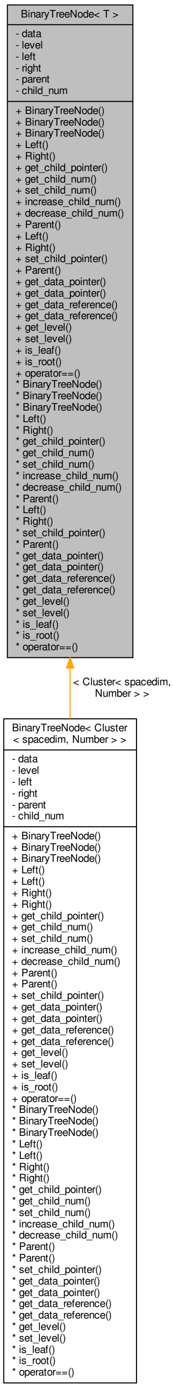
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.4 BinaryTreeNode< T > Class Template Reference

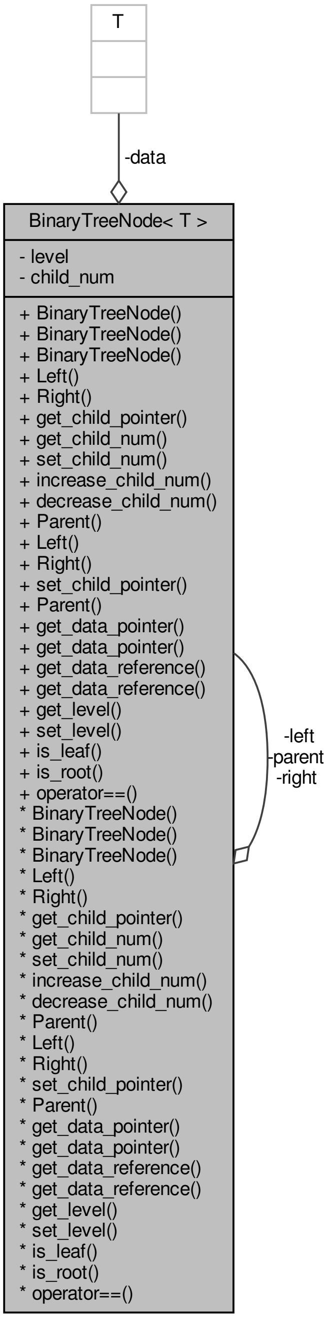
Class for binary tree node.

```
#include <tree.h>
```

Inheritance diagram for BinaryTreeNode< T >:



Collaboration diagram for BinaryTreeNode< T >:



Public Member Functions

- BinaryTreeNode ()
- BinaryTreeNode (const BinaryTreeNode &node)

- BinaryTreeNode (const T &data, unsigned int level, BinaryTreeNode *left=nullptr, BinaryTreeNode *right=nullptr, BinaryTreeNode *parent=nullptr)
- BinaryTreeNode * Left (void) const
- BinaryTreeNode * Right (void) const
- BinaryTreeNode * get_child_pointer (std::size_t index) const
- unsigned int get_child_num () const
- void set_child_num (const unsigned int child_num)
- void increase_child_num (const unsigned int incr_num=1)
- void decrease_child_num (const unsigned int decr_num=1)
- BinaryTreeNode * Parent (void) const
- void Left (const BinaryTreeNode *node_pointer)
- void Right (const BinaryTreeNode *node_pointer)
- void set_child_pointer (std::size_t index, const BinaryTreeNode *node_pointer)
- void Parent (const BinaryTreeNode *node_pointer)
- T * get_data_pointer ()
- const T * get_data_pointer () const
- T & get_data_reference ()
- const T & get_data_reference () const
- unsigned int get_level () const
- void set_level (const unsigned int level)
- bool is_leaf () const
- bool is_root () const
- bool operator== (const BinaryTreeNode< T > &node) const

Private Attributes

- T **data**
- unsigned int **level**
- BinaryTreeNode * **left**
- BinaryTreeNode * **right**
- BinaryTreeNode * **parent**
- unsigned int child_num

8.4.1 Detailed Description

template<typename T>
class BinaryTreeNode< T >

Class for binary tree node.

In the implementation of hierarchical matrices, a binary tree node is designed to hold a cluster (Cluster) in a cluster tree (ClusterTree) along with two pointers `left` and `right` pointing to the two children belong to the node itself. The template argument `T` should be assigned the type of the cluster.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 BinaryTreeNode() [1/3]

```
template<typename T >
BinaryTreeNode< T >::BinaryTreeNode ( )
```

Default constructor.

8.4.2.2 BinaryTreeNode() [2/3]

```
template<typename T >
BinaryTreeNode< T >::BinaryTreeNode (
            const BinaryTreeNode< T > & node )
```

Copy constructor.

8.4.2.3 BinaryTreeNode() [3/3]

```
template<typename T>
BinaryTreeNode< T >::BinaryTreeNode (
            const T & data,
            unsigned int level,
            BinaryTreeNode< T > * left = nullptr,
            BinaryTreeNode< T > * right = nullptr,
            BinaryTreeNode< T > * parent = nullptr )
```

Constructor from the given data.

N.B. The data of type `T` will be copied into the created node. Increment the `child_num` of the parent node.

8.4.3 Member Function Documentation

8.4.3.1 decrease_child_num()

```
template<typename T >
void BinaryTreeNode< T >::decrease_child_num (
            const unsigned int decr_num = 1 )
```

Decrease the total number of children.

Returns

8.4.3.2 get_child_num()

```
template<typename T >
unsigned int BinaryTreeNode< T >::get_child_num ( ) const
```

Get the total number of nonempty children.

Referenced by GetTreeLeaves(), and PrintTreeNode().

8.4.3.3 get_child_pointer()

```
template<typename T >
BinaryTreeNode< T > * BinaryTreeNode< T >::get_child_pointer (
            std::size_t index ) const
```

Get the pointer to the `index'th` child.

Parameters

index	Index of the child

Returns

8.4.3.4 get_data_pointer() [1/2]

```
template<typename T >
T * BinaryTreeNode< T >::get_data_pointer ( )
```

Get the pointer to the node data.

Referenced by ClusterTree< spacedim, Number >::partition_from_cluster_node().

8.4.3.5 get_data_pointer() [2/2]

```
template<typename T >
const T * BinaryTreeNode< T >::get_data_pointer ( ) const
```

Get the pointer to the node data (const version).

8.4.3.6 get_data_reference() [1/2]

```
template<typename T >
T & BinaryTreeNode< T >::get_data_reference ( )
```

Get the reference to the node data.

Referenced by CopyTree(), and PrintTreeNode().

8.4.3.7 get_data_reference() [2/2]

```
template<typename T >
const T & BinaryTreeNode< T >::get_data_reference ( ) const
```

Get the reference to the node data (const version).

8.4.3.8 get_level()

```
template<typename T >
unsigned int BinaryTreeNode< T >::get_level ( ) const
```

Get the level of the node in the tree.

Referenced by CopyTree(), ClusterTree< spacedim, Number >::partition_from_cluster_node(), and PrintTree↩
Node().

8.4.3.9 increase_child_num()

```
template<typename T >
void BinaryTreeNode< T >::increase_child_num (
             const unsigned int incr_num = 1 )
```

Increase the total number of children.

Referenced by CopyTree().

8.4.3.10 is_leaf()

```
template<typename T >
bool BinaryTreeNode< T >::is_leaf ( ) const
```

Return whether the current binary tree node is a leaf.

Returns

8.4.3.11 is_root()

```
template<typename T >
bool BinaryTreeNode< T >::is_root ( ) const
```

Return whether the current binary tree node is the root.

Returns

8.4.3.12 Left() [1/2]

```
template<typename T >
BinaryTreeNode< T > * BinaryTreeNode< T >::Left (
            void ) const
```

Get the pointer to the left child node.

Referenced by calc_depth(), CopyTree(), CountTreeNodes(), BinaryTreeNode< Cluster< spacedim, Number > >::get_child_pointer(), GetTreeLeaves(), Inorder(), ClusterTree< spacedim, Number >::partition_from_cluster_↩ node(), Postorder(), Preorder(), and BinaryTreeNode< Cluster< spacedim, Number > >::set_child_pointer().

8.4.3.13 Left() [2/2]

```
template<typename T >
void BinaryTreeNode< T >::Left (
            const BinaryTreeNode< T > * node_pointer )
```

Set the left child node pointer.

N.B. The const pointer type in the argument will be converted to non-const pointer type. In this way, even a const node can be added to the tree.

8.4.3.14 operator==()

```
template<typename T>
bool BinaryTreeNode< T >::operator== (
            const BinaryTreeNode< T > & node ) const
```

Check the equality of two binary tree nodes by comparing the contained data.

Parameters

node	

Returns

8.4.3.15 Parent() [1/2]

```
template<typename T >
BinaryTreeNode< T > * BinaryTreeNode< T >::Parent (
            void ) const
```

Get the pointer to the parent node.

Referenced by MakeIntExampleTree(), and PrintTreeNode().

8.4.3.16 Parent() [2/2]

```
template<typename T >
void BinaryTreeNode< T >::Parent (
            const BinaryTreeNode< T > * node_pointer )
```

Set the pointer to the parent.

Parameters

node_pointer	

8.4.3.17 Right() [1/2]

```
template<typename T >
BinaryTreeNode< T > * BinaryTreeNode< T >::Right (
            void ) const
```

Get the pointer to the right child node.

Referenced by calc_depth(), CopyTree(), CountTreeNodes(), BinaryTreeNode< Cluster< spacedim, Number > >::get_child_pointer(), GetTreeLeaves(), Inorder(), ClusterTree< spacedim, Number >::partition_from_cluster_↩ node(), Postorder(), Preorder(), and BinaryTreeNode< Cluster< spacedim, Number > >::set_child_pointer().

8.4.3.18 Right() [2/2]

```
template<typename T >
void BinaryTreeNode< T >::Right (
            const BinaryTreeNode< T > * node_pointer )
```

Set the right child node pointer.

N.B. The const pointer type in the argument will be converted to non-const pointer type. In this way, even a const node can be added to the tree.

8.4.3.19 set_child_num()

```
template<typename T >
void BinaryTreeNode< T >::set_child_num (
            const unsigned int child_num )
```

Set the total number of nonempty children.

Parameters

child_num	

8.4.3.20 set_child_pointer()

```
template<typename T >
void BinaryTreeNode< T >::set_child_pointer (
            std::size_t index,
            const BinaryTreeNode< T > * node_pointer )
```

Set the pointer to the `index'th` child.

Parameters

index	Index of the child
node_pointer	Pointer value to be assigned to the specified child.

8.4.3.21 set_level()

```
template<typename T >
void BinaryTreeNode< T >::set_level (
            const unsigned int level )
```

Set the level of the node.

Parameters

level	

8.4.4 Member Data Documentation

8.4.4.1 child_num

```
template<typename T>
unsigned int BinaryTreeNode< T >::child_num [private]
```

Total number of nonempty children.

Referenced by BinaryTreeNode< Cluster< spacedim, Number > >::BinaryTreeNode(), BinaryTreeNode< Cluster< spacedim, Number > >::decrease_child_num(), BinaryTreeNode< Cluster< spacedim, Number > >::get_child_num(), BinaryTreeNode< Cluster< spacedim, Number > >::increase_child_num(), BinaryTree↩ Node< Cluster< spacedim, Number > >::is_leaf(), and BinaryTreeNode< Cluster< spacedim, Number > >::set_child_num().

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

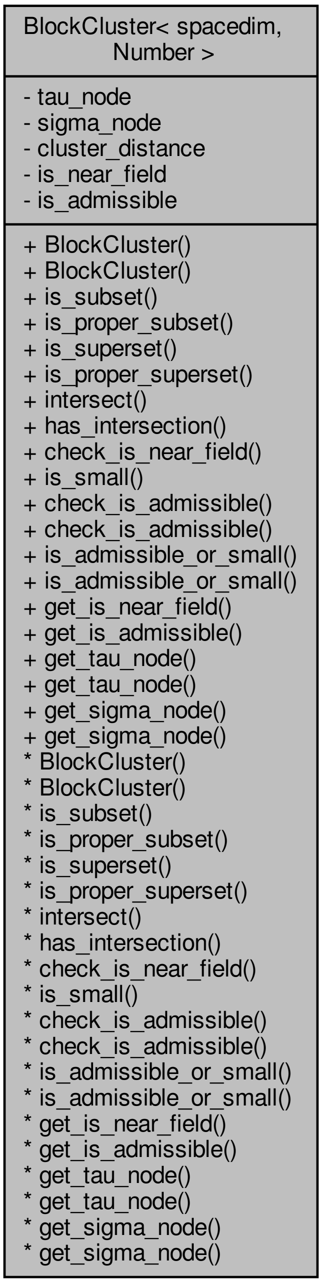
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/tree.h

8.5 BlockCluster< spacedim, Number > Class Template Reference

Class for block cluster.

```
#include <block_cluster.h>
```

Collaboration diagram for BlockCluster< spacedim, Number >:



Public Member Functions

- BlockCluster ()

- BlockCluster (typename ClusterTree< spacedim, Number >::node_pointer_type tau_node, typename ClusterTree< spacedim, Number >::node_pointer_type sigma_node)
- bool is_subset (const BlockCluster &block_cluster) const
- bool is_proper_subset (const BlockCluster &block_cluster) const
- bool is_superset (const BlockCluster &block_cluster) const
- bool is_proper_superset (const BlockCluster &block_cluster) const
- void intersect (const BlockCluster &block_cluster, std::vector< types::global_dof_index > &tau_index_set_intersection, std::vector< types::global_dof_index > &sigma_index_set_intersection) const
- bool has_intersection (const BlockCluster &block_cluster) const
- void check_is_near_field (unsigned int n_min)
- bool is_small (unsigned int n_min)
- void check_is_admissible (Number eta, const std::vector< Point< spacedim, Number >> &all_support_points)
- void check_is_admissible (Number eta, const std::vector< Point< spacedim, Number >> &all_support_points, const std::vector< Number > &cell_size_at_dofs)
- bool is_admissible_or_small (Number eta, const std::vector< Point< spacedim, Number >> &all_support_points, unsigned int n_min)
- bool is_admissible_or_small (Number eta, const std::vector< Point< spacedim, Number >> &all_support_points, const std::vector< Number > &cell_size_at_dofs, unsigned int n_min)
- bool get_is_near_field () const
- bool get_is_admissible () const
- ClusterTree< spacedim, Number >::node_pointer_type **get_tau_node** ()
- ClusterTree< spacedim, Number >::node_const_pointer_type **get_tau_node** () const
- ClusterTree< spacedim, Number >::node_pointer_type **get_sigma_node** ()
- ClusterTree< spacedim, Number >::node_const_pointer_type **get_sigma_node** () const

Private Attributes

- ClusterTree< spacedim, Number >::node_pointer_type tau_node
- ClusterTree< spacedim, Number >::node_pointer_type sigma_node
- Number cluster_distance
- bool is_near_field
- bool is_admissible

Friends

- template<int spacedim1, typename Number1 >
  std::ostream & operator<< (std::ostream &out, const BlockCluster< spacedim1, Number1 > &block_cluster)
- template<int spacedim1, typename Number1 >
  bool **operator==** (const BlockCluster< spacedim1, Number1 > &block_cluster1, const BlockCluster< spacedim1, Number1 > &block_cluster2)
- template<int spacedim1, typename Number1 >
  bool is_equal (const BlockCluster< spacedim, Number > &block_cluster1, const BlockCluster< spacedim, Number > &block_cluster2)

8.5.1 Detailed Description

template<int spacedim, typename Number = double>
class BlockCluster< spacedim, Number >

Class for block cluster.

A block cluster is a Cartesian product of two clusters τ and σ from two cluster trees $T(I)$ and $T(J)$, i.e. $\tau \times \sigma$. This class contains pointers to the cluster tree nodes which hold the data of the two clusters. Because the BlockCluster class only holds pointers to nodes in cluster trees and the ClusterTree class has its own memory management, the BlockCluster class does not need a destroyer.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 BlockCluster() [1/2]

```
template<int spacedim, typename Number >
BlockCluster< spacedim, Number >::BlockCluster ( )
```

Default constructor.

8.5.2.2 BlockCluster() [2/2]

```
template<int spacedim, typename Number >
BlockCluster< spacedim, Number >::BlockCluster (
            typename ClusterTree< spacedim, Number >::node_pointer_type tau_node,
            typename ClusterTree< spacedim, Number >::node_pointer_type sigma_node )
```

Construct from two pointers associated with the nodes in the cluster trees $T(I)$ and $T(J)$.

Parameters

tau_node	The pointer associated to the node in the cluster tree $T(I)$, which holds the cluster τ.
sigma_node	The pointer associated to the node in the cluster tree $T(J)$, which holds the cluster σ.

8.5.3 Member Function Documentation

8.5.3.1 check_is_admissible() [1/2]

```
template<int spacedim, typename Number >
void BlockCluster< spacedim, Number >::check_is_admissible (
            Number eta,
            const std::vector< Point< spacedim, Number >> & all_support_points )
```

Determine if the block cluster is admissible. The admissibility condition is evaluated without mesh cell size correction.

Parameters

eta	Admissibility constant.

Returns

References BlockCluster< spacedim, Number >::cluster_distance, BlockCluster< spacedim, Number >::is_↩ admissible, BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau↩ _node.

Referenced by BlockCluster< spacedim, Number >::is_admissible_or_small().

8.5.3.2 check_is_admissible() [2/2]

```
template<int spacedim, typename Number >
void BlockCluster< spacedim, Number >::check_is_admissible (
            Number eta,
            const std::vector< Point< spacedim, Number >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs )
```

Determine if the block cluster is admissible. The admissibility condition is evaluated with mesh cell size correction.

Parameters

eta	Admissibility constant.

Returns

N.B. The contained clusters τ and σ in the block cluster should be created with the parameter `cell_size_↩ at_dofs`. In this way, the returned cluster diameter is calculated with mesh cell size correction. This is achieved when creating the two cluster trees.

References BlockCluster< spacedim, Number >::cluster_distance, BlockCluster< spacedim, Number >::is_↩ admissible, BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau↩ _node.

8.5.3.3 check_is_near_field()

```
template<int spacedim, typename Number >
void BlockCluster< spacedim, Number >::check_is_near_field (
            unsigned int n_min )
```

Determine if the block cluster belongs to the near field set.

When both contained clusters are large, the block cluster is considered as large.

Parameters

n_min	The size threshold value for determining if a cluster is large.

Returns

References BlockCluster< spacedim, Number >::is_near_field, BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_node.

Referenced by BlockCluster< spacedim, Number >::is_admissible_or_small(), and BlockCluster< spacedim, Number >::is_small().

8.5.3.4 get_is_admissible()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::get_is_admissible ( ) const
```

Get the boolean value whether the block cluster is admissible.

References BlockCluster< spacedim, Number >::is_admissible, BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_node.

8.5.3.5 get_is_near_field()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::get_is_near_field ( ) const
```

Get the boolean value whether the block cluster is near field.

References BlockCluster< spacedim, Number >::is_near_field.

8.5.3.6 has_intersection()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::has_intersection (
            const BlockCluster< spacedim, Number > & block_cluster ) const
```

Determine if the index set of the current block cluster has a nonempty intersection with the index set of the given block cluster.

Returns

References BlockCluster< spacedim, Number >::intersect().

8.5.3.7 intersect()

```
template<int spacedim, typename Number >
void BlockCluster< spacedim, Number >::intersect (
            const BlockCluster< spacedim, Number > & block_cluster,
            std::vector< types::global_dof_index > & tau_index_set_intersection,
            std::vector< types::global_dof_index > & sigma_index_set_intersection ) const
```

Calculate the intersection of the index sets of the current and the given block clusters.

References BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_↩
node.

Referenced by BlockCluster< spacedim, Number >::has_intersection().

8.5.3.8 is_admissible_or_small() [1/2]

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::is_admissible_or_small (
            Number eta,
            const std::vector< Point< spacedim, Number >> & all_support_points,
            unsigned int n_min )
```

Determine if the block cluster is either admissible or small. The admissibility condition is evaluated without mesh cell size correction.

Parameters

eta	Admissibility constant.
n_min	The size threshold value for determining if a cluster is large.

Returns

References BlockCluster< spacedim, Number >::check_is_admissible(), BlockCluster< spacedim, Number >↩
::check_is_near_field(), BlockCluster< spacedim, Number >::is_admissible, and BlockCluster< spacedim, Number >::is_near_field.

8.5.3.9 is_admissible_or_small() [2/2]

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::is_admissible_or_small (
            Number eta,
            const std::vector< Point< spacedim, Number >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs,
            unsigned int n_min )
```

Determine if the block cluster is either admissible or small. The admissibility condition is evaluated with mesh cell size correction.

Parameters

eta	Admissibility constant.
n_min	The size threshold value for determining if a cluster is large.

Returns

References BlockCluster< spacedim, Number >::check_is_admissible(), BlockCluster< spacedim, Number >↩ ::check_is_near_field(), BlockCluster< spacedim, Number >::is_admissible, and BlockCluster< spacedim, Number >::is_near_field.

8.5.3.10 is_proper_subset()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::is_proper_subset (
            const BlockCluster< spacedim, Number > & block_cluster ) const
```

Check if the index set of the current block cluster is a proper subset of that of the given block cluster.

Parameters

block_cluster	

Returns

References BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_↩ node.

8.5.3.11 is_proper_superset()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::is_proper_superset (
            const BlockCluster< spacedim, Number > & block_cluster ) const
```

Check if the index set of the current block cluster is a proper superset of that of the given block cluster.

Parameters

block_cluster	

Returns

References BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_↩ node.

8.5.3.12 is_small()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::is_small (
            unsigned int n_min )
```

Determine if the block cluster belongs to the near field set.

Parameters

n_min	

Returns

References BlockCluster< spacedim, Number >::check_is_near_field(), and BlockCluster< spacedim, Number >::is_near_field.

8.5.3.13 is_subset()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::is_subset (
            const BlockCluster< spacedim, Number > & block_cluster ) const
```

Check if the index set of the current block cluster is a subset of that of the given block cluster.

Parameters

block_cluster	

Returns

References BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_↩ node.

8.5.3.14 is_superset()

```
template<int spacedim, typename Number >
bool BlockCluster< spacedim, Number >::is_superset (
            const BlockCluster< spacedim, Number > & block_cluster ) const
```

Check if the index set of the current block cluster is a superset of that of the given block cluster.

Parameters

block_cluster	

Returns

References BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_node.

8.5.4 Friends And Related Function Documentation

8.5.4.1 is_equal

```
template<int spacedim, typename Number = double>
template<int spacedim1, typename Number1 >
bool is_equal (
            const BlockCluster< spacedim, Number > & block_cluster1,
            const BlockCluster< spacedim, Number > & block_cluster2 )   [friend]
```

Check the equality of two block cluster by comparing the contents of block cluster's index sets. Compared to `BlockCluster<spacedim`, Number>operator==, this can be considered as deep comparison.

Parameters

block_cluster1	
block_cluster2	

Returns

8.5.4.2 operator<<

```
template<int spacedim, typename Number = double>
template<int spacedim1, typename Number1 >
```

```
std::ostream& operator<< (
            std::ostream & out,
            const BlockCluster< spacedim1, Number1 > & block_cluster )  [friend]
```

Print out the block cluster data.

Parameters

out	
block_cluster	

Returns

8.5.5 Member Data Documentation

8.5.5.1 cluster_distance

```
template<int spacedim, typename Number = double>
Number BlockCluster< spacedim, Number >::cluster_distance  [private]
```

The distance between cluster τ and cluster σ. Its value is computed when evaluating the admissibility condition.

Referenced by BlockCluster< spacedim, Number >::check_is_admissible().

8.5.5.2 is_admissible

```
template<int spacedim, typename Number = double>
bool BlockCluster< spacedim, Number >::is_admissible  [private]
```

Is admissible.

Referenced by BlockCluster< spacedim, Number >::check_is_admissible(), BlockCluster< spacedim, Number >::get_is_admissible(), and BlockCluster< spacedim, Number >::is_admissible_or_small().

8.5.5.3 is_near_field

```
template<int spacedim, typename Number = double>
bool BlockCluster< spacedim, Number >::is_near_field  [private]
```

Whether the block cluster is of near-field, so that it needs a full matrix representation. Otherwise, it requires a rank-r matrix representation.

Referenced by BlockCluster< spacedim, Number >::check_is_near_field(), BlockCluster< spacedim, Number >↩ ::get_is_near_field(), BlockCluster< spacedim, Number >::is_admissible_or_small(), and BlockCluster< spacedim, Number >::is_small().

8.5.5.4 sigma_node

```
template<int spacedim, typename Number = double>
ClusterTree<spacedim, Number>::node_pointer_type BlockCluster< spacedim, Number >::sigma_node
[private]
```

Pointer to a node in the binary tree which holds the cluster σ.

Referenced by BlockCluster< spacedim, Number >::check_is_admissible(), BlockCluster< spacedim, Number >::check_is_near_field(), BlockCluster< spacedim, Number >::get_is_admissible(), BlockCluster< spacedim, Number >::intersect(), BlockCluster< spacedim, Number >::is_proper_subset(), BlockCluster< spacedim, Number >::is_proper_superset(), BlockCluster< spacedim, Number >::is_subset(), BlockCluster< spacedim, Number >::is_superset(), and operator==().

8.5.5.5 tau_node

```
template<int spacedim, typename Number = double>
ClusterTree<spacedim, Number>::node_pointer_type BlockCluster< spacedim, Number >::tau_node
[private]
```

Pointer to a node in the binary tree which holds the cluster τ.

Referenced by BlockCluster< spacedim, Number >::check_is_admissible(), BlockCluster< spacedim, Number >::check_is_near_field(), BlockCluster< spacedim, Number >::get_is_admissible(), BlockCluster< spacedim, Number >::intersect(), BlockCluster< spacedim, Number >::is_proper_subset(), BlockCluster< spacedim, Number >::is_proper_superset(), BlockCluster< spacedim, Number >::is_subset(), BlockCluster< spacedim, Number >::is_superset(), and operator==().

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

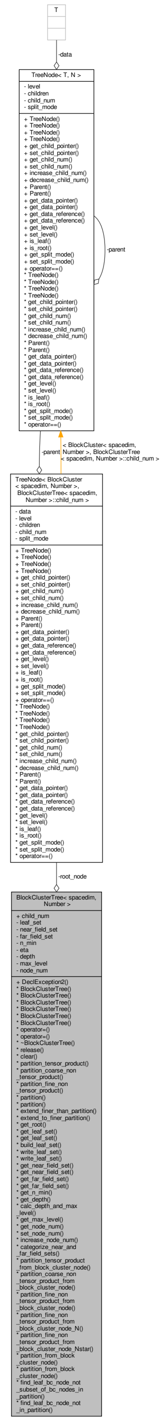
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/block_cluster.h

8.6 BlockClusterTree< spacedim, Number > Class Template Reference

Class for block cluster tree.

```
#include <block_cluster_tree.h>
```

Collaboration diagram for BlockClusterTree< spacedim, Number >:



Public Types

- typedef TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > node_value_type
- typedef TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > ∗ **node_pointer_type**
- typedef const TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > ∗ **node_const_pointer_type**

- typedef TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > & **node_reference_type**
- typedef const TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > & **node_const_reference_type**
- typedef BlockCluster< spacedim, Number > data_value_type
- typedef BlockCluster< spacedim, Number > * **data_pointer_type**
- typedef const BlockCluster< spacedim, Number > * **data_const_pointer_type**
- typedef BlockCluster< spacedim, Number > & **data_reference_type**
- typedef const BlockCluster< spacedim, Number > & **data_const_reference_type**

Public Member Functions

- **DeclException2** (ExcClusterLevelMismatch, unsigned int, unsigned int,<< "The level of cluster tau "<< arg1<< " is different from that of cluster sigma"<< arg2<< " which is not allowed in a level preserving construction of a block cluster tree.")

Static Public Attributes

- static const unsigned int child_num = 4

Private Attributes

- node_pointer_type **root_node**
- std::vector< node_pointer_type > **leaf_set**
- std::vector< node_pointer_type > **near_field_set**
- std::vector< node_pointer_type > **far_field_set**
- unsigned int **n_min**
- Number **eta**
- unsigned int depth
- int max_level
- unsigned int node_num

Friends

- template<int spacedim1, typename Number1 >
  std::ostream & operator<< (std::ostream &out, const BlockClusterTree< spacedim1, Number1 > &block_cluster_tree)
- template<int spacedim1, typename Number1 >
  void **split_block_cluster_node** (TreeNode< BlockCluster< spacedim1, Number1 >, BlockClusterTree< spacedim1, Number1 >::child_num > *bc_node, BlockClusterTree< spacedim1, Number1 > &bc_tree, const TreeNodeSplitMode split_mode, const bool if_add_child_nodes_to_leaf_set)
- template<int spacedim1, typename Number1 >
  TreeNode< BlockCluster< spacedim1, Number1 >, BlockClusterTree< spacedim1, Number1 >::child_num > * **find_bc_node_in_partition_intersect_current_bc_node** (TreeNode< BlockCluster< spacedim1, Number1 >, BlockClusterTree< spacedim1, Number1 >::child_num > *current_bc_node, const std::vector< TreeNode< BlockCluster< spacedim1, Number1 >, BlockClusterTree< spacedim1, Number1 >::child_num > *> &partition)
- template<int spacedim1, typename Number1 >
  TreeNode< BlockCluster< spacedim1, Number1 >, BlockClusterTree< spacedim1, Number1 >::child_num > * **find_bc_node_in_partition_proper_subset_of_current_bc_node** (TreeNode< BlockCluster< spacedim1, Number1 >, BlockClusterTree< spacedim1, Number1 >::child_num > *current_bc_node, const std::vector< TreeNode< BlockCluster< spacedim1, Number1 >, BlockClusterTree< spacedim1, Number1 >::child_num > *> &partition)

- BlockClusterTree ()
- BlockClusterTree (const ClusterTree< spacedim, Number > &TI, const ClusterTree< spacedim, Number > &TJ, const unsigned int n_min=0)
- BlockClusterTree (typename ClusterTree< spacedim, Number >::node_pointer_type tau_root_node, typename ClusterTree< spacedim, Number >::node_pointer_type sigma_root_node, const unsigned int n_min=1)
- BlockClusterTree (const ClusterTree< spacedim, Number > &TI, const ClusterTree< spacedim, Number > &TJ, const Number eta, const unsigned int n_min)
- BlockClusterTree (const BlockClusterTree< spacedim, Number > &bct)
- BlockClusterTree (BlockClusterTree< spacedim, Number > &&bct)
- BlockClusterTree< spacedim, Number > & operator= (const BlockClusterTree< spacedim, Number > &bct)
- BlockClusterTree< spacedim, Number > & operator= (BlockClusterTree< spacedim, Number > &&bct)
- ~BlockClusterTree ()
- void release ()
- void clear ()
- void partition_tensor_product ()
- void partition_coarse_non_tensor_product ()
- void partition_fine_non_tensor_product ()
- void partition (const std::vector< Point< spacedim >> &all_support_points)
- void partition (const std::vector< Point< spacedim >> &all_support_points, const std::vector< Number > &cell_size_at_dofs)
- bool extend_finer_than_partition (const std::vector< node_pointer_type > &partition)
- bool extend_to_finer_partition (const std::vector< node_pointer_type > &partition)
- node_pointer_type get_root () const
- std::vector< node_pointer_type > & get_leaf_set ()
- const std::vector< node_pointer_type > & get_leaf_set () const
- void build_leaf_set ()
- void write_leaf_set (std::ostream &out) const
- template<typename Number1 = double>
  void write_leaf_set (std::ostream &out, const LAPACKFullMatrixExt< Number1 > &matrix, const Number1 singular_value_threshold=0.) const
- std::vector< node_pointer_type > & get_near_field_set ()
- const std::vector< node_pointer_type > & get_near_field_set () const
- std::vector< node_pointer_type > & get_far_field_set ()
- const std::vector< node_pointer_type > & get_far_field_set () const
- unsigned int get_n_min () const
- unsigned int get_depth () const
- void calc_depth_and_max_level ()
- int get_max_level () const
- unsigned int get_node_num () const
- void set_node_num (unsigned int node_num)
- void increase_node_num (unsigned int increased_node_num=1)
- void categorize_near_and_far_field_sets ()
- void partition_tensor_product_from_block_cluster_node (node_pointer_type current_block_cluster_node, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- void partition_coarse_non_tensor_product_from_block_cluster_node (node_pointer_type current_block_cluster_node, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- void partition_fine_non_tensor_product_from_block_cluster_node (node_pointer_type current_block_cluster_node, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- void partition_fine_non_tensor_product_from_block_cluster_node_N (node_pointer_type current_block_cluster_node, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- void partition_fine_non_tensor_product_from_block_cluster_node_Nstar (node_pointer_type current_block_cluster_node, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- void partition_from_block_cluster_node (node_pointer_type current_block_cluster_node, const std::vector< Point< spacedim >> &all_support_points, std::vector< node_pointer_type > &leaf_set_wrt_current_node)

- void partition_from_block_cluster_node (node_pointer_type current_block_cluster_node, const std::vector< Point< spacedim >> &all_support_points, const std::vector< Number > &cell_size_at_dofs, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- node_pointer_type find_leaf_bc_node_not_subset_of_bc_nodes_in_partition (const std::vector< node_pointer_type > &partition, typename std::vector< node_pointer_type >::const_iterator &it_for_desired_bc_node) const
- node_pointer_type find_leaf_bc_node_not_in_partition (const std::vector< node_pointer_type > &partition, typename std::vector< node_pointer_type >::const_iterator &it_for_desired_bc_node) const

8.6.1 Detailed Description

template<int spacedim, typename Number = double>
class BlockClusterTree< spacedim, Number >

Class for block cluster tree.

A block cluster tree is a quad-tree which holds a hierarchy of doubly linked nodes with the type TreeNode. Because a node in the block cluster tree has four children, the template argument `T` required by `TreeNode` should be 4.

8.6.2 Member Typedef Documentation

8.6.2.1 data_value_type

```
template<int spacedim, typename Number = double>
typedef BlockCluster<spacedim, Number> BlockClusterTree< spacedim, Number >::data_value_type
```

Data type of the data held by a tree node.

8.6.2.2 node_value_type

```
template<int spacedim, typename Number = double>
typedef TreeNode<BlockCluster<spacedim, Number>, BlockClusterTree<spacedim, Number>::child_num> BlockClusterTree< spacedim, Number >::node_value_type
```

Data type of the tree node.

8.6.3 Constructor & Destructor Documentation

8.6.3.1 BlockClusterTree() [1/6]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::BlockClusterTree ( )
```

Default constructor, which initializes an empty quad-tree.

8.6.3.2 BlockClusterTree() [2/6]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::BlockClusterTree (
            const ClusterTree< spacedim, Number > & TI,
            const ClusterTree< spacedim, Number > & TJ,
            const unsigned int n_min = 0 )
```

Construct from two cluster trees built from pure cardinality based partition, which has no admissibility condition.

Parameters

TI	
TJ	
n_min	

References CreateTreeNode(), ClusterTree< spacedim, Number >::get_n_min(), and ClusterTree< spacedim, Number >::get_root().

8.6.3.3 BlockClusterTree() [3/6]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::BlockClusterTree (
            typename ClusterTree< spacedim, Number >::node_pointer_type tau_root_node,
            typename ClusterTree< spacedim, Number >::node_pointer_type sigma_root_node,
            const unsigned int n_min = 1 )
```

Construct from two cluster nodes, whose Cartesian product is the root node of the block cluster tree.

Parameters

tau_root_node	
sigma_root_node	

References CreateTreeNode().

8.6.3.4 BlockClusterTree() [4/6]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::BlockClusterTree (
            const ClusterTree< spacedim, Number > & TI,
            const ClusterTree< spacedim, Number > & TJ,
            const Number eta,
            const unsigned int n_min )
```

Construct from two cluster trees and admissibility condition.

The Cartesian product of the two clusters in the root nodes of $T(I)$ and $T(J)$ becomes the data in the root node of the block cluster tree.

References CreateTreeNode(), ClusterTree< spacedim, Number >::get_n_min(), and ClusterTree< spacedim, Number >::get_root().

8.6.3.5 BlockClusterTree() [5/6]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::BlockClusterTree (
            const BlockClusterTree< spacedim, Number > & bct )
```

Copy constructor for BlockClusterTree, which realizes deep copy internally.

Parameters

bct	

References BlockClusterTree< spacedim, Number >::build_leaf_set(), BlockClusterTree< spacedim, Number >↩
::categorize_near_and_far_field_sets(), CopyTree(), and BlockClusterTree< spacedim, Number >::get_root().

8.6.3.6 BlockClusterTree() [6/6]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::BlockClusterTree (
            BlockClusterTree< spacedim, Number > && bct )
```

Copy constructor via shallow copy.

Parameters

bct	

Returns

References BlockClusterTree< spacedim, Number >::operator=().

8.6.3.7 ∼BlockClusterTree()

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::∼BlockClusterTree ( )
```

Destructor which recursively destroys every node in the block cluster tree.

References BlockClusterTree< spacedim, Number >::release().

8.6.4 Member Function Documentation

8.6.4.1 build_leaf_set()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::build_leaf_set ( )
```

Build the leaf set by tree recursion.

References GetTreeLeaves().

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree(), and BlockClusterTree< spacedim, Number >::operator=().

8.6.4.2 calc_depth_and_max_level()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::calc_depth_and_max_level ( )
```

Calculate the depth.

References calc_depth(), BlockClusterTree< spacedim, Number >::depth, and BlockClusterTree< spacedim, Number >::max_level.

Referenced by BlockClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and BlockClusterTree< spacedim, Number >::partition_tensor_product().

8.6.4.3 categorize_near_and_far_field_sets()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::categorize_near_and_far_field_sets ( )
```

Categorize the leaf set into near field set and far field set.

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree(), BlockClusterTree< spacedim, Number >::operator=(), BlockClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and BlockClusterTree< spacedim, Number >::partition_tensor_product().

8.6.4.4 clear()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::clear ( )
```

Clear the data field of the block cluster tree because its memory has been migrated to another object via shallow copy.

References BlockClusterTree< spacedim, Number >::depth, BlockClusterTree< spacedim, Number >::max_level, and BlockClusterTree< spacedim, Number >::node_num.

8.6.4.5 extend_finer_than_partition()

```
template<int spacedim, typename Number >
bool BlockClusterTree< spacedim, Number >::extend_finer_than_partition (
            const std::vector< node_pointer_type > & partition )
```

Extend the current block cluster tree to be finer than the given partition.

This member functions implements (7.10a) in Hackbusch's $\mathcal{H}$-matrix book.

Note **This algorithm iterates over each element in the leaf set of the block cluster tree to be extended. During the iteration, because leaf nodes may be further split into smaller ones, the leaf set is not a constant. Hence, the leaf set should be updated immediately whenever a block cluster node is split. This behavior is different from other functions such as `HMatrix<spacedim, Number>::refine_to_supertree`, the leaf set of of which will be built after the whole tree hierarchy has been constructed.**

Parameters

P	

Returns

Whether the block cluster tree has really been extended.

Iterate over the leaf set of the current block cluster tree and look for a block cluster which is not contained by any block cluster in the given partition.

Enter the loop by selecting a block cluster node from the leaf set, which is not contained in any block cluster nodes in the given partition.

Because the selected block cluster node in the leaf set is about to be split, we delete the current block cluster node from the leaf set first and add its children to the leaf set later on.

Select a block cluster node in the given partition, whose index set has a nonempty intersection with the index set of the current block cluster node.

Because the given partition is a covering of the complete block cluster index set $I \times J$, such block cluster node must exist. Hence we make an assertion here.

Extend the current block cluster node by splitting its τ node, which is horizontal split, then make Cartesian product with its σ node.

Extend the current block cluster node by splitting its σ node, which is vertical split, then make Cartesian product with its τ node.

Extend the current block cluster node by splitting its τ node, which is horizontal split, then make Cartesian product with its σ node.

Extend the current block cluster node by splitting its σ node, which is vertical split, then make Cartesian product with its τ node.

Update the maximum level and depth of the current tree if necessary.

After block cluster tree extension, re-categorize the near field set and far field set based on the new leaf set. N.B. The value of `n_min` does not change.

References BlockClusterTree< spacedim, Number >::find_leaf_bc_node_not_subset_of_bc_nodes_in_partition(), and TreeNode< T, N >::get_data_reference().

Referenced by HMatrix< spacedim, Number >::convert_between_different_block_cluster_trees(), and main().

8.6.4.6 extend_to_finer_partition()

```
template<int spacedim, typename Number >
bool BlockClusterTree< spacedim, Number >::extend_to_finer_partition (
            const std::vector< node_pointer_type > & partition )
```

Extend the current block cluster tree to the given finer partition.

This member functions implements (7.10b) in Hackbusch's $\mathcal{H}$-matrix book.

Note This algorithm iterates over each element in the leaf set of the block cluster tree to be extended. During the iteration, because leaf nodes may be further split into smaller ones, the leaf set is not a constant. Hence, the leaf set should be updated immediately whenever a block cluster node is split. This behavior is different from other functions such as `HMatrix<spacedim, Number>::refine_to_supertree`, the leaf set of of which will be built after the whole tree hierarchy has been constructed.

Iterate over the leaf set of the current block cluster tree and look for a block cluster which does not belong to the partition.

Enter the loop by selecting a block cluster node from the leaf set, which does not belong to the given partition.

Because the selected block cluster node in the leaf set is about to be split, we delete the current block cluster node from the leaf set first and add its children to the leaf set later on.

Select a block cluster node in the given partition, whose index set is a proper subset of the index set of the current block cluster node.

Because the partition is a covering of the complete block cluster index set $I \times J$ and it is finer than the leaf set of the current block cluster tree, such block cluster node must exist. Hence we make an assertion here.

Here we ensure that the level difference between τ and σ clusters is at most 1.

Extend the current block cluster node by splitting its τ node, which is horizontal split, then make Cartesian product with its σ node.

Extend the current block cluster node by splitting its σ node, which is vertical split, then make Cartesian product with its τ node.

Extend the current block cluster node by splitting its τ node, which is horizontal split, then make Cartesian product with its σ node.

Extend the current block cluster node by splitting its σ node, which is vertical split, then make Cartesian product with its τ node.

Update the maximum level and depth of the current tree if necessary.

After block cluster tree extension, re-categorize the near field set and far field set based on the new leaf set. N.B. The value of `n_min` does not change.

References BlockClusterTree< spacedim, Number >::find_leaf_bc_node_not_in_partition(), and TreeNode< T, N >::get_data_reference().

Referenced by HMatrix< spacedim, Number >::convert_between_different_block_cluster_trees(), and main().

8.6.4.7 find_leaf_bc_node_not_in_partition()

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::node_pointer_type BlockClusterTree< spacedim, Number
>::find_leaf_bc_node_not_in_partition (
            const std::vector< node_pointer_type > & partition,
            typename std::vector< node_pointer_type >::const_iterator & it_for_desired_bc_↩
node ) const [private]
```

Find a block cluster node in the leaf set of the current block cluster tree, which does not belong to the partition.

Parameters

partition	the given partition.
it_for_desired_bc_node	the returned iterator which points to the selected block cluster node in the leaf set of the current block cluster tree. N.B. Only when the returned pointer is not `nullptr`, this iterator is meaningful.

Returns

pointer to the selected block cluster node.

Work flow Iterate over each block cluster node in the leaf set of the current block cluster tree.

Iterate over each block cluster node in the given partition.

When the index set of the current block cluster node is equal to the index set of some block cluster node in the given partition, terminate the inner loop and jump to the next leaf node for checking.

Here the desired block cluster node in the leaf set is found. Then exist the outer loop and the iterator `it_↩ for_desired_bc_node` now pointing to this node will also be returned.

References TreeNode< T, N >::get_data_reference(), and is_equal().

Referenced by BlockClusterTree< spacedim, Number >::extend_to_finer_partition().

8.6.4.8 find_leaf_bc_node_not_subset_of_bc_nodes_in_partition()

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::node_pointer_type BlockClusterTree< spacedim, Number
>::find_leaf_bc_node_not_subset_of_bc_nodes_in_partition (
            const std::vector< node_pointer_type > & partition,
            typename std::vector< node_pointer_type >::const_iterator & it_for_desired_bc_node ) const  [private]
```

Find a block cluster node in the leaf set of the current block cluster tree, which is not a subset of any block cluster in the given partition.

Parameters

partition	the given partition.
it_for_desired_bc_node	the returned iterator which points to the selected block cluster node in the leaf set of the current block cluster tree. N.B. Only when the returned pointer is not `nullptr`, this iterator is meaningful.

Returns

pointer to the selected block cluster node.

Work flow Iterate over each block cluster node in the leaf set of the current block cluster tree.

Iterate over each block cluster node in the given partition.

When the index set of the current block cluster node in the leaf set is a subset of the index set of some block cluster node in the given partition, terminate the inner loop and jump to the next leaf node for checking.

Here the desired block cluster node in the leaf set is found. Then exist the outer loop and the iterator `it_for_desired_bc_node` now pointing to this node will also be returned.

References TreeNode< T, N >::get_data_reference().

Referenced by BlockClusterTree< spacedim, Number >::extend_finer_than_partition().

8.6.4.9 get_depth()

```
template<int spacedim, typename Number >
unsigned int BlockClusterTree< spacedim, Number >::get_depth ( ) const
```

Get the tree depth.

References BlockClusterTree< spacedim, Number >::depth.

8.6.4.10 get_far_field_set() [1/2]

```
template<int spacedim, typename Number >
std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer_type > & Block←
ClusterTree< spacedim, Number >::get_far_field_set ( )
```

Get the reference to the block cluster list which belongs to the far field.

8.6.4.11 get_far_field_set() [2/2]

```
template<int spacedim, typename Number >
const std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer_type > & Block←
ClusterTree< spacedim, Number >::get_far_field_set ( ) const
```

Get the reference to the block cluster list which belongs to the far field (const version).

8.6.4.12 get_leaf_set() [1/2]

```
template<int spacedim, typename Number >
std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer_type > & Block←
ClusterTree< spacedim, Number >::get_leaf_set ( )
```

Get the reference to the block cluster list.

Referenced by HMatrix< spacedim, Number >::coarsen_to_subtree(), HMatrix< spacedim, Number >::convert←
_between_different_block_cluster_trees(), and main().

8.6.4.13 get_leaf_set() [2/2]

```
template<int spacedim, typename Number >
const std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer_type > & Block←
ClusterTree< spacedim, Number >::get_leaf_set ( ) const
```

Get the reference to the block cluster list (const version).

8.6.4.14 get_max_level()

```
template<int spacedim, typename Number >
int BlockClusterTree< spacedim, Number >::get_max_level ( ) const
```

Get the maximum level of the tree.

References BlockClusterTree< spacedim, Number >::max_level.

8.6.4.15 get_n_min()

```
template<int spacedim, typename Number >
unsigned int BlockClusterTree< spacedim, Number >::get_n_min ( ) const
```

Get the minimum cluster size.

Referenced by HMatrix< spacedim, Number >::mmult().

8.6.4.16 get_near_field_set() [1/2]

```
template<int spacedim, typename Number >
std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer_type > & Block↩
ClusterTree< spacedim, Number >::get_near_field_set ( )
```

Get the reference to the block cluster list which belongs to the near field.

8.6.4.17 get_near_field_set() [2/2]

```
template<int spacedim, typename Number >
const std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer_type > & Block↩
ClusterTree< spacedim, Number >::get_near_field_set ( ) const
```

Get the reference to the block cluster list which belongs to the near field (const version).

8.6.4.18 get_node_num()

```
template<int spacedim, typename Number >
unsigned int BlockClusterTree< spacedim, Number >::get_node_num ( ) const
```

Get the total number of clusters in the tree.

References BlockClusterTree< spacedim, Number >::node_num.

8.6.4.19 get_root()

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number >::node_pointer_type BlockClusterTree< spacedim, Number
>::get_root ( ) const
```

Get the pointer to the root node of the block cluster tree.

Returns

References BlockClusterTree< spacedim, Number >::partition_tensor_product_from_block_cluster_node().

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree(), HMatrix< spacedim, Number >::H↩
Matrix(), HMatrix< spacedim, Number >::mmult(), and BlockClusterTree< spacedim, Number >::operator=().

8.6.4.20 increase_node_num()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::increase_node_num (
            unsigned int increased_node_num = 1 )
```

Increase the total number of nodes in the tree.

References BlockClusterTree< spacedim, Number >::node_num.

8.6.4.21 operator=() [1/2]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number > & BlockClusterTree< spacedim, Number >::operator= (
            const BlockClusterTree< spacedim, Number > & bct )
```

Overloaded assignment operator with deep copy.

Parameters

bct	

Returns

References BlockClusterTree< spacedim, Number >::build_leaf_set(), BlockClusterTree< spacedim, Number >::categorize_near_and_far_field_sets(), CopyTree(), BlockClusterTree< spacedim, Number >::depth, BlockClusterTree< spacedim, Number >::get_root(), BlockClusterTree< spacedim, Number >::max_level, BlockClusterTree< spacedim, Number >::node_num, and BlockClusterTree< spacedim, Number >::release().

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree().

8.6.4.22 operator=() [2/2]

```
template<int spacedim, typename Number >
BlockClusterTree< spacedim, Number > & BlockClusterTree< spacedim, Number >::operator= (
            BlockClusterTree< spacedim, Number > && bct )
```

Overloaded assignment operator with shallow copy.

Parameters

bct	

Returns

References BlockClusterTree< spacedim, Number >::depth, BlockClusterTree< spacedim, Number >::max_level, BlockClusterTree< spacedim, Number >::node_num, and BlockClusterTree< spacedim, Number >::release().

8.6.4.23 partition() [1/2]

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition (
            const std::vector< Point< spacedim >> & all_support_points )
```

Perform a recursive partition by starting from the root node. The evaluation of the admissibility condition has no mesh cell size correction.

Parameters

all_support_points	All the support points.

References BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), BlockClusterTree< spacedim, Number >::categorize_near_and_far_field_sets(), and BlockClusterTree< spacedim, Number >::partition_from_block_cluster_node().

Referenced by find_bc_node_in_partition_intersect_current_bc_node(), and main().

8.6.4.24 partition() [2/2]

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition (
            const std::vector< Point< spacedim >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs )
```

Perform a recursive partition by starting from the root node. The evaluation of the admissibility condition has mesh cell size correction.

Parameters

all_support_points	All the support points.

References BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), BlockClusterTree< spacedim, Number >::categorize_near_and_far_field_sets(), and BlockClusterTree< spacedim, Number >::partition_from_block_cluster_node().

8.6.4.25 partition_coarse_non_tensor_product()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product ( )
```

Perform a recursive partition in coarse non-tensor product form without the admissibility condition because the two comprising cluster trees are built from pure cardinality based partition.

References BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), BlockClusterTree< spacedim, Number >::categorize_near_and_far_field_sets(), and BlockClusterTree< spacedim, Number >::partition_↩ coarse_non_tensor_product_from_block_cluster_node().

Referenced by main().

8.6.4.26 partition_coarse_non_tensor_product_from_block_cluster_node()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product_from_block_↩
cluster_node (
            node_pointer_type current_block_cluster_node,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a recursive non-tensor product type coarse partition by starting from a block cluster node in the tree.

No admissibility condition is enabled in this situation, because the two comprising cluster trees are built from pure cardinality based partition.

Reference: Section 2.2.2 in Hackbusch, W. 1999. "A Sparse Matrix Arithmetic Based on H-Matrices. Part I↩ : Introduction to H-Matrices." Computing 62 (2): 89–108.

Parameters

current_block_cluster_node	
leaf_set_wrt_current_node	

Push back the current cluster node, which is small, to the leaf set and set its split mode as `UnsplitMode`.

Make sure that the two clusters τ and σ have the same level in their respective cluster trees, i.e. level preserving property should be satisfied.

Create a new block cluster node as child. Then append this new node as one of the children of the current block cluster node. Finally, recursively partition from this node if the two component clusters have the same indices, i.e. $I_1 \times I_1$ and $I_2 \times I_2$; otherwise, for $I_1 \times I_2$ and $I_2 \times I_1$, stop the recursion and directly add them to the leaf set.

Append this new node as one of the children of the current block cluster node.

Handle the case for $I_1 \times I_1$ and $I_2 \times I_2$.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Handle the case for $I_1 \times I_2$ and $I_2 \times I_1$. Because the recursion stops here, we need to check and update its near field property.

Make sure the current block cluster have four children, which is ensured by the non-tensor product construction.

Note The second argument `BlockClusterTree<spacedim`, Number>::child_num should be wrapped between the brackets, otherwise, the program cannot be compiled.

Set the split mode of the current block cluster node as cross.

References TreeNode< T, N >::get_data_pointer(), and TreeNode< T, N >::set_split_mode().

Referenced by BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product().

8.6.4.27 partition_fine_non_tensor_product()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product ( )
```

Perform a recursive partition in fine non-tensor product form without the admissibility condition because the two comprising cluster trees are built from pure cardinality based partition.

References BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), BlockClusterTree< spacedim, Number >::categorize_near_and_far_field_sets(), and BlockClusterTree< spacedim, Number >::partition_fine_↩
non_tensor_product_from_block_cluster_node().

Referenced by main().

8.6.4.28 partition_fine_non_tensor_product_from_block_cluster_node()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product_from_block_↩
cluster_node (
            node_pointer_type current_block_cluster_node,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node ) [private]
```

Perform a recursive non-tensor product type fine partition of $\mathcal{M}_{\mathcal{H},k}$ type by starting from a block cluster node in the tree.

No admissibility condition is enabled in this situation, because the two comprising cluster trees are built from pure cardinality based partition.

Reference: Section 5 in Hackbusch, W. 1999. “A Sparse Matrix Arithmetic Based on H-Matrices. Part I: Introduction to H-Matrices.” Computing 62 (2): 89–108.

Parameters

current_block_cluster_node	
leaf_set_wrt_current_node	

Push back the current cluster node, which is small, to the leaf set and set its split mode as `UnsplitMode`.

Set the split mode of the current block cluster node as cross.

Make sure that the two clusters τ and σ have the same level in their respective cluster trees, i.e. level preserving property should be satisfied.

Create a new block cluster node as child. Then append this new node as one of the children of the current block cluster node. Finally, recursively partition from this node using the fine non-tensor product method if the two component clusters have the same indices, i.e. $I_1 \times I_1$ and $I_2 \times I_2$; for $I_1 \times I_2$, recursively partition from it using the $\mathcal{N}$-type partition method; for $I_2 \times I_1$, recursively partition from it using the $\mathcal{N}^*$-type partition method.

Append this new node as one of the children of the current block cluster node.

Handle the case for $I_1 \times I_1$ and $I_2 \times I_2$ by recursively applying the fine non-tensor product partition method itself.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Handle the case for $I_1 \times I_2$ and perform the $\mathcal{N}$-type recursive partition.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Handle the case for $I_2 \times I_1$ and perform the $\mathcal{N}^*$-type recursive partition.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

This case can never happen.

Make sure the current block cluster have four children, which is ensured by the fine non-tensor product construction.

Note **The second argument `BlockClusterTree`<`spacedim`, Number>::child_num should be wrapped between the brackets, otherwise, the program cannot be compiled.**

References TreeNode< T, N >::get_data_pointer(), and TreeNode< T, N >::set_split_mode().

Referenced by BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product().

8.6.4.29 partition_fine_non_tensor_product_from_block_cluster_node_N()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product_from_block_↩
cluster_node_N (
            node_pointer_type current_block_cluster_node,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a recursive non-tensor product type fine partition of $\mathcal{M}_{\mathcal{N},k}$ type by starting from a block cluster node in the tree.

No admissibility condition is enabled in this situation, because the two comprising cluster trees are built from pure cardinality based partition.

Reference: Section 5 in Hackbusch, W. 1999. “A Sparse Matrix Arithmetic Based on H-Matrices. Part I: Introduction to H-Matrices.” Computing 62 (2): 89–108.

Parameters

current_block_cluster_node	
leaf_set_wrt_current_node	

Push back the current cluster node, which is small, to the leaf set and set its split mode as `UnsplitMode`.

Make sure that the two clusters τ and σ have the same level in their respective cluster trees, i.e. level preserving property should be satisfied.

Create a new block cluster node as child. Then append this new node as one of the children of the current block cluster node. Finally, recursively partition from this node if it is on the bottom left corner, i.e. it is the $I_2 \times I_1$ block cluster; otherwise, for $I_1 \times I_1$, $I_1 \times I_2$ and $I_2 \times I_2$, stop the recursion and directly add them to the leaf set.

Append this new node as one of the children of the current block cluster node.

Handle the case for $I_2 \times I_1$ and perform the $\mathcal{N}$-type recursive partition.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Handle the case for $I_1 \times I_1$, $I_1 \times I_2$ and $I_2 \times I_2$. Because the recursion stops here, we need to check and update its near field property.

Make sure the current block cluster have four children, which is ensured by the fine non-tensor product $\mathcal{N}$-type construction.

Note The second argument `BlockClusterTree<spacedim`, Number>::child_num should be wrapped between the brackets, otherwise, the program cannot be compiled.

Set the split mode of the current block cluster node as cross.

References TreeNode< T, N >::get_data_pointer(), and TreeNode< T, N >::set_split_mode().

8.6.4.30 partition_fine_non_tensor_product_from_block_cluster_node_Nstar()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product_from_block_↩
cluster_node_Nstar (
            node_pointer_type current_block_cluster_node,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a recursive non-tensor product type fine partition of $\mathcal{M}_{\mathcal{N}^*,k}$ type by starting from a block cluster node in the tree.

No admissibility condition is enabled in this situation, because the two comprising cluster trees are built from pure cardinality based partition.

Reference: Section 5 in Hackbusch, W. 1999. “A Sparse Matrix Arithmetic Based on H-Matrices. Part I: Introduction to H-Matrices.” Computing 62 (2): 89–108.

Parameters

current_block_cluster_node	
leaf_set_wrt_current_node	

Push back the current cluster node, which is small, to the leaf set and set its split mode as `UnsplitMode`.

Make sure that the two clusters τ and σ have the same level in their respective cluster trees, i.e. level preserving property should be satisfied.

Create a new block cluster node as child. Then append this new node as one of the children of the current block cluster node. Finally, recursively partition from this node if it is on the top right corner, i.e. it is the $I_1 \times I_2$ block cluster; otherwise, for $I_1 \times I_1$, $I_2 \times I_1$ and $I_2 \times I_2$, stop the recursion and directly add them to the leaf set.

Append this new node as one of the children of the current block cluster node.

Handle the case for $I_1 \times I_2$ and perform the $\mathcal{N}^*$-type recursive partition.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Handle the case for $I_1 \times I_1$, $I_2 \times I_1$ and $I_2 \times I_2$. Because the recursion stops here, we need to check and update its near field property.

Make sure the current block cluster have four children, which is ensured by the fine non-tensor product $\mathcal{N}^*$-type construction.

Note The second argument `BlockClusterTree<spacedim`, Number>::child_num should be wrapped between the brackets, otherwise, the program cannot be compiled.

Set the split mode of the current block cluster node as cross.

References TreeNode< T, N >::get_data_pointer(), and TreeNode< T, N >::set_split_mode().

8.6.4.31 partition_from_block_cluster_node() [1/2]

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_from_block_cluster_node (
            node_pointer_type current_block_cluster_node,
            const std::vector< Point< spacedim >> & all_support_points,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a recursive partition by starting from a block cluster node in the tree.

N.B. The evaluation of the admissibility condition has no mesh cell size correction.

The algorithm performs an iteration over all the children of the current block cluster $b = \tau \times \sigma$. Because the map S for generating the children of b is realized from a tensor product of the children of τ and σ, the algorithm contains nested double loops.

Parameters

current_block_cluster_node	The pointer to the block cluster node in the tree, from which the admissible partition will be performed.
all_support_points	Spatial coordinates for all the support points.
leaf_set	A list of block cluster node pointers which comprise the leaf set with respect to `current_block_cluster_node`.

Push back the current cluster node, which is small, to the leaf set and set its split mode as `UnsplitMode`.

Make sure that the two clusters τ and σ have the same level in their respective cluster trees, i.e. level preserving property should be satisfied.

Create a new block cluster node as child and recursively partition from it.

Append this new node as one of the children of the current block cluster node.

Update the total number of nodes in the tree.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Make sure the current block cluster have four children, which is ensured by the tensor product construction.

Note The second argument `BlockClusterTree<spacedim`, Number>::child_num should be wrapped between the brackets, otherwise, the program cannot be compiled.

Set the split mode of the current block cluster node as cross.

References TreeNode< T, N >::get_data_pointer(), and TreeNode< T, N >::set_split_mode().

Referenced by BlockClusterTree< spacedim, Number >::partition().

8.6.4.32 partition_from_block_cluster_node() [2/2]

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_from_block_cluster_node (
            node_pointer_type current_block_cluster_node,
            const std::vector< Point< spacedim >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Same as above but the evaluation of the admissibility condition has mesh cell size correction. Push back the current cluster node, which is small, to the leaf set and set its split mode as `UnsplitMode`.

Make sure that the two clusters τ and σ have the same level in their respective cluster trees, i.e. level preserving property should be satisfied.

Create a new block cluster node as child and recursively partition from it.

Append this new node as one of the children of the current block cluster node.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Make sure the current block cluster have four children, which is ensured by the tensor product construction.

Note The second argument `BlockClusterTree<spacedim`, Number>::child_num should be wrapped between the brackets, otherwise, the program cannot be compiled.

Set the split mode of the current block cluster node as cross.

References TreeNode< T, N >::get_data_pointer(), and TreeNode< T, N >::set_split_mode().

8.6.4.33 partition_tensor_product()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_tensor_product ( )
```

Perform a recursive partition in tensor product form without the admissibility condition because the two comprising cluster trees are built from pure cardinality based partition.

References BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), BlockClusterTree< spacedim, Number >::categorize_near_and_far_field_sets(), and BlockClusterTree< spacedim, Number >::partition_↩ tensor_product_from_block_cluster_node().

Referenced by main().

8.6.4.34 partition_tensor_product_from_block_cluster_node()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::partition_tensor_product_from_block_cluster_node (
            node_pointer_type current_block_cluster_node,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a recursive tensor product type partition by starting from a block cluster node in the tree.

No admissibility condition is enabled in this situation, because the two comprising cluster trees are built from pure cardinality based partition.

Parameters

current_block_cluster_node	
leaf_set_wrt_current_node	

Push back the current cluster node, which is small, to the leaf set and set its split mode as `UnsplitMode`.

Make sure that the two clusters τ and σ have the same level in their respective cluster trees, i.e. level preserving property should be satisfied.

Create a new block cluster node as child and recursively partition from it.

Append this new node as one of the children of the current block cluster node.

Merge the leaf set wrt. the child block cluster node into the leaf set of the current block cluster node.

Make sure the current block cluster have four children, which is ensured by the tensor product construction.

Note The second argument `BlockClusterTree<spacedim`, Number>::child_num should be wrapped between the brackets, otherwise, the program cannot be compiled.

Set the split mode of the current block cluster node as cross.

References TreeNode< T, N >::get_data_pointer(), and TreeNode< T, N >::set_split_mode().

Referenced by BlockClusterTree< spacedim, Number >::get_root(), and BlockClusterTree< spacedim, Number >::partition_tensor_product().

8.6.4.35 release()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::release ( )
```

Release the memory of the block cluster tree.

References DeleteTree(), BlockClusterTree< spacedim, Number >::depth, BlockClusterTree< spacedim, Number >::max_level, and BlockClusterTree< spacedim, Number >::node_num.

Referenced by BlockClusterTree< spacedim, Number >::operator=(), and BlockClusterTree< spacedim, Number >::~BlockClusterTree().

8.6.4.36 set_node_num()

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::set_node_num (
            unsigned int node_num )
```

Set the total number of nodes in the tree.

References BlockClusterTree< spacedim, Number >::node_num.

8.6.4.37 write_leaf_set() [1/2]

```
template<int spacedim, typename Number >
void BlockClusterTree< spacedim, Number >::write_leaf_set (
            std::ostream & out ) const
```

Write formatted leaf set to the output stream.

Each leaf node is written in the following format:

[list-of-indices-in-cluster-tau],[list-of-indices-in-cluster-sigma],is_near_field

For example,

    [1 2 3 ...],[7 8 9 ...],1

Parameters

out	

Print index set of cluster τ.

Print index set of cluster σ.

Print the `is_near_field` flag.

Referenced by main().

8.6.4.38 write_leaf_set() [2/2]

```
template<int spacedim, typename Number >
template<typename Number1 >
void BlockClusterTree< spacedim, Number >::write_leaf_set (
            std::ostream & out,
            const LAPACKFullMatrixExt< Number1 > & matrix,
            const Number1 singular_value_threshold = 0. ) const
```

Write formatted leaf set to the output stream as well as the rank of each matrix block.

Each leaf node is written in the following format:

[list-of-indices-in-cluster-tau],[list-of-indices-in-cluster-sigma],is_near_field,rank

For example,

[1 2 3 ...],[7 8 9 ...],1,1

Parameters

out	

Print index set of cluster τ.

Print index set of cluster σ.

Print the `is_near_field` flag.

Make a local copy of the matrix block and calculate its rank using SVD.

Print the `rank` flag.

8.6.5 Friends And Related Function Documentation

8.6.5.1 operator<<

```
template<int spacedim, typename Number = double>
template<int spacedim1, typename Number1 >
std::ostream& operator<< (
            std::ostream & out,
            const BlockClusterTree< spacedim1, Number1 > & block_cluster_tree )  [friend]
```

Print a whole block cluster tree using recursion.

Parameters

out	
block_cluster_tree	

Returns

8.6.6 Member Data Documentation

8.6.6.1 child_num

```
template<int spacedim, typename Number = double>
const unsigned int BlockClusterTree< spacedim, Number >::child_num = 4  [static]
```

Number of children in a block cluster tree.

At present, only quad-tree is allowed.

8.6.6.2 depth

```
template<int spacedim, typename Number = double>
unsigned int BlockClusterTree< spacedim, Number >::depth  [private]
```

Depth of the tree, which is the maximum level plus one.

Referenced by BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), BlockClusterTree< spacedim, Number >::clear(), BlockClusterTree< spacedim, Number >::get_depth(), BlockClusterTree< spacedim, Number >::operator=(), and BlockClusterTree< spacedim, Number >::release().

8.6.6.3 max_level

```
template<int spacedim, typename Number = double>
int BlockClusterTree< spacedim, Number >::max_level [private]
```

Maximum node level in the tree, which is `depth` - 1.

Referenced by BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), BlockClusterTree< spacedim, Number >::clear(), BlockClusterTree< spacedim, Number >::get_max_level(), BlockClusterTree< spacedim, Number >::operator=(), and BlockClusterTree< spacedim, Number >::release().

8.6.6.4 node_num

```
template<int spacedim, typename Number = double>
unsigned int BlockClusterTree< spacedim, Number >::node_num [private]
```

Total number of block clusters in the tree.

Referenced by BlockClusterTree< spacedim, Number >::clear(), BlockClusterTree< spacedim, Number >::get_↩ node_num(), BlockClusterTree< spacedim, Number >::increase_node_num(), BlockClusterTree< spacedim, Number >::operator=(), BlockClusterTree< spacedim, Number >::release(), and BlockClusterTree< spacedim, Number >::set_node_num().

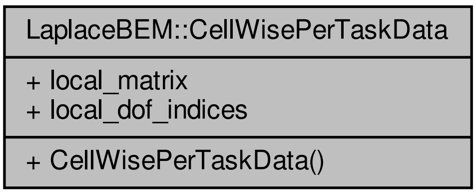
The documentation for this class was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/block_cluster_tree.h

8.7 LaplaceBEM::CellWisePerTaskData Struct Reference

```
#include <laplace_bem.h>
```

Collaboration diagram for LaplaceBEM::CellWisePerTaskData:



Public Member Functions

- **CellWisePerTaskData** (const FiniteElement< 2, 3 > &fe)

Public Attributes

- FullMatrix< double > **local_matrix**
- std::vector< types::global_dof_index > **local_dof_indices**

8.7.1 Detailed Description

Structure holding cell-wise local matrix data and DoF indices, which is used for SMP parallel computation of the term $(v, {1}{2}u)$.

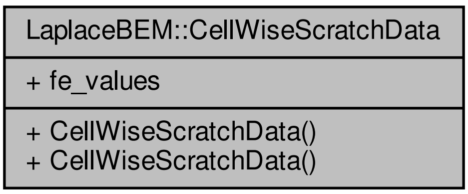
The documentation for this struct was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.8 LaplaceBEM::CellWiseScratchData Struct Reference

```
#include <laplace_bem.h>
```

Collaboration diagram for LaplaceBEM::CellWiseScratchData:



Public Member Functions

- CellWiseScratchData (const FiniteElement< 2, 3 > &fe, const Quadrature< 2 > &quadrature, const UpdateFlags update_flags)
- CellWiseScratchData (const CellWiseScratchData &scratch_data)

Public Attributes

- FEValues< 2, 3 > **fe_values**

8.8.1 Detailed Description

Structure holding temporary data which are needed for cell-wise integration for the term $(v, {1}{2}u)$.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 CellWiseScratchData() [1/2]

```
LaplaceBEM::CellWiseScratchData::CellWiseScratchData (
            const FiniteElement< 2, 3 > & fe,
            const Quadrature< 2 > & quadrature,
            const UpdateFlags update_flags )  [inline]
```

Constructor for the structure.

Parameters

fe	
quadrature	
update_flags	

8.8.2.2 CellWiseScratchData() [2/2]

```
LaplaceBEM::CellWiseScratchData::CellWiseScratchData (
            const CellWiseScratchData & scratch_data )  [inline]
```

Copy constructor for the structure. Because `FEValues` is neither copyable nor has it copy constructor, this copy constructor is mandatory for replication into each task.

Parameters

scratch_data	

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

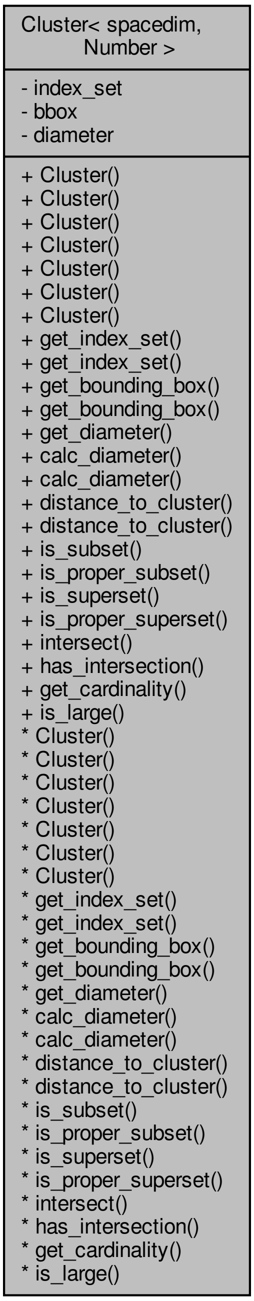
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.9 Cluster< spacedim, Number > Class Template Reference

Class for an index cluster.

```
#include <cluster.h>
```

Collaboration diagram for Cluster< spacedim, Number >:



Public Member Functions

- Cluster ()
- Cluster (const std::vector< types::global_dof_index > &index_set)

- Cluster (const std::vector< types::global_dof_index > &index_set, const std::vector< Point< spacedim, Number >> &all_support_points)
- Cluster (const std::vector< types::global_dof_index > &index_set, const std::vector< Point< spacedim, Number >> &all_support_points, const std::vector< Number > &cell_size_at_dofs)
- Cluster (const std::vector< types::global_dof_index > &index_set, const SimpleBoundingBox< spacedim, Number > &bbox, const std::vector< Point< spacedim, Number >> &all_support_points)
- Cluster (const std::vector< types::global_dof_index > &index_set, const SimpleBoundingBox< spacedim, Number > &bbox, const std::vector< Point< spacedim, Number >> &all_support_points, const std::vector< Number > &cell_size_at_dofs)
- Cluster (const Cluster< spacedim, Number > &cluster)
- std::vector< types::global_dof_index > & get_index_set ()
- const std::vector< types::global_dof_index > & get_index_set () const
- SimpleBoundingBox< spacedim, Number > & get_bounding_box ()
- const SimpleBoundingBox< spacedim, Number > & get_bounding_box () const
- Number get_diameter () const
- Number calc_diameter (const std::vector< Point< spacedim, Number >> &all_support_points) const
- Number calc_diameter (const std::vector< Point< spacedim, Number >> &all_support_points, const std::vector< Number > &cell_size_at_dofs) const
- Number distance_to_cluster (const Cluster &cluster, const std::vector< Point< spacedim, Number >> &all_support_points) const
- Number distance_to_cluster (const Cluster &cluster, const std::vector< Point< spacedim, Number >> &all_support_points, const std::vector< Number > &cell_size_at_dofs) const
- bool is_subset (const Cluster &cluster) const
- bool is_proper_subset (const Cluster &cluster) const
- bool is_superset (const Cluster &cluster) const
- bool is_proper_superset (const Cluster &cluster) const
- void intersect (const Cluster &cluster, std::vector< types::global_dof_index > &index_set_intersection) const
- bool has_intersection (const Cluster &cluster) const
- std::size_t get_cardinality () const
- bool is_large (unsigned int n_min) const

Private Attributes

- std::vector< types::global_dof_index > **index_set**
- SimpleBoundingBox< spacedim, Number > **bbox**
- Number **diameter**

Friends

- template<int spacedim1, typename Number1 >
  std::ostream & **operator<<** (std::ostream &out, const Cluster< spacedim1, Number1 > &cluster)
- template<int spacedim1, typename Number1 >
  Number1 **calc_cluster_distance** (const Cluster< spacedim1, Number1 > &cluster1, const Cluster< spacedim1, Number1 > &cluster2, const std::vector< Point< spacedim1, Number1 >> &all_support_points)
- template<int spacedim1, typename Number1 >
  Number1 **calc_cluster_distance** (const Cluster< spacedim1, Number1 > &cluster1, const Cluster< spacedim1, Number1 > &cluster2, const std::vector< Point< spacedim1, Number1 >> &all_support_points, const std::vector< Number1 > &cell_size_at_dofs)
- template<int spacedim1, typename Number1 >
  bool operator== (const Cluster< spacedim1, Number1 > &cluster1, const Cluster< spacedim1, Number1 > &cluster2)

8.9.1 Detailed Description

template<int spacedim, typename Number = double>
class Cluster< spacedim, Number >

Class for an index cluster.

The `Cluster` class contains both the DoF index set `index_set` and the corresponding bounding box `bbox`.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 Cluster() [1/7]

```
template<int spacedim, typename Number >
Cluster< spacedim, Number >::Cluster ( )
```

Default constructor.

8.9.2.2 Cluster() [2/7]

```
template<int spacedim, typename Number >
Cluster< spacedim, Number >::Cluster (
            const std::vector< types::global_dof_index > & index_set )
```

Constructor from an index set only without support points and associated bounding box.

Parameters

index_set	

8.9.2.3 Cluster() [3/7]

```
template<int spacedim, typename Number >
Cluster< spacedim, Number >::Cluster (
            const std::vector< types::global_dof_index > & index_set,
            const std::vector< Point< spacedim, Number >> & all_support_points )
```

Constructor from an index set without cluster diameter correction.

The bounding box will be recalculated.

Parameters

index_set	
all_support_points	

8.9.2.4 Cluster() [4/7]

```
template<int spacedim, typename Number >
Cluster< spacedim, Number >::Cluster (
            const std::vector< types::global_dof_index > & index_set,
            const std::vector< Point< spacedim, Number >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs )
```

Constructor from an index set with cluster diameter correction.

The bounding box will be recalculated.

Parameters

index_set	
all_support_points	

8.9.2.5 Cluster() [5/7]

```
template<int spacedim, typename Number >
Cluster< spacedim, Number >::Cluster (
            const std::vector< types::global_dof_index > & index_set,
            const SimpleBoundingBox< spacedim, Number > & bbox,
            const std::vector< Point< spacedim, Number >> & all_support_points )
```

Constructor from an index set and a bounding box without cluster diameter correction.

The input bounding box will be copied into the cluster without recalculation. However, the diameter of the cluster is recalculated.

Parameters

index_set	
bbox	

8.9.2.6 Cluster() [6/7]

```
template<int spacedim, typename Number >
Cluster< spacedim, Number >::Cluster (
            const std::vector< types::global_dof_index > & index_set,
            const SimpleBoundingBox< spacedim, Number > & bbox,
            const std::vector< Point< spacedim, Number >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs )
```

Constructor from an index set and a bounding box with cluster diameter correction.

The input bounding box will be copied into the cluster without recalculation. However, the diameter of the cluster is recalculated.

Parameters

index_set	
bbox	

8.9.2.7 Cluster() [7/7]

```
template<int spacedim, typename Number >
Cluster< spacedim, Number >::Cluster (
            const Cluster< spacedim, Number > & cluster )
```

Copy constructor.

8.9.3 Member Function Documentation

8.9.3.1 calc_diameter() [1/2]

```
template<int spacedim, typename Number >
Number Cluster< spacedim, Number >::calc_diameter (
            const std::vector< Point< spacedim, Number >> & all_support_points ) const
```

Calculate the diameter of the cluster. There is no cell size correction. Calculate the number of point pairs in the cluster. Let $[0, 1, 2, 3, 4, 5]$ be the indices of support points in the cluster, whose pairwise inter-distance will be calculated. The calculation is only needed for the marked pairs of points as shown below.

```
  0 1 2 3 4 5
0   - - - - -
1     - - - -
2       - - -
3         - -
4           -
5
```

Therefore, the total number of effective point pairs is $\frac{n^2-n}{2}$.

Referenced by Cluster< spacedim, Number >::calc_diameter().

8.9.3.2 calc_diameter() [2/2]

```
template<int spacedim, typename Number >
Number Cluster< spacedim, Number >::calc_diameter (
            const std::vector< Point< spacedim, Number >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs ) const
```

Calculate the diameter of the cluster. Cell size correction is applied.

N.B. Doubled estimated cell size is adopted as an approximation of the support set diameter $\mathrm{diam}(Q_j)$. The correction is calculated according to the following formula.

$$\widetilde{\mathrm{diam}}(\tau) := \mathrm{diam}(\hat{Q}_\tau) + \max_{j \in \tau} \mathrm{diam}(Q_j)$$

Parameters

all_support_points	
cell_size_at_dofs	

Returns

References Cluster< spacedim, Number >::calc_diameter().

8.9.3.3 distance_to_cluster() [1/2]

```
template<int spacedim, typename Number >
Number Cluster< spacedim, Number >::distance_to_cluster (
            const Cluster< spacedim, Number > & cluster,
            const std::vector< Point< spacedim, Number >> & all_support_points ) const
```

Calculate the minimum distance of the current cluster to the given cluster. There is no cell size correction.

8.9.3.4 distance_to_cluster() [2/2]

```
template<int spacedim, typename Number >
Number Cluster< spacedim, Number >::distance_to_cluster (
            const Cluster< spacedim, Number > & cluster,
            const std::vector< Point< spacedim, Number >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs ) const
```

Calculate the minimum distance of the current cluster to the given cluster. Cell size correction is applied.

8.9.3.5 get_bounding_box() [1/2]

```
template<int spacedim, typename Number >
SimpleBoundingBox< spacedim, Number > & Cluster< spacedim, Number >::get_bounding_box ( )
```

Get the reference to the bounding box.

8.9.3.6 get_bounding_box() [2/2]

```
template<int spacedim, typename Number >
const SimpleBoundingBox< spacedim, Number > & Cluster< spacedim, Number >::get_bounding_box (
) const
```

Get the reference to the bounding box (const version).

8.9.3.7 get_cardinality()

```
template<int spacedim, typename Number >
std::size_t Cluster< spacedim, Number >::get_cardinality ( ) const
```

Get the cardinality of the index set.

Returns

8.9.3.8 get_diameter()

```
template<int spacedim, typename Number >
Number Cluster< spacedim, Number >::get_diameter ( ) const
```

Get the diameter of the cluster.

8.9.3.9 get_index_set() [1/2]

```
template<int spacedim, typename Number >
std::vector< types::global_dof_index > & Cluster< spacedim, Number >::get_index_set ( )
```

Get the reference to the index set.

Referenced by calc_cluster_distance().

8.9.3.10 get_index_set() [2/2]

```
template<int spacedim, typename Number >
const std::vector< types::global_dof_index > & Cluster< spacedim, Number >::get_index_set ( )
const
```

Get the reference to the index set (const version).

8.9.3.11 has_intersection()

```
template<int spacedim, typename Number >
bool Cluster< spacedim, Number >::has_intersection (
            const Cluster< spacedim, Number > & cluster ) const
```

Determine if the index set of the current cluster has a nonempty intersection with the index set of the given cluster.

Note The index sets associated with clusters should be sorted before calling this function. In the current implementation of cluster tree construction, all the index sets have already been sorted.

Parameters

cluster	

Returns

References Cluster< spacedim, Number >::intersect().

8.9.3.12 intersect()

```
template<int spacedim, typename Number >
void Cluster< spacedim, Number >::intersect (
            const Cluster< spacedim, Number > & cluster,
            std::vector< types::global_dof_index > & index_set_intersection ) const
```

Calculate the intersection of the index sets of the current and the given clusters.

Note The index sets associated with clusters should be sorted before calling this function. In the current implementation of cluster tree construction, all the index sets have already been sorted.

Parameters

cluster	
index_set_intersection	

Referenced by Cluster< spacedim, Number >::has_intersection().

8.9.3.13 is_large()

```
template<int spacedim, typename Number >
bool Cluster< spacedim, Number >::is_large (
            unsigned int n_min ) const
```

Determine if the cluster is large enough.

Parameters

n_min	The size threshold value for determining if a cluster is large.

Returns

References Cluster< spacedim, Number >::operator==.

8.9.3.14 is_proper_subset()

```
template<int spacedim, typename Number >
bool Cluster< spacedim, Number >::is_proper_subset (
            const Cluster< spacedim, Number > & cluster ) const
```

Check if the index set of the current cluster is a proper subset of that of the given cluster.

Note The index sets associated with clusters should be sorted before calling this function. In the current implementation of cluster tree construction, all the index sets have already been sorted.

Parameters

cluster	

Returns

8.9.3.15 is_proper_superset()

```
template<int spacedim, typename Number >
bool Cluster< spacedim, Number >::is_proper_superset (
            const Cluster< spacedim, Number > & cluster ) const
```

Check if the index set of the current cluster is a proper superset of that of the given cluster.

Note The index sets associated with clusters should be sorted before calling this function. In the current implementation of cluster tree construction, all the index sets have already been sorted.

Parameters

cluster	

Returns

8.9.3.16 is_subset()

```
template<int spacedim, typename Number >
bool Cluster< spacedim, Number >::is_subset (
            const Cluster< spacedim, Number > & cluster ) const
```

Check if the index set of the current cluster is a subset of that of the given cluster.

Note The index sets associated with clusters should be sorted before calling this function. In the current implementation of cluster tree construction, all the index sets have already been sorted.

Parameters

cluster	

Returns

8.9.3.17 is_superset()

```
template<int spacedim, typename Number >
bool Cluster< spacedim, Number >::is_superset (
            const Cluster< spacedim, Number > & cluster ) const
```

Check if the index set of the current cluster is a superset of that of the given cluster.

Note The index sets associated with clusters should be sorted before calling this function. In the current implementation of cluster tree construction, all the index sets have already been sorted.

Parameters

cluster	

Returns

8.9.4 Friends And Related Function Documentation

8.9.4.1 operator==

```
template<int spacedim, typename Number = double>
template<int spacedim1, typename Number1 >
bool operator== (
            const Cluster< spacedim1, Number1 > & cluster1,
            const Cluster< spacedim1, Number1 > & cluster2 )  [friend]
```

Check the equality of two clusters by comparing their index sets.

This function firstly check the equality of the sizes/cardinalities of the index sets in the two clusters. If their sizes are equal, then check the contents.

Parameters

cluster1	
cluster2	

Returns

Referenced by Cluster< spacedim, Number >::is_large().

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

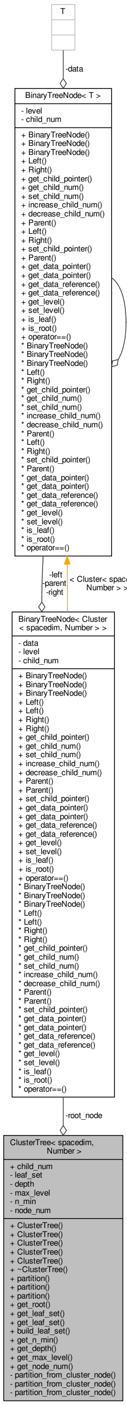
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/cluster.h

8.10 ClusterTree< spacedim, Number > Class Template Reference

Class for cluster tree.

```
#include <cluster_tree.h>
```

Collaboration diagram for ClusterTree< spacedim, Number >:



Public Types

- typedef BinaryTreeNode< Cluster< spacedim, Number > > node_value_type
- typedef BinaryTreeNode< Cluster< spacedim, Number > > ∗ node_pointer_type
- typedef const BinaryTreeNode< Cluster< spacedim, Number > > ∗ node_const_pointer_type
- typedef BinaryTreeNode< Cluster< spacedim, Number > > & node_reference_type
- typedef const BinaryTreeNode< Cluster< spacedim, Number > > & node_const_reference_type

- typedef Cluster< spacedim, Number > data_value_type
- typedef Cluster< spacedim, Number > * data_pointer_type
- typedef const Cluster< spacedim, Number > * data_const_pointer_type
- typedef Cluster< spacedim, Number > & data_reference_type
- typedef const Cluster< spacedim, Number > & data_const_reference_type

Public Member Functions

- ClusterTree ()
- ClusterTree (const std::vector< types::global_dof_index > &index_set, const unsigned int n_min)
- ClusterTree (const std::vector< types::global_dof_index > &index_set, const std::vector< Point< spacedim >> &all_support_points, const unsigned int n_min)
- ClusterTree (const std::vector< types::global_dof_index > &index_set, const std::vector< Point< spacedim >> &all_support_points, const std::vector< Number > &cell_size_at_dofs, const unsigned int n_min)
- ClusterTree (const ClusterTree< spacedim, Number > &cluster_tree)
- ∼ClusterTree ()
- void partition ()
- void partition (const std::vector< Point< spacedim >> &all_support_points)
- void partition (const std::vector< Point< spacedim >> &all_support_points, const std::vector< Number > &cell_size_at_dofs)
- node_pointer_type get_root () const
- std::vector< node_pointer_type > & get_leaf_set ()
- const std::vector< node_pointer_type > & get_leaf_set () const
- void build_leaf_set ()
- unsigned int get_n_min () const
- unsigned int get_depth () const
- unsigned int get_max_level () const
- unsigned int get_node_num () const

Static Public Attributes

- static const unsigned int child_num = 2

Private Member Functions

- void partition_from_cluster_node (node_pointer_type current_cluster_node, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- void partition_from_cluster_node (node_pointer_type current_cluster_node, const std::vector< Point< spacedim >> &all_support_points, std::vector< node_pointer_type > &leaf_set_wrt_current_node)
- void partition_from_cluster_node (node_pointer_type current_cluster_node, const std::vector< Point< spacedim >> &all_support_points, const std::vector< Number > &cell_size_at_dofs, std::vector< node_↩ pointer_type > &leaf_set_wrt_current_node)

Private Attributes

- node_pointer_type **root_node**
- std::vector< node_pointer_type > **leaf_set**
- unsigned int depth
- int max_level
- const unsigned int n_min
- unsigned int node_num

Friends

- template<int spacedim1, typename Number1 >
  std::ostream & operator<< (std::ostream &out, const ClusterTree< spacedim1, Number1 > &cluster_tree)

8.10.1 Detailed Description

template<int spacedim, typename Number = double>
class ClusterTree< spacedim, Number >

Class for cluster tree.

A cluster tree is a binary tree which holds a hierarchy of linked nodes with the type BinaryTreeNode.

8.10.2 Member Typedef Documentation

8.10.2.1 data_const_pointer_type

```
template<int spacedim, typename Number = double>
typedef const Cluster<spacedim, Number>* ClusterTree< spacedim, Number >::data_const_pointer↩
_type
```

Const pointer type for the content held by a node in the ClusterTree.

8.10.2.2 data_const_reference_type

```
template<int spacedim, typename Number = double>
typedef const Cluster<spacedim, Number>& ClusterTree< spacedim, Number >::data_const_reference↩
_type
```

Const reference type for the content held by a node in the ClusterTree.

8.10.2.3 data_pointer_type

```
template<int spacedim, typename Number = double>
typedef Cluster<spacedim, Number>* ClusterTree< spacedim, Number >::data_pointer_type
```

Pointer type for the content held by a node in the ClusterTree.

8.10.2.4 data_reference_type

```
template<int spacedim, typename Number = double>
typedef Cluster<spacedim, Number>& ClusterTree< spacedim, Number >::data_reference_type
```

Reference type for the content held by a node in the ClusterTree.

8.10.2.5 data_value_type

```
template<int spacedim, typename Number = double>
typedef Cluster<spacedim, Number> ClusterTree< spacedim, Number >::data_value_type
```

Data type for the content held by a node in the ClusterTree.

8.10.2.6 node_const_pointer_type

```
template<int spacedim, typename Number = double>
typedef const BinaryTreeNode<Cluster<spacedim, Number> >* ClusterTree< spacedim, Number >↩
::node_const_pointer_type
```

Const pointer type for a node in the ClusterTree.

8.10.2.7 node_const_reference_type

```
template<int spacedim, typename Number = double>
typedef const BinaryTreeNode<Cluster<spacedim, Number> >& ClusterTree< spacedim, Number >↩
::node_const_reference_type
```

Const reference type for a node in the ClusterTree.

8.10.2.8 node_pointer_type

```
template<int spacedim, typename Number = double>
typedef BinaryTreeNode<Cluster<spacedim, Number> >* ClusterTree< spacedim, Number >::node_↩
pointer_type
```

Pointer type for a node in the ClusterTree.

8.10.2.9 node_reference_type

```
template<int spacedim, typename Number = double>
typedef BinaryTreeNode<Cluster<spacedim, Number> >& ClusterTree< spacedim, Number >::node_↩
reference_type
```

Reference type for a node in the ClusterTree.

8.10.2.10 node_value_type

```
template<int spacedim, typename Number = double>
typedef BinaryTreeNode<Cluster<spacedim, Number> > ClusterTree< spacedim, Number >::node_↩
value_type
```

Data type for a node in the ClusterTree.

8.10.3 Constructor & Destructor Documentation

8.10.3.1 ClusterTree() [1/5]

```
template<int spacedim, typename Number >
ClusterTree< spacedim, Number >::ClusterTree ( )
```

Default constructor, which initializes an empty binary tree.

8.10.3.2 ClusterTree() [2/5]

```
template<int spacedim, typename Number >
ClusterTree< spacedim, Number >::ClusterTree (
            const std::vector< types::global_dof_index > & index_set,
            const unsigned int n_min )
```

Construct from only an index set without support point and the partition will be only based on the cardinality of the index set.

Parameters

index_set	
n_min	

References ClusterTree< spacedim, Number >::depth, ClusterTree< spacedim, Number >::max_level, and ClusterTree< spacedim, Number >::node_num.

8.10.3.3 ClusterTree() [3/5]

```
template<int spacedim, typename Number >
ClusterTree< spacedim, Number >::ClusterTree (
            const std::vector< types::global_dof_index > & index_set,
            const std::vector< Point< spacedim >> & all_support_points,
            const unsigned int n_min )
```

Constructor from a full index set and associated support point coordinates.

This constructor will create the root node of the cluster tree based on the given data. There is no mesh cell size correction for the cluster diameter.

Parameters

index_set	The full DoF index set, which will be assigned to the root node.
all_support_points	All the support points.

References ClusterTree< spacedim, Number >::depth, ClusterTree< spacedim, Number >::max_level, and ClusterTree< spacedim, Number >::node_num.

8.10.3.4 ClusterTree() [4/5]

```
template<int spacedim, typename Number >
ClusterTree< spacedim, Number >::ClusterTree (
            const std::vector< types::global_dof_index > & index_set,
            const std::vector< Point< spacedim >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs,
            const unsigned int n_min )
```

Constructor from a full index set and associated support point coordinates.

This constructor will create the root node of the cluster tree based on the given data. There is mesh cell size correction for the cluster diameter.

Parameters

index_set	The full DoF index set, which will be assigned to the root node.
all_support_points	All the support points.

References ClusterTree< spacedim, Number >::depth, ClusterTree< spacedim, Number >::max_level, and ClusterTree< spacedim, Number >::node_num.

8.10.3.5 ClusterTree() [5/5]

```
template<int spacedim, typename Number >
ClusterTree< spacedim, Number >::ClusterTree (
            const ClusterTree< spacedim, Number > & cluster_tree )
```

Copy constructor.

Parameters

cluster_tree	

References ClusterTree< spacedim, Number >::build_leaf_set(), CopyTree(), and ClusterTree< spacedim, Number >::get_root().

8.10.3.6 ∼ClusterTree()

```
template<int spacedim, typename Number >
ClusterTree< spacedim, Number >::∼ClusterTree ( )
```

Destructor which recursively destroys every node in the cluster tree.

References DeleteTree().

8.10.4 Member Function Documentation

8.10.4.1 build_leaf_set()

```
template<int spacedim, typename Number >
void ClusterTree< spacedim, Number >::build_leaf_set ( )
```

Build the leaf set by tree recursion.

References GetTreeLeaves().

Referenced by ClusterTree< spacedim, Number >::ClusterTree().

8.10.4.2 get_depth()

```
template<int spacedim, typename Number >
unsigned int ClusterTree< spacedim, Number >::get_depth ( ) const
```

Get the tree depth.

References ClusterTree< spacedim, Number >::depth.

8.10.4.3 get_leaf_set() [1/2]

```
template<int spacedim, typename Number >
std::vector< typename ClusterTree< spacedim, Number >::node_pointer_type > & ClusterTree<
spacedim, Number >::get_leaf_set ( )
```

Get the reference to the block cluster list.

Referenced by main().

8.10.4.4 get_leaf_set() [2/2]

```
template<int spacedim, typename Number >
const std::vector< typename ClusterTree< spacedim, Number >::node_pointer_type > & Cluster←
Tree< spacedim, Number >::get_leaf_set ( ) const
```

Get the reference to the block cluster list (const version).

8.10.4.5 get_max_level()

```
template<int spacedim, typename Number >
unsigned int ClusterTree< spacedim, Number >::get_max_level ( ) const
```

Get the maximum tree level.

References ClusterTree< spacedim, Number >::max_level.

8.10.4.6 get_n_min()

```
template<int spacedim, typename Number >
unsigned int ClusterTree< spacedim, Number >::get_n_min ( ) const
```

Get the minimum cluster size.

References ClusterTree< spacedim, Number >::n_min.

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree().

8.10.4.7 get_node_num()

```
template<int spacedim, typename Number >
unsigned int ClusterTree< spacedim, Number >::get_node_num ( ) const
```

Get the total number of clusters in the tree.

References ClusterTree< spacedim, Number >::node_num.

8.10.4.8 get_root()

```
template<int spacedim, typename Number >
ClusterTree< spacedim, Number >::node_pointer_type ClusterTree< spacedim, Number >::get_root
( ) const
```

Get the pointer to the root node of the cluster tree.

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree(), and ClusterTree< spacedim, Number >::ClusterTree().

8.10.4.9 partition() [1/3]

```
template<int spacedim, typename Number >
void ClusterTree< spacedim, Number >::partition ( )
```

Perform a pure cardinality based recursive partition, which will ultimately be used in constructing an $\mathcal{H}^p$ matrix for example.

Note

1. If the initial complete cluster index set I is sorted, which is usually $[0, 1, \cdots, N]$, the cardinality based cluster partition produces cluster index sets following the same order, i.e. the cardinality based partition is order preserving.
2. If the initial complete cluster index set I is also continuous, i.e. it is a continuous integer array, the cardinality based cluster partition also produces continuous cluster index sets. Hence, the cardinality based partition is continuity preserving.

References calc_depth(), ClusterTree< spacedim, Number >::depth, ClusterTree< spacedim, Number >::max_↩ level, and ClusterTree< spacedim, Number >::partition_from_cluster_node().

Referenced by main().

8.10.4.10 partition() [2/3]

```
template<int spacedim, typename Number >
void ClusterTree< spacedim, Number >::partition (
            const std::vector< Point< spacedim >> & all_support_points )
```

Perform a recursive partition dependent on the coordinates of DoF support points by starting from the root node.

In this version, there is no mesh cell size correction to the cluster diameter and cluster pair distance.

Note

1. If the initial complete cluster index set I is sorted, which is usually $[0, 1, \cdots, N]$, the support point coordinates based partition is also order preserving. This is because the two child clusters of the current cluster are built by scanning the index set of the current cluster from beginning to end.
2. The support point coordinates based partition is not continuity preserving.

References calc_depth(), ClusterTree< spacedim, Number >::depth, ClusterTree< spacedim, Number >::max_↩ level, and ClusterTree< spacedim, Number >::partition_from_cluster_node().

8.10.4.11 partition() [3/3]

```
template<int spacedim, typename Number >
void ClusterTree< spacedim, Number >::partition (
            const std::vector< Point< spacedim >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs )
```

Perform a recursive partition dependent on the coordinates of DoF support points by starting from the root node.

In this version, there is mesh cell size correction to the cluster diameter and cluster pair distance.

Note

1. If the initial complete cluster index set I is sorted, which is usually $[0, 1, \cdots, N]$, the support point coordinates based partition is also order preserving. This is because the two child clusters of the current cluster are built by scanning the index set of the current cluster from beginning to end.
2. The support point coordinates based partition is not continuity preserving.

References calc_depth(), ClusterTree< spacedim, Number >::depth, ClusterTree< spacedim, Number >::max_↩ level, and ClusterTree< spacedim, Number >::partition_from_cluster_node().

8.10.4.12 partition_from_cluster_node() [1/3]

```
template<int spacedim, typename Number >
void ClusterTree< spacedim, Number >::partition_from_cluster_node (
            node_pointer_type current_cluster_node,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a pure cardinality based recursive partition by starting from a cluster node.

Note

1. If the initial complete cluster index set I is sorted, which is usually $[0, 1, \cdots, N]$, the cardinality based cluster partition produces cluster index sets following the same order, i.e. the cardinality based partition is order preserving.
2. If the initial complete cluster index set I is also continuous, i.e. it is a continuous integer array, the cardinality based cluster partition also produces continuous cluster index sets. Hence, the cardinality based partition is continuity preserving.

Parameters

current_cluster_node	
leaf_set_wrt_current_node	

When the cardinality of the current cluster is large enough, continue the partition.

Declare the two child index sets.

Split the index set of the current node into halves.

Calculate the splitting index in the middle of the index set, which is to be used for constructing half-closed and half-open subintervals.

Construct the left child index set.

Construct the right child index set.

Append this new node as the left child of the current cluster node.

Continue the recursive partition by starting from this child node.

Merge the leaf set wrt. the child cluster node into the leaf set of the current cluster node.

Append this new node as the right child of the current cluster node.

Continue the recursive partition by starting from this child node.

Merge the leaf set wrt. the child cluster node into the leaf set of the current cluster node.

References BinaryTreeNode< T >::get_data_pointer(), BinaryTreeNode< T >::get_level(), BinaryTreeNode< T >::Left(), ClusterTree< spacedim, Number >::n_min, ClusterTree< spacedim, Number >::node_num, and BinaryTreeNode< T >::Right().

Referenced by ClusterTree< spacedim, Number >::partition(), and ClusterTree< spacedim, Number >::partition↩_from_cluster_node().

8.10.4.13 partition_from_cluster_node() [2/3]

```
template<int spacedim, typename Number >
void ClusterTree< spacedim, Number >::partition_from_cluster_node (
            node_pointer_type current_cluster_node,
            const std::vector< Point< spacedim >> & all_support_points,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a recursive partition dependent on the coordinates of DoF support points by starting from a cluster node.

In this version, there is no mesh cell size correction to the cluster diameter and cluster pair distance.

Note

1. If the initial complete cluster index set I is sorted, which is usually $[0, 1, \cdots, N]$, the support point coordinates based partition is also order preserving. This is because the two child clusters of the current cluster are built by scanning the index set of the current cluster from beginning to end.
2. The support point coordinates based partition is not continuity preserving.

Parameters

all_support_points	All the support points.

When the size/cardinality of the current cluster is large enough, continue the partition.

Divide the bounding box of the current cluster into halves.

Declare the two child index sets.

Determine to which child index set each support point in the original bounding box belongs to.

If the support point associated with the current DoF index belongs to the left child box, add this DoF index to the left child index set.

Otherwise, add this DoF index to the right child index set.

N.B. During the creation of the new child cluster, its bounding box will be recalculated, which may be smaller than the child bounding box obtained from the previous bounding box geometric bisection.

Append this new node as the left child of the current cluster node.

Continue the recursive partition by starting from this child node.

Merge the leaf set wrt. the child cluster node into the leaf set of the current cluster node.

N.B. During the creation of the new child cluster, its bounding box will be recalculated, which may be smaller than the child bounding box obtained from the previous bounding box geometric bisection.

Append this new node as the right child of the current cluster node.

Continue the recursive partition by starting from this child node.

Merge the leaf set wrt. the child cluster node into the leaf set of the current cluster node.

References BinaryTreeNode< T >::get_data_pointer(), BinaryTreeNode< T >::get_level(), BinaryTreeNode< T >::Left(), ClusterTree< spacedim, Number >::n_min, ClusterTree< spacedim, Number >::node_num, ClusterTree< spacedim, Number >::partition_from_cluster_node(), and BinaryTreeNode< T >::Right().

8.10.4.14 partition_from_cluster_node() [3/3]

```
template<int spacedim, typename Number >
void ClusterTree< spacedim, Number >::partition_from_cluster_node (
            node_pointer_type current_cluster_node,
            const std::vector< Point< spacedim >> & all_support_points,
            const std::vector< Number > & cell_size_at_dofs,
            std::vector< node_pointer_type > & leaf_set_wrt_current_node )  [private]
```

Perform a recursive partition dependent on the coordinates of DoF support points by starting from a cluster node.

In this version, there is mesh cell size correction to the cluster diameter and cluster pair distance.

Note

1. If the initial complete cluster index set I is sorted, which is usually $[0, 1, \cdots, N]$, the support point coordinates based partition is also order preserving. This is because the two child clusters of the current cluster are built by scanning the index set of the current cluster from beginning to end.
2. The support point coordinates based partition is not continuity preserving.

Parameters

all_support_points	All the support points.

When the size/cardinality of the current cluster is large enough, continue the partition.

Divide the bounding box of the current cluster into halves.

Declare the two child index sets.

Determine to which child index set each support point in the original bounding box belongs to.

If the support point associated with the current DoF index belongs to the left child box, add this DoF index to the left child index set.

Otherwise, add this DoF index to the right child index set.

N.B. During the creation of the new child cluster, its bounding box will be recalculated, which may be smaller than the child bounding box obtained from the previous bounding box geometric bisection.

Append this new node as the left child of the current cluster node.

Continue the recursive partition by starting from this child node.

Merge the leaf set wrt. the child cluster node into the leaf set of the current cluster node.

N.B. During the creation of the new child cluster, its bounding box will be recalculated, which may be smaller than the child bounding box obtained from the previous bounding box geometric bisection.

Append this new node as the right child of the current cluster node.

Continue the recursive partition by starting from this child node.

Merge the leaf set wrt. the child cluster node into the leaf set of the current cluster node.

References BinaryTreeNode< T >::get_data_pointer(), BinaryTreeNode< T >::get_level(), BinaryTreeNode< T >::Left(), ClusterTree< spacedim, Number >::n_min, ClusterTree< spacedim, Number >::node_num, ClusterTree< spacedim, Number >::partition_from_cluster_node(), and BinaryTreeNode< T >::Right().

8.10.5 Friends And Related Function Documentation

8.10.5.1 operator<<

```
template<int spacedim, typename Number = double>
template<int spacedim1, typename Number1 >
std::ostream& operator<< (
            std::ostream & out,
            const ClusterTree< spacedim1, Number1 > & cluster_tree )  [friend]
```

Print a whole cluster tree using recursion.

Parameters

out	
cluster_tree	

Returns

8.10.6 Member Data Documentation

8.10.6.1 child_num

```
template<int spacedim, typename Number = double>
const unsigned int ClusterTree< spacedim, Number >::child_num = 2  [static]
```

Number of children in a cluster tree.

At present, only binary tree is allowed.

8.10.6.2 depth

```
template<int spacedim, typename Number = double>
unsigned int ClusterTree< spacedim, Number >::depth  [private]
```

Depth of the tree, which is the maximum level plus one.

Referenced by ClusterTree< spacedim, Number >::ClusterTree(), ClusterTree< spacedim, Number >::get_depth(), and ClusterTree< spacedim, Number >::partition().

8.10.6.3 max_level

```
template<int spacedim, typename Number = double>
int ClusterTree< spacedim, Number >::max_level [private]
```

Maximum level of the cluster tree, which is `depth` - 1.

Referenced by ClusterTree< spacedim, Number >::ClusterTree(), ClusterTree< spacedim, Number >::get_max↩
_level(), and ClusterTree< spacedim, Number >::partition().

8.10.6.4 n_min

```
template<int spacedim, typename Number = double>
const unsigned int ClusterTree< spacedim, Number >::n_min [private]
```

Minimum cluster size, which is used as the condition for stopping box division.

Referenced by ClusterTree< spacedim, Number >::get_n_min(), and ClusterTree< spacedim, Number >↩
::partition_from_cluster_node().

8.10.6.5 node_num

```
template<int spacedim, typename Number = double>
unsigned int ClusterTree< spacedim, Number >::node_num [private]
```

Total number of clusters in the tree.

Referenced by ClusterTree< spacedim, Number >::ClusterTree(), ClusterTree< spacedim, Number >::get_node↩
_num(), and ClusterTree< spacedim, Number >::partition_from_cluster_node().

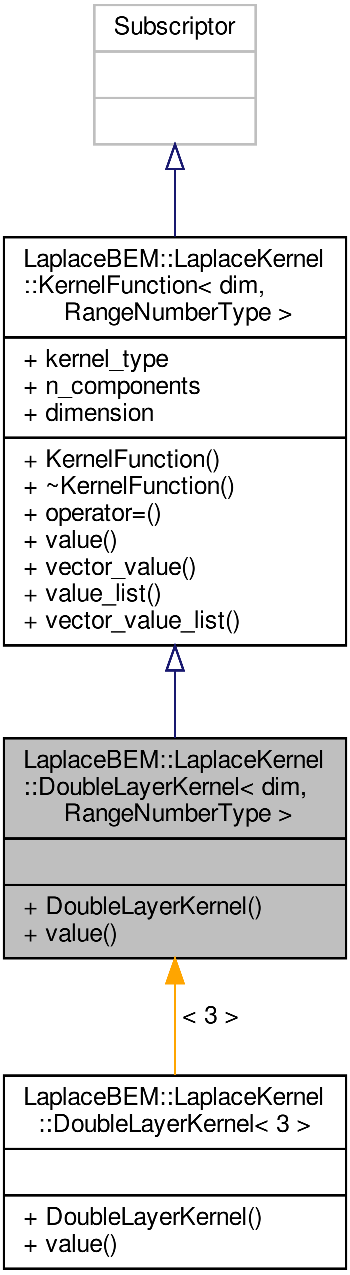
The documentation for this class was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/cluster_tree.h

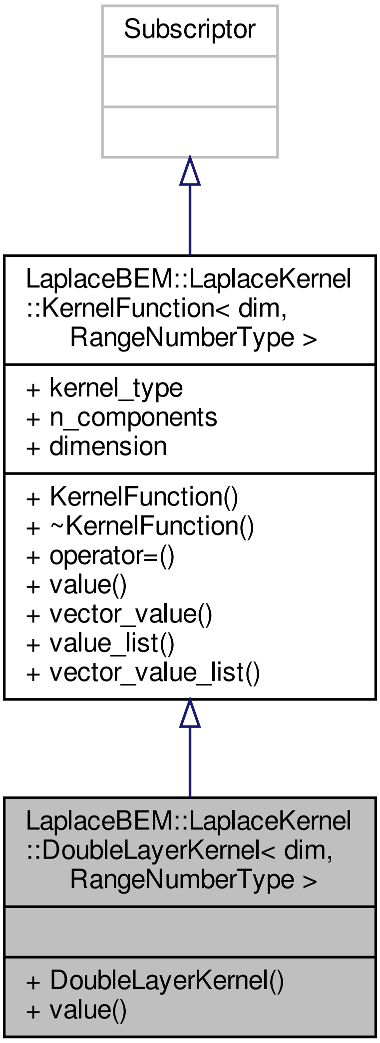
8.11 LaplaceBEM::LaplaceKernel::DoubleLayerKernel< dim, RangeNumberType > Class Template Reference

```
#include <laplace_bem.h>
```

Inheritance diagram for LaplaceBEM::LaplaceKernel::DoubleLayerKernel< dim, RangeNumberType >:



Collaboration diagram for LaplaceBEM::LaplaceKernel::DoubleLayerKernel< dim, RangeNumberType >:



Public Member Functions

- virtual RangeNumberType **value** (const Point< dim > &x, const Point< dim > &y, const Tensor< 1, dim > &nx, const Tensor< 1, dim > &ny, const unsigned int component=0) const override

Additional Inherited Members

8.11.1 Detailed Description

template<int dim, typename RangeNumberType = double>
class LaplaceBEM::LaplaceKernel::DoubleLayerKernel< dim, RangeNumberType >

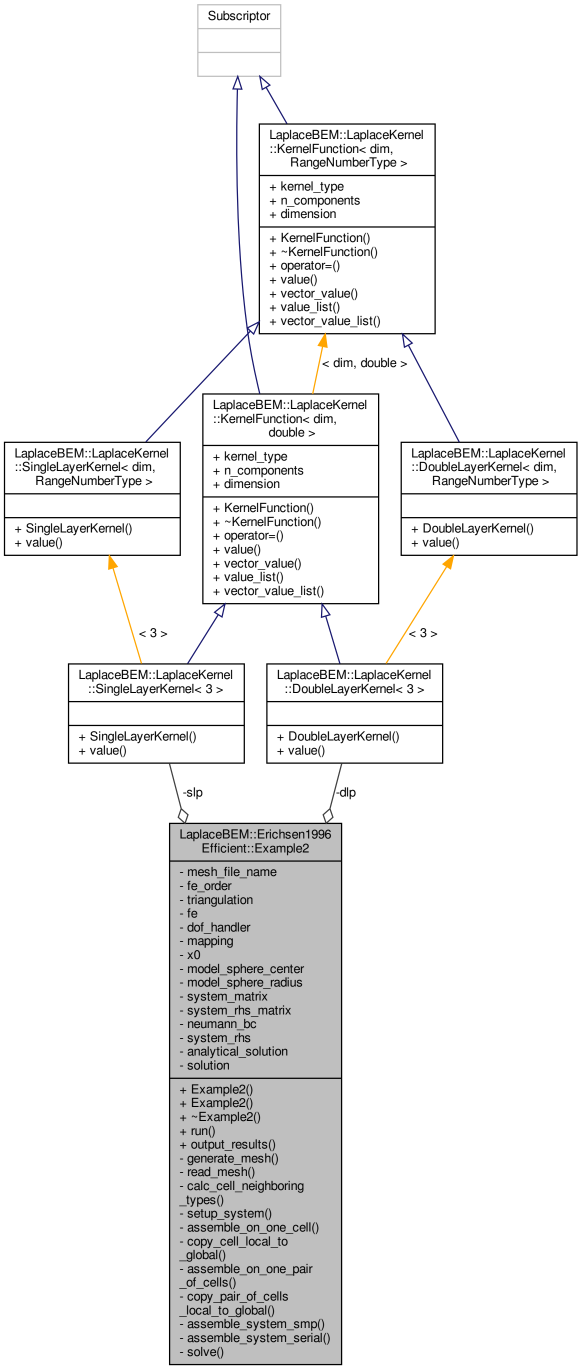
Double layer kernel.

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

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.12 LaplaceBEM::Erichsen1996Efficient::Example2 Class Reference

Collaboration diagram for LaplaceBEM::Erichsen1996Efficient::Example2:



Classes

- class AnalyticalSolution
- class NeumannBC

Public Member Functions

- **Example2** (const std::string &mesh_file_name, unsigned int fe_order=2)
- void **run** ()
- void **output_results** ()

Private Member Functions

- void **generate_mesh** (unsigned int number_of_refinements=0)
- void read_mesh ()
- void calc_cell_neighboring_types ()
- void **setup_system** ()
- void **assemble_on_one_cell** (const typename DoFHandler< 2, 3 >::active_cell_iterator &cell_iter, CellWiseScratchData &scratch, CellWisePerTaskData &data)
- void **copy_cell_local_to_global** (const CellWisePerTaskData &data)
- void assemble_on_one_pair_of_cells (const typename DoFHandler< 2, 3 >::active_cell_iterator &kx_cell_iter, const typename DoFHandler< 2, 3 >::active_cell_iterator &ky_cell_iter, const BEMValues< 2, 3 > &bem_values, PairCellWiseScratchData &scratch, PairCellWisePerTaskData &data)
- void **copy_pair_of_cells_local_to_global** (const PairCellWisePerTaskData &data)
- void **assemble_system_smp** ()
- void **assemble_system_serial** ()
- void **solve** ()

Private Attributes

- std::string **mesh_file_name**
- unsigned int **fe_order**
- Triangulation< 2, 3 > **triangulation**
- FE_Q< 2, 3 > **fe**
- DoFHandler< 2, 3 > **dof_handler**
- MappingQGeneric< 2, 3 > **mapping**
- LaplaceKernel::SingleLayerKernel< 3 > **slp**
- LaplaceKernel::DoubleLayerKernel< 3 > **dlp**
- Point< 3 > **x0**
- Point< 3 > **model_sphere_center**
- double **model_sphere_radius**
- FullMatrix< double > system_matrix
- FullMatrix< double > system_rhs_matrix
- Vector< double > neumann_bc
- Vector< double > system_rhs
- Vector< double > **analytical_solution**
- Vector< double > **solution**

8.12.1 Member Function Documentation

8.12.1.1 assemble_on_one_pair_of_cells()

```
void LaplaceBEM::Erichsen1996Efficient::Example2::assemble_on_one_pair_of_cells (
            const typename DoFHandler< 2, 3 >::active_cell_iterator & kx_cell_iter,
            const typename DoFHandler< 2, 3 >::active_cell_iterator & ky_cell_iter,
            const BEMValues< 2, 3 > & bem_values,
            PairCellWiseScratchData & scratch,
            PairCellWisePerTaskData & data )  [private]
```

Assemble BEM matrices on a pair of cells, i.e. K_x as the field cell and K_y as the source cell.

Parameters

kx_cell_iter	
ky_cell_iter	
scratch	
data	

References LaplaceBEM::hierarchical_support_points_in_real_cell().

8.12.1.2 calc_cell_neighboring_types()

```
void LaplaceBEM::Erichsen1996Efficient::Example2::calc_cell_neighboring_types ( )  [private]
```

Calculate the neighboring type for each pair of cells.

References neumann_bc, system_matrix, system_rhs, and system_rhs_matrix.

8.12.1.3 read_mesh()

```
void LaplaceBEM::Erichsen1996Efficient::Example2::read_mesh ( )  [private]
```

Read the mesh from a file, which abandons the manifold description.

8.12.2 Member Data Documentation

8.12.2.1 neumann_bc

```
Vector<double> LaplaceBEM::Erichsen1996Efficient::Example2::neumann_bc  [private]
```

Neumann boundary condition data at each DoF support point.

Referenced by calc_cell_neighboring_types().

8.12.2.2 system_matrix

`FullMatrix<double> LaplaceBEM::Erichsen1996Efficient::Example2::system_matrix [private]`

System matrix obtained from $(v, {1}{2}u) + (v, Ku)$. The first integral term in the sum is carried on each cell, while the second integral term is carried out on each pair of cells.

Referenced by calc_cell_neighboring_types().

8.12.2.3 system_rhs

`Vector<double> LaplaceBEM::Erichsen1996Efficient::Example2::system_rhs [private]`

Right hand side vector for the problem obtained from the product of `system_rhs_matrix` and `neumann_bc`

Referenced by calc_cell_neighboring_types().

8.12.2.4 system_rhs_matrix

`FullMatrix<double> LaplaceBEM::Erichsen1996Efficient::Example2::system_rhs_matrix [private]`

The right hand side matrix obtained from (v, Vu).

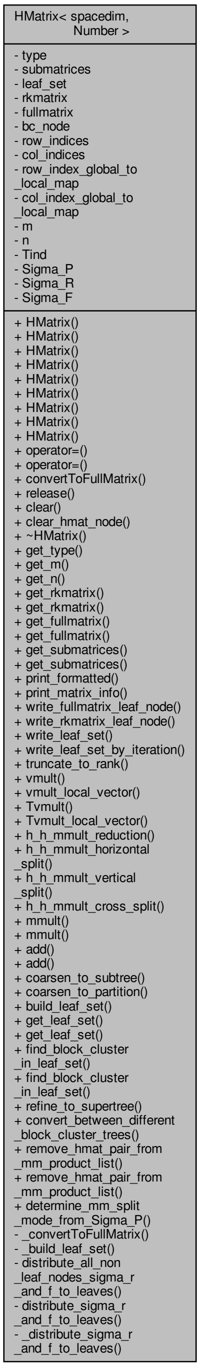
Referenced by calc_cell_neighboring_types().

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

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/erichsen1996efficient_↩ example2.h

8.13 HMatrix< spacedim, Number > Class Template Reference

Collaboration diagram for HMatrix< spacedim, Number >:



Public Types

- using size_type = std::make_unsigned< types::blas_int >::type

Public Member Functions

- HMatrix ()
- HMatrix (const BlockClusterTree< spacedim, Number > &bct, const unsigned int fixed_rank_k=1)
- HMatrix (typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, const unsigned int fixed_rank_k=1)
- HMatrix (const BlockClusterTree< spacedim, Number > &bct, const LAPACKFullMatrixExt< Number > &M, const unsigned int fixed_rank_k=1)
- HMatrix (typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, const LAPACKFullMatrixExt< Number > &M, const unsigned int fixed_rank_k=1)
- HMatrix (typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, HMatrix< spacedim, Number > &&H)
- HMatrix (const BlockClusterTree< spacedim, Number > &bct, HMatrix< spacedim, Number > &&H)
- HMatrix (const HMatrix< spacedim, Number > &H)
- HMatrix (HMatrix< spacedim, Number > &&H)
- HMatrix< spacedim, Number > & operator= (HMatrix< spacedim, Number > &&H)
- HMatrix< spacedim, Number > & operator= (const HMatrix< spacedim, Number > &H)
- template<typename MatrixType >
  void convertToFullMatrix (MatrixType &M) const
- void release ()
- void clear ()
- void clear_hmat_node ()
- ~HMatrix ()
- HMatrixType get_type () const
- size_type get_m () const
- size_type get_n () const
- RkMatrix< Number > * get_rkmatrix ()
- const RkMatrix< Number > * get_rkmatrix () const
- LAPACKFullMatrixExt< Number > * get_fullmatrix ()
- const LAPACKFullMatrixExt< Number > * get_fullmatrix () const
- std::vector< HMatrix< spacedim, Number > * > & get_submatrices ()
- const std::vector< HMatrix< spacedim, Number > * > & get_submatrices () const
- void print_formatted (std::ostream &out, const unsigned int precision=3, const bool scientific=true, const unsigned int width=0, const char *zero_string=" ", const double denominator=1., const double threshold=0.) const
- void print_matrix_info (std::ostream &out) const
- void write_fullmatrix_leaf_node (std::ostream &out, const Number singular_value_threshold=0.) const
- void write_rkmatrix_leaf_node (std::ostream &out) const
- void write_leaf_set (std::ostream &out, const Number singular_value_threshold=0.) const
- void write_leaf_set_by_iteration (std::ostream &out, const Number singular_value_threshold=0.) const
- void truncate_to_rank (size_type new_rank)
- void vmult (Vector< Number > &y, const Vector< Number > &x) const
- void vmult_local_vector (Vector< Number > &y, const std::map< types::global_dof_index, size_t > &y_index_global_to_local_map, const Vector< Number > &x, const std::map< types::global_dof_index, size_t > &x_index_global_to_local_map) const
- void Tvmult (Vector< Number > &y, const Vector< Number > &x) const
- void Tvmult_local_vector (Vector< Number > &y, const std::map< types::global_dof_index, size_t > &y_index_global_to_local_map, const Vector< Number > &x, const std::map< types::global_dof_index, size_t > &x_index_global_to_local_map) const
- void h_h_mmult_reduction ()
- void h_h_mmult_horizontal_split (BlockClusterTree< spacedim, Number > &bc_tree)
- void h_h_mmult_vertical_split (BlockClusterTree< spacedim, Number > &bc_tree)
- void h_h_mmult_cross_split (BlockClusterTree< spacedim, Number > &bc_tree)
- void mmult (HMatrix< spacedim, Number > &C, HMatrix< spacedim, Number > &B, BlockClusterTree< spacedim, Number > &bct_a, BlockClusterTree< spacedim, Number > &bct_b, BlockClusterTree< spacedim, Number > &bct_c, const unsigned int fixed_rank=1)

- void **mmult** (HMatrix< spacedim, Number > &C, HMatrix< spacedim, Number > &B, BlockClusterTree< spacedim, Number > &bct_a, BlockClusterTree< spacedim, Number > &bct_b, BlockClusterTree< spacedim, Number > &bct_c, const unsigned int fixed_rank, const bool adding)
- void add (HMatrix< spacedim, Number > &C, const HMatrix< spacedim, Number > &B, const size_type fixed_rank_k) const
- void add (const HMatrix< spacedim, Number > &B, const size_type fixed_rank_k) const
- void coarsen_to_subtree (const BlockClusterTree< spacedim, Number > &subtree, const unsigned int fixed_rank_k)
- void coarsen_to_partition (const std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer_type > &partition, const unsigned int fixed_rank_k)
- void build_leaf_set ()
- std::vector< HMatrix< spacedim, Number > * > & get_leaf_set ()
- const std::vector< HMatrix< spacedim, Number > * > & get_leaf_set () const
- std::vector< HMatrix< spacedim, Number > * >::iterator find_block_cluster_in_leaf_set (const BlockCluster< spacedim, Number > &block_cluster)
- std::vector< HMatrix< spacedim, Number > * >::const_iterator find_block_cluster_in_leaf_set (const BlockCluster< spacedim, Number > &block_cluster) const
- void refine_to_supertree ()
- void convert_between_different_block_cluster_trees (BlockClusterTree< spacedim, Number > &bct1, BlockClusterTree< spacedim, Number > &bct2, const unsigned int fixed_rank_k2=1)
- void remove_hmat_pair_from_mm_product_list (const HMatrix< spacedim, Number > *M1, const HMatrix< spacedim, Number > *M2)
- void remove_hmat_pair_from_mm_product_list (const std::pair< const HMatrix< spacedim, Number > *, const HMatrix< spacedim, Number > *> &hmat_pair)
- TreeNodeSplitMode determine_mm_split_mode_from_Sigma_P ()

Private Member Functions

- template<typename MatrixType >
  void _convertToFullMatrix (MatrixType &M) const
- void _build_leaf_set (std::vector< HMatrix *> &total_leaf_set) const
- void distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves ()
- void **distribute_sigma_r_and_f_to_leaves** ()
- void _distribute_sigma_r_and_f_to_leaves (HMatrix< spacedim, Number > &starting_hmat)

Private Attributes

- HMatrixType type
- std::vector< HMatrix * > submatrices
- std::vector< HMatrix * > leaf_set
- RkMatrix< Number > * rkmatrix
- LAPACKFullMatrixExt< Number > * fullmatrix
- BlockClusterTree< spacedim, Number >::node_pointer_type bc_node
- std::vector< types::global_dof_index > * row_indices
- std::vector< types::global_dof_index > * col_indices
- std::map< types::global_dof_index, size_t > row_index_global_to_local_map
- std::map< types::global_dof_index, size_t > col_index_global_to_local_map
- size_type m
- size_type n
- BlockClusterTree< spacedim, Number > Tind
- std::vector< std::pair< HMatrix< spacedim, Number > *, HMatrix< spacedim, Number > * > > Sigma_P
- std::vector< RkMatrix< Number > * > Sigma_R
- std::vector< LAPACKFullMatrixExt< Number > * > Sigma_F

Friends

- template<int spacedim1, typename Number1 >
  void **InitHMatrixWrtBlockClusterNode** (HMatrix< spacedim1, Number1 > &hmat, typename Block←ClusterTree< spacedim1, Number1 >::node_const_pointer_type bc_node)
- template<int spacedim1, typename Number1 >
  void **InitHMatrixWrtBlockClusterNode** (HMatrix< spacedim1, Number1 > &hmat, typename Block←ClusterTree< spacedim1, Number1 >::node_const_pointer_type bc_node, const std::vector< std::pair< H←Matrix< spacedim1, Number1 > *, HMatrix< spacedim1, Number1 > *>> &Sigma_P)
- template<int spacedim1, typename Number1 >
  void **InitHMatrixWrtBlockClusterNode** (HMatrix< spacedim1, Number1 > &hmat, typename Block←ClusterTree< spacedim1, Number1 >::node_const_pointer_type bc_node, const std::pair< HMatrix< spacedim1, Number1 > *, HMatrix< spacedim1, Number1 > *> &hmat_pair)
- template<int spacedim1, typename Number1 >
  void **InitAndCreateHMatrixChildren** (HMatrix< spacedim1, Number1 > *hmat, typename BlockCluster←Tree< spacedim1, Number1 >::node_const_pointer_type bc_node, const unsigned int fixed_rank_k, bool is_build_index_set_global_to_local_map)
- template<int spacedim1, typename Number1 >
  void **InitAndCreateHMatrixChildren** (HMatrix< spacedim1, Number1 > *hmat, typename BlockCluster←Tree< spacedim1, Number1 >::node_const_pointer_type bc_node, const unsigned int fixed_rank_k, const LAPACKFullMatrixExt< Number1 > &M, bool is_build_index_set_global_to_local_map)
- template<int spacedim1, typename Number1 >
  void **InitAndCreateHMatrixChildren** (HMatrix< spacedim1, Number1 > *hmat, typename BlockCluster←Tree< spacedim1, Number1 >::node_const_pointer_type bc_node, const unsigned int fixed_rank_k, const LAPACKFullMatrixExt< Number1 > &M, const std::map< types::global_dof_index, size_t > &row_index←_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_←local_map_for_M, bool is_build_index_set_global_to_local_map)
- template<int spacedim1, typename Number1 >
  void **InitAndCreateHMatrixChildren** (HMatrix< spacedim1, Number1 > *hmat, typename BlockCluster←Tree< spacedim1, Number1 >::node_const_pointer_type bc_node, HMatrix< spacedim1, Number1 > &&H)
- template<int spacedim1, typename Number1 >
  void **RefineHMatrixWrtExtendedBlockClusterTree** (HMatrix< spacedim1, Number1 > *starting_hmat, HMatrix< spacedim1, Number1 > *current_hmat)
- template<int spacedim1, typename Number1 >
  void **convertHMatBlockToRkMatrix** (HMatrix< spacedim1, Number1 > *hmat_block, const unsigned int fixed_rank_k, const HMatrix< spacedim1, Number1 > *hmat_root_block, size_t *calling_counter, const std::string &output_file_base_name)
- void **build_index_set_global_to_local_map** (const std::vector< types::global_dof_index > &index_set_←as_local_to_global_map, std::map< types::global_dof_index, size_t > &global_to_local_map)
- template<int spacedim1, typename Number1 >
  void **h_rk_mmult** (HMatrix< spacedim1, Number1 > &M1, const RkMatrix< Number1 > &M2, RkMatrix< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **h_rk_mmult_for_h_h_mmult** (HMatrix< spacedim1, Number1 > *M1, const HMatrix< spacedim1, Number1 > *M2, HMatrix< spacedim1, Number1 > *M, bool is_M1M2_last_in_M_Sigma_P)
- template<int spacedim1, typename Number1 >
  void **rk_h_mmult** (const RkMatrix< Number1 > &M1, HMatrix< spacedim1, Number1 > &M2, RkMatrix< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **rk_h_mmult_for_h_h_mmult** (const HMatrix< spacedim1, Number1 > *M1, HMatrix< spacedim1, Number1 > *M2, HMatrix< spacedim1, Number1 > *M, bool is_M1M2_last_in_M_Sigma_P)
- template<int spacedim1, typename Number1 >
  void **h_f_mmult** (HMatrix< spacedim1, Number1 > &M1, const LAPACKFullMatrixExt< Number1 > &M2, LAPACKFullMatrixExt< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **h_f_mmult** (HMatrix< spacedim1, Number1 > &M1, const LAPACKFullMatrixExt< Number1 > &M2, RkMatrix< Number1 > &M)

- template<int spacedim1, typename Number1 >
  void **h_f_mmult_for_h_h_mmult** (HMatrix< spacedim1, Number1 > ∗M1, const HMatrix< spacedim1, Number1 > ∗M2, HMatrix< spacedim1, Number1 > ∗M, bool is_M1M2_last_in_M_Sigma_P)
- template<int spacedim1, typename Number1 >
  void **f_h_mmult** (const LAPACKFullMatrixExt< Number1 > &M1, HMatrix< spacedim1, Number1 > &M2, LAPACKFullMatrixExt< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **f_h_mmult** (const LAPACKFullMatrixExt< Number1 > &M1, HMatrix< spacedim1, Number1 > &M2, RkMatrix< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **f_h_mmult_for_h_h_mmult** (const HMatrix< spacedim1, Number1 > ∗M1, HMatrix< spacedim1, Number1 > ∗M2, HMatrix< spacedim1, Number1 > ∗M, bool is_M1M2_last_in_M_Sigma_P)
- template<int spacedim1, typename Number1 >
  void **h_h_mmult_phase1_recursion** (HMatrix< spacedim1, Number1 > ∗M, BlockClusterTree< spacedim1, Number1 > &Tind)
- template<int spacedim1, typename Number1 >
  void **h_h_mmult_phase2** (HMatrix< spacedim1, Number1 > &M, BlockClusterTree< spacedim1, Number1 > &target_bc_tree, const unsigned int fixed_rank)
- template<int spacedim1, typename Number1 >
  void **copy_hmatrix_node** (HMatrix< spacedim1, Number1 > &hmat_dst, const HMatrix< spacedim1, Number1 > &hmat_src)
- template<int spacedim1, typename Number1 >
  void **copy_hmatrix_node** (HMatrix< spacedim1, Number1 > &hmat_dst, HMatrix< spacedim1, Number1 > &&hmat_src)
- template<int spacedim1, typename Number1 >
  void **copy_hmatrix** (HMatrix< spacedim1, Number1 > &hmat_dst, const HMatrix< spacedim1, Number1 > &hmat_src)
- template<int spacedim1, typename Number1 >
  void **print_h_submatrix_accessor** (std::ostream &out, const std::string &name, const HMatrix< spacedim1, Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **print_h_h_submatrix_mmult_accessor** (std::ostream &out, const std::string &name1, const HMatrix< spacedim1, Number1 > &M1, const std::string &name2, const HMatrix< spacedim1, Number1 > &M2)

8.13.1 Member Typedef Documentation

8.13.1.1 size_type

```
template<int spacedim, typename Number = double>
using HMatrix< spacedim, Number >::size_type = std::make_unsigned<types::blas_int>::type
```

Declare the type for container size.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 HMatrix() [1/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix ( )
```

Default constructor.

8.13.2.2 HMatrix() [2/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            const BlockClusterTree< spacedim, Number > & bct,
            const unsigned int fixed_rank_k = 1 )
```

Construct the hierarchical structure without data from the root node of a BlockClusterTree.

References HMatrix< spacedim, Number >::build_leaf_set(), and BlockClusterTree< spacedim, Number >::get←_root().

8.13.2.3 HMatrix() [3/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            const unsigned int fixed_rank_k = 1 )
```

Construct the hierarchical structure without data from a TreeNode in a BlockClusterTree.

References HMatrix< spacedim, Number >::build_leaf_set().

8.13.2.4 HMatrix() [4/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            const BlockClusterTree< spacedim, Number > & bct,
            const LAPACKFullMatrixExt< Number > & M,
            const unsigned int fixed_rank_k = 1 )
```

Construct from the root node of a BlockClusterTree while copying the data of a global full matrix, which is created on the complete block cluster $I \times J$.

References HMatrix< spacedim, Number >::build_leaf_set(), and BlockClusterTree< spacedim, Number >::get←_root().

8.13.2.5 HMatrix() [5/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            const LAPACKFullMatrixExt< Number > & M,
            const unsigned int fixed_rank_k = 1 )
```

Construct from a TreeNode in a BlockClusterTree while copying the data of a global full matrix, which is created on the complete block cluster $I \times J$.

References HMatrix< spacedim, Number >::build_leaf_set().

8.13.2.6 HMatrix() [6/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            HMatrix< spacedim, Number > && H )
```

Construct from a `TreeNode` in a `BlockClusterTree` while moving the data from the leaf set of the $\mathcal{H}$-matrix `H`.

Parameters

bc_node	
H	

References HMatrix< spacedim, Number >::build_leaf_set().

8.13.2.7 HMatrix() [7/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            const BlockClusterTree< spacedim, Number > & bct,
            HMatrix< spacedim, Number > && H )
```

Construct from the root node of a BlockClusterTree while moving the data from the leaf set of the $\mathcal{H}$-matrix `H`.

Parameters

bct	
H	

References HMatrix< spacedim, Number >::build_leaf_set(), and BlockClusterTree< spacedim, Number >::get↩
_root().

8.13.2.8 HMatrix() [8/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            const HMatrix< spacedim, Number > & H )
```

Deep copy constructor.

Parameters

H	

References HMatrix< spacedim, Number >::build_leaf_set().

8.13.2.9 HMatrix() [9/9]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::HMatrix (
            HMatrix< spacedim, Number > && H )
```

Shallow copy constructor.

After the copy operation, the data in the source matrix `H` are transferred to the current $\mathcal{H}$-matrix node and `H` is cleared.

Parameters

H	

8.13.2.10 ∼HMatrix()

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::~HMatrix ( )
```

Destructor which releases the memory by recursion.

References HMatrix< spacedim, Number >::release().

8.13.3 Member Function Documentation

8.13.3.1 _build_leaf_set()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::_build_leaf_set (
            std::vector< HMatrix< spacedim, Number > *> & total_leaf_set ) const [private]
```

Collect $\mathcal{H}$-matrix nodes in the leaf set into a vector.

Parameters

total_leaf_set	

References FullMatrixType, HierarchicalMatrixType, RkMatrixType, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

Referenced by HMatrix< spacedim, Number >::build_leaf_set().

8.13.3.2 _convertToFullMatrix()

```
template<int spacedim, typename Number >
template<typename MatrixType >
void HMatrix< spacedim, Number >::_convertToFullMatrix (
            MatrixType & M ) const  [private]
```

Convert an HMatrix to a full matrix by recursion.

Parameters

matrix	

References HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrix↩Type, HierarchicalMatrixType, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

Referenced by HMatrix< spacedim, Number >::convertToFullMatrix().

8.13.3.3 _distribute_sigma_r_and_f_to_leaves()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves (
            HMatrix< spacedim, Number > & starting_hmat )  [private]
```

Restrict each rank-k matrix in the list `Sigma_R` of `starting_hmat` to the block as a full matrix.

Restrict each full matrix in the list `Sigma_F` of `starting_hmat` to the block as a full matrix.

Restrict each rank-k matrix in the list `Sigma_R` of `starting_hmat` to the block as a rank-k matrix.

Restrict each full matrix in the list `Sigma_F` of `starting_hmat` to the block as a rank-k matrix.

References HMatrix< spacedim, Number >::col_index_global_to_local_map, HMatrix< spacedim, Number >↩::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, RkMatrix< Number >::restrictToFull↩Matrix(), HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_index_↩global_to_local_map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, H↩Matrix< spacedim, Number >::Sigma_R, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

Referenced by HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves().

8.13.3.4 add() [1/2]

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::add (
            HMatrix< spacedim, Number > & C,
            const HMatrix< spacedim, Number > & B,
            const size_type fixed_rank_k ) const
```

Add the current HMatrix `A` with another HMatrix `B` into `C`, i.e. whole matrix addition instead of addition limited to a specific block, where `C` will be truncated to a fixed rank `fixed_rank`.

This algorithm is intrinsically recursive, i.e. the addition of parent HMatrices will perform the addition of each pair of child HMatrices corresponding to a same block cluster. Strictly speaking, this member function `add` is not a recursive function, because the class instance which calls `add` changes from parent to child HMatrix.

N.B.

1. The two operands should have the same partition.
2. The hierarchical structure of `C` should be pre-generated.

Parameters

C	
B	
fixed_rank	

Work flow

Recursively add each pair of submatrices.

Perform addition of full matrices.

Perform addition of rank-k matrices.

References HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, HierarchicalMatrixType, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by main(), and HMatrix< spacedim, Number >::mmult().

8.13.3.5 add() [2/2]

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::add (
            const HMatrix< spacedim, Number > & B,
            const size_type fixed_rank_k ) const
```

Add the HMatrix `B` into the current HMatrix `A`, i.e. whole matrix addition instead of addition limited to a specific block, where `C` will be truncated to a fixed rank `fixed_rank`.

This algorithm is intrinsically recursive, i.e. the addition of parent HMatrices will perform the addition of each pair of child HMatrices corresponding to a same block cluster. Strictly speaking, this member function `add` is not a recursive function, because the class instance which calls `add` changes from parent to child HMatrix.

N.B. The two operands should have the same partition.

Parameters

B	
fixed_rank	

Work flow

Recursively add each pair of submatrices.

Perform addition of full matrices.

Perform addition of rank-k matrices.

References HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, HierarchicalMatrixType, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

8.13.3.6 build_leaf_set()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::build_leaf_set ( )
```

Build the leaf set of the current $\mathcal{H}$-matrix node.

References HMatrix< spacedim, Number >::_build_leaf_set(), and HMatrix< spacedim, Number >::leaf_set.

Referenced by HMatrix< spacedim, Number >::convert_between_different_block_cluster_trees(), HMatrix< spacedim, Number >::HMatrix(), HMatrix< spacedim, Number >::mmult(), HMatrix< spacedim, Number >↩ ::operator=(), and HMatrix< spacedim, Number >::refine_to_supertree().

8.13.3.7 clear()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::clear ( )
```

Clear the whole $\mathcal{H}$-matrix hierarchy. Recursively clear submatrices.

Clear the current matrix node.

References HMatrix< spacedim, Number >::clear_hmat_node(), HierarchicalMatrixType, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

8.13.3.8 clear_hmat_node()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::clear_hmat_node ( )
```

Clear the current $\mathcal{H}$-matrix node.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_index_global_to_↩ local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, HMatrix< spacedim, Number >::leaf_set, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, H↩ Matrix< spacedim, Number >::rkmatrix, HMatrix< spacedim, Number >::row_index_global_to_local_map, H↩ Matrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, HMatrix< spacedim, Number >::submatrices, H↩ Matrix< spacedim, Number >::Tind, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by HMatrix< spacedim, Number >::clear().

8.13.3.9 coarsen_to_partition()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::coarsen_to_partition (
            const std::vector< typename BlockClusterTree< spacedim, Number >::node_pointer↩
_type > & partition,
            const unsigned int fixed_rank_k )
```

Coarsen the current $\mathcal{H}$-matrix so that it corresponds to the given partition. Each rank-k matrix in the hierarchical matrix structure will be truncated to `fixed_rank_k`.

This member function implements the operator $\mathcal{T}_{P' \leftarrow P}^{\mathcal{H} \leftarrow \mathcal{H}}$ for the case $T(I \times J, P') \subset T(I \times J, P)$ in (7.9) in Hackbusch's $\mathcal{H}$-matrix book. Because there is no internal check about this, users should ensure this set inclusion relationship.

Parameters

partition	
fixed_rank↩ _k	

N.B. The function call `find_pointer_data` here involves the comparison of two block cluster node, which internally compares the two block clusters, which further compares the contained tau node and sigma node pointers. Therefore, at the moment, the inner most comparison is shallow comparison.

The block cluster node associated with the current $\mathcal{H}$-matrix node belongs to the given `partition`. Then $\mathcal{T}_r^{\mathcal{R} \leftarrow \mathcal{H}}$ will be applied to this $\mathcal{H}$-matrix node.

When the block cluster node associated with the current $\mathcal{H}$-matrix node does not belong to the `partition`, recursively call this same member function of its each child.

References HMatrix< spacedim, Number >::bc_node, find_pointer_data(), and HMatrix< spacedim, Number >↩ ::submatrices.

Referenced by HMatrix< spacedim, Number >::coarsen_to_subtree(), and HMatrix< spacedim, Number >↩ ::convert_between_different_block_cluster_trees().

8.13.3.10 coarsen_to_subtree()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::coarsen_to_subtree (
            const BlockClusterTree< spacedim, Number > & subtree,
            const unsigned int fixed_rank_k )
```

Coarsen the current $\mathcal{H}$-matrix so that it corresponds to the partition determined by the `subtree`. Each rank-k matrix in the hierarchical matrix structure will be truncated to `fixed_rank_k`.

This member function implements the operator $\mathcal{T}_{P' \leftarrow P}^{\mathcal{H} \leftarrow \mathcal{H}}$ for the case $T(I \times J, P') \subset T(I \times J, P)$ in (7.9) in Hackbusch's $\mathcal{H}$-matrix book. Because there is no internal check about this, users should ensure that the given `subtree` is really a subtree of the block cluster tree associated with this $\mathcal{H}$-matrix hierarchy.

Parameters

subtree	
fixed_rank↩_k	

References HMatrix< spacedim, Number >::coarsen_to_partition(), and BlockClusterTree< spacedim, Number >::get_leaf_set().

Referenced by main().

8.13.3.11 convert_between_different_block_cluster_trees()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::convert_between_different_block_cluster_trees (
            BlockClusterTree< spacedim, Number > & bct1,
            BlockClusterTree< spacedim, Number > & bct2,
            const unsigned int fixed_rank_k2 = 1 )
```

Convert an $\mathcal{H}$-matrix between two different block cluster trees T and T', where $T := T(I \times J, P)$ and $T' := T'(I \times J, P')$. The two trees have incompatible partitions and do not contain each other. However, they are constructed on the same cluster trees $T(I)$ and $T(J)$. This enables us to make a **shallow** comparison of two block cluster nodes based on the pointer addresses related to the comprising clusters, which is useful for verify the equality of two block cluster nodes.

The procedures of this algorithm are as below. Assume the current $\mathcal{H}$-matrix to be converted is associated with the block cluster tree T.

1. Extend T to be finer than T', from which we get the new block cluster tree T''.
2. Refine the original $\mathcal{H}$-matrix with respect to the extended tree T''.
3. Get and keep a record of the leaf set of the block cluster tree T', which will be used for matrix coarsening in the last step.
4. Extend T' to the finer block cluster tree T'', from which we get $\tilde{T}'$.
5. Build a new $\mathcal{H}$-matrix with respect to $\tilde{T}'$ with the actual data migrated from the leaf nodes of the original $\mathcal{H}$-matrix.
6. Coarsen the new $\mathcal{H}$-matrix to the original partition of T'.
7. Delete the hierarchy of the original $\mathcal{H}$-matrix.
8. Assign the new $\mathcal{H}$-matrix object to the original $\mathcal{H}$-matrix object.

Parameters

bct1	The block cluster tree which is associated with the current $\mathcal{H}$-matrix.
bct2	The block cluster tree to which the current $\mathcal{H}$-matrix is to be converted.

Make a copy of the leaf set of the target block cluster tree, which will be used for the final coarsening.

Extend the block cluster tree associated with the current $\mathcal{H}$-matrix to the coarsest tree which is finer than the target block cluster tree. If the block cluster tree has really been extended, refine the $\mathcal{H}$-matrix to its extended block cluster tree.

Extend `bct2` to the finer partition obtained from `bct1` as above. N.B. Now the leaf set of `bct1` after refinement is the same as that of `bct2` after this extension.

Create a new $\mathcal{H}$-matrix with respect to the extended `bct2`, which accepts the data migrated from the leaf set of the current $\mathcal{H}$-matrix.

Note

- This hierarchical structure of the new $\mathcal{H}$-matrix is built with respect to the extended block cluster tree `bct2`.
- The shallow copy constructor cannot be used here because the new $\mathcal{H}$-matrix has a different block cluster tree structure from the current $\mathcal{H}$-matrix, even though they have the same partition after tree extension.

Coarsen the new $\mathcal{H}$-matrix to the original leaf set of `bct2`. Then rebuild its leaf set.

Move the new $\mathcal{H}$-matrix to the current $\mathcal{H}$-matrix by shallow assignment.

References HMatrix< spacedim, Number >::build_leaf_set(), HMatrix< spacedim, Number >::coarsen_to_↩ partition(), BlockClusterTree< spacedim, Number >::extend_finer_than_partition(), BlockClusterTree< spacedim, Number >::extend_to_finer_partition(), BlockClusterTree< spacedim, Number >::get_leaf_set(), and HMatrix< spacedim, Number >::refine_to_supertree().

Referenced by h_h_mmult_phase2(), and main().

8.13.3.12 convertToFullMatrix()

```
template<int spacedim, typename Number >
template<typename MatrixType >
void HMatrix< spacedim, Number >::convertToFullMatrix (
            MatrixType & M ) const
```

Convert an HMatrix to a full matrix by calling the internal recursive function.

Note This function only has the verification purpose. In reality, a large dense matrix cannot be saved as a full matrix.

Parameters

matrix	

References HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::m, and HMatrix< spacedim, Number >::n.

Referenced by main().

8.13.3.13 determine_mm_split_mode_from_Sigma_P()

```
template<int spacedim, typename Number >
TreeNodeSplitMode HMatrix< spacedim, Number >::determine_mm_split_mode_from_Sigma_P ( )
```

Check the consistency of the tree node split modes which are associated with the $\mathcal{H}$-matrix node pairs stored in the list Σ_P of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::Sigma_P.

Referenced by h_h_mmult_phase1_recursion().

8.13.3.14 distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves ( )
[private]
```

Only non-leaf $\mathcal{H}$-matrix nodes need to be processed.

Since the current $\mathcal{H}$-matrix node has children, its type should be `HierarchicalMatrixType`.

Distribute matrices in Σ_b^R and Σ_b^F of the current $\mathcal{H}$-matrix node to its leaves, which is also a recursive function call.

Distribute matrices in Σ_b^R and Σ_b^F of each child matrix of the current $\mathcal{H}$-matrix node to its leaves

References HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::col_index_global_to_local_map, HMatrix< spacedim, Number >::col_indices, HierarchicalMatrixType, HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_R, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

Referenced by h_h_mmult_phase2().

8.13.3.15 find_block_cluster_in_leaf_set() [1/2]

```
template<int spacedim, typename Number >
std::vector< HMatrix< spacedim, Number > * >::iterator HMatrix< spacedim, Number >::find_block_cluster_in_leaf_set (
            const BlockCluster< spacedim, Number > & block_cluster )
```

Find a block cluster in the leaf set of the current $\mathcal{H}$-matrix and returns the iterator of the corresponding $\mathcal{H}$-matrix node in the leaf set.

Parameters

block_cluster	

Returns

Perform a shallow comparison, i.e. compare by pointer address, of the block clusters.

Note The data held by those previously found leaf nodes of the source $\mathcal{H}$-matrix have already been migrated to the leaf nodes of the new $\mathcal{H}$-matrix, which will make the data fields in these leaf nodes being empty. Hence, we will bypass them.

References HMatrix< spacedim, Number >::leaf_set.

8.13.3.16 find_block_cluster_in_leaf_set() [2/2]

```
template<int spacedim, typename Number >
std::vector< HMatrix< spacedim, Number > * >::const_iterator HMatrix< spacedim, Number >↩
::find_block_cluster_in_leaf_set (
            const BlockCluster< spacedim, Number > & block_cluster ) const
```

Find a block cluster in the leaf set of the current $\mathcal{H}$-matrix and returns the iterator of the corresponding $\mathcal{H}$-matrix node in the leaf set (const version).

Parameters

block_cluster	

Returns

Perform a shallow comparison, i.e. compare by pointer address, of the block clusters.

Note The data held by those previously found leaf nodes of the source $\mathcal{H}$-matrix have already been migrated to the leaf nodes of the new $\mathcal{H}$-matrix, which will make the data fields in these leaf nodes being empty. Hence, we will bypass them.

References HMatrix< spacedim, Number >::leaf_set.

8.13.3.17 get_fullmatrix() [1/2]

```
template<int spacedim, typename Number >
LAPACKFullMatrixExt< Number > * HMatrix< spacedim, Number >::get_fullmatrix ( )
```

Get the pointer to the full matrix of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::fullmatrix.

Referenced by main().

8.13.3.18 get_fullmatrix() [2/2]

```
template<int spacedim, typename Number >
const LAPACKFullMatrixExt< Number > * HMatrix< spacedim, Number >::get_fullmatrix ( ) const
```

Get the pointer to the full matrix of the current $\mathcal{H}$-matrix node (const version).

Returns

References HMatrix< spacedim, Number >::fullmatrix.

8.13.3.19 get_leaf_set() [1/2]

```
template<int spacedim, typename Number >
std::vector< HMatrix< spacedim, Number > * > & HMatrix< spacedim, Number >::get_leaf_set ( )
```

Get the reference to the leaf set of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::leaf_set.

8.13.3.20 get_leaf_set() [2/2]

```
template<int spacedim, typename Number >
const std::vector< HMatrix< spacedim, Number > * > & HMatrix< spacedim, Number >::get_leaf↩
_set ( ) const
```

Get the reference to the leaf set of the current $\mathcal{H}$-matrix node (const version).

Returns

References HMatrix< spacedim, Number >::leaf_set.

8.13.3.21 get_m()

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::size_type HMatrix< spacedim, Number >::get_m ( ) const
```

Get the number of rows of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::m.

8.13.3.22 get_n()

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number >::size_type HMatrix< spacedim, Number >::get_n ( ) const
```

Get the number of columns of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::n.

8.13.3.23 get_rkmatrix() [1/2]

```
template<int spacedim, typename Number >
RkMatrix< Number > * HMatrix< spacedim, Number >::get_rkmatrix ( )
```

Get the pointer to the rank-k matrix of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::rkmatrix.

Referenced by main().

8.13.3.24 get_rkmatrix() [2/2]

```
template<int spacedim, typename Number >
const RkMatrix< Number > * HMatrix< spacedim, Number >::get_rkmatrix ( ) const
```

Get the pointer to the rank-k matrix of the current $\mathcal{H}$-matrix node (const version).

Returns

References HMatrix< spacedim, Number >::rkmatrix.

8.13.3.25 get_submatrices() [1/2]

```
template<int spacedim, typename Number >
std::vector< HMatrix< spacedim, Number > * > & HMatrix< spacedim, Number >::get_submatrices
( )
```

Get the reference to the vector of submatrices of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::submatrices.

8.13.3.26 get_submatrices() [2/2]

```
template<int spacedim, typename Number >
const std::vector< HMatrix< spacedim, Number > * > & HMatrix< spacedim, Number >::get_↩
submatrices ( ) const
```

Get the reference to the vector of submatrices of the current $\mathcal{H}$-matrix node (const version).

Returns

References HMatrix< spacedim, Number >::submatrices.

8.13.3.27 get_type()

```
template<int spacedim, typename Number >
HMatrixType HMatrix< spacedim, Number >::get_type ( ) const
```

Get the matrix type of the current $\mathcal{H}$-matrix node.

Returns

References HMatrix< spacedim, Number >::type.

Referenced by main().

8.13.3.28 h_h_mmult_cross_split()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::h_h_mmult_cross_split (
            BlockClusterTree< spacedim, Number > & bc_tree )
```

This function implements `MM_C` in Hackbusch's $\mathcal{H}$-matrix book. Split the block cluster b in T_{ind}.

Append the initialized child to the list of submatrices of `M`.

Append the initialized child to the list of submatrices of `hmat`.

Append the initialized child to the list of submatrices of `M`.

Append the initialized child to the list of submatrices of `hmat`.

Iterate over each multiplication subtask.

Create $\mathcal{H}$-matrices corresponding to the child block clusters after splitting.

$\Sigma_{b_s(1)}^P := \Sigma_{b_s(1)}^P \cup \{[\tilde{M}_1(1), \tilde{M}_2(1)], [\tilde{M}_1(2), \tilde{M}_2(3)]\}$

$\Sigma_{b_s(2)}^P := \Sigma_{b_s(2)}^P \cup \{[\tilde{M}_1(1), \tilde{M}_2(2)], [\tilde{M}_1(2), \tilde{M}_2(4)]\}$

$\Sigma_{b_s(3)}^P := \Sigma_{b_s(3)}^P \cup \{[\tilde{M}_1(3), \tilde{M}_2(1)], [\tilde{M}_1(4), \tilde{M}_2(3)]\}$

$\Sigma_{b_s(4)}^P := \Sigma_{b_s(4)}^P \cup \{[\tilde{M}_1(3), \tilde{M}_2(2)], [\tilde{M}_1(4), \tilde{M}_2(4)]\}$

Remove the current $\mathcal{H}$-matrix pair from the list `Sigma_P` of the current matrix node.

Remove the current $\mathcal{H}$-matrix pair from the list `Sigma_P` of the current matrix node.

Update the matrix type of the current $\mathcal{H}$-matrix.

References HMatrix< spacedim, Number >::bc_node, and split_block_cluster_node().

8.13.3.29 h_h_mmult_horizontal_split()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::h_h_mmult_horizontal_split (
            BlockClusterTree< spacedim, Number > & bc_tree )
```

This function implements `MM_H` in Hackbusch's $\mathcal{H}$-matrix book. Split the block cluster b in T_{ind}.

Append the initialized child to the list of submatrices of `M`.

Append the initialized child to the list of submatrices of `hmat`.

Iterate over each multiplication subtask.

Create $\mathcal{H}$-matrices corresponding to the child block clusters after splitting.

Remove the current $\mathcal{H}$-matrix pair from the list `Sigma_P` of the current matrix node.

Remove the current $\mathcal{H}$-matrix pair from the list `Sigma_P` of the current matrix node.

Update the matrix type of the current $\mathcal{H}$-matrix.

References HMatrix< spacedim, Number >::bc_node, and split_block_cluster_node().

8.13.3.30 h_h_mmult_reduction()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::h_h_mmult_reduction ( )
```

Perform $\mathcal{H}$-matrix MM multiplication reduction. This is (7.21) in Hackbusch's $\mathcal{H}$-matrix book. When one of the operands is either full matrix or rank-k matrix, perform direct multiplication.

Migrate the current H-matrix node pair to the list `Sigma_P_cannot_reduced`.

Remove the current H-matrix node pair from the original list in `M`.

Merge the elements in `Sigma_P_cannot_reduced` back to `Sigma_P` in `M` for further processing.

References HMatrix< spacedim, Number >::bc_node, FullMatrixType, RkMatrixType, HMatrix< spacedim, Number >::Sigma_P, and HMatrix< spacedim, Number >::type.

Referenced by h_h_mmult_phase1_recursion().

8.13.3.31 h_h_mmult_vertical_split()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::h_h_mmult_vertical_split (
            BlockClusterTree< spacedim, Number > & bc_tree )
```

This function implements `MM_V` in Hackbusch's $\mathcal{H}$-matrix book. Split the block cluster b in T_{ind}.

Append the initialized child to the list of submatrices of `M`.

Append the initialized child to the list of submatrices of `hmat`.

Iterate over each multiplication subtask.

Create $\mathcal{H}$-matrices corresponding to the child block clusters after splitting.

Remove the current $\mathcal{H}$-matrix pair from the list `Sigma_P` of the current matrix node.

Remove the current $\mathcal{H}$-matrix pair from the list `Sigma_P` of the current matrix node.

Update the matrix type of the current $\mathcal{H}$-matrix.

References HMatrix< spacedim, Number >::bc_node, and split_block_cluster_node().

8.13.3.32 mmult()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::mmult (
            HMatrix< spacedim, Number > & C,
            HMatrix< spacedim, Number > & B,
            BlockClusterTree< spacedim, Number > & bct_a,
            BlockClusterTree< spacedim, Number > & bct_b,
            BlockClusterTree< spacedim, Number > & bct_c,
            const unsigned int fixed_rank = 1 )
```

Multiplication of two $\mathcal{H}$-matrices $C = A \cdot B$.

Parameters

C	
B	
bct_a	
bct_b	
bct_c	
fixed_rank	

Work flow

- Release the resource of the result matrix.
- Initialize the induced block cluster tree T_{ind} for the result matrix with a single root node.
- Associate with the root node of the induced block cluster tree T_{ind}.
- Perform recursive multiplication while constructing the induced block cluster tree T_{ind}.

- After the construction of the induced block cluster tree T_{ind}, rebuild its leaf set as well as near field and far field sets, and update the tree depth and maximum level.
  DEBUG: Print the structure of the T_{ind} block cluster tree.
- Build the leaf set of the result matrix.
  DEBUG: Print the structure of the T_{ind} block cluster tree.

References HMatrix< spacedim, Number >::add(), HMatrix< spacedim, Number >::build_leaf_set(), Tree↩ Node< T, N >::get_data_reference(), BlockClusterTree< spacedim, Number >::get_n_min(), BlockClusterTree< spacedim, Number >::get_root(), HMatrix< spacedim, Number >::release(), and HMatrix< spacedim, Number >::Tind.

Referenced by main().

8.13.3.33 operator=() [1/2]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number > & HMatrix< spacedim, Number >::operator= (
            HMatrix< spacedim, Number > && H )
```

Assignment via shallow copy.

Parameters

H	

Returns

References HMatrix< spacedim, Number >::release().

8.13.3.34 operator=() [2/2]

```
template<int spacedim, typename Number >
HMatrix< spacedim, Number > & HMatrix< spacedim, Number >::operator= (
            const HMatrix< spacedim, Number > & H )
```

Assignment via deep copy.

Parameters

H	

Returns

References HMatrix< spacedim, Number >::build_leaf_set(), and HMatrix< spacedim, Number >::release().

8.13.3.35 print_formatted()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::print_formatted (
            std::ostream & out,
            const unsigned int precision = 3,
            const bool scientific = true,
            const unsigned int width = 0,
            const char * zero_string = " ",
            const double denominator = 1.,
            const double threshold = 0. )  const
```

Print the HMatrix.

Parameters

out	
precision	
scientific	
width	
zero_string	
denominator	
threshold	

References HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, HierarchicalMatrixType, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by main().

8.13.3.36 print_matrix_info()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::print_matrix_info (
            std::ostream & out ) const
```

Print the size of Σ_b^P, Σ_b^R and Σ_b^F.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, and HMatrix< spacedim, Number >::submatrices.

8.13.3.37 refine_to_supertree()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::refine_to_supertree ( )
```

Refine the current $\mathcal{H}$-matrix, whose associated block cluster tree has been extended. The operation has no accuracy loss.

This member function implements the operator $\mathcal{T}^{\mathcal{H}\leftarrow\mathcal{H}}_{P'\leftarrow P}$ for the case $T(I \times J, P') \supset T(I \times J, P)$ in (7.9) in Hackbusch's $\mathcal{H}$-matrix book. Because there is no internal check about this, users should ensure that the original block cluster tree associated with this $\mathcal{H}$-matrix hierarchy has really been extended.

Work flow Iterate over the leaf set of the $\mathcal{H}$-matrix hierarchy.

Refine from the current $\mathcal{H}$-matrix leaf node.

After the refinement operation, we check the number of child matrices of the current $\mathcal{H}$-matrix leaf node.

If the current $\mathcal{H}$-matrix leaf node has a non-empty collection of submatrices, it has really been refined. Then delete its originally associated matrix data, either a full matrix or a rank-k matrix, and modify its matrix type as `HierarchicalMatrixType`.

After the refinement operation for all the leaf nodes of the original $\mathcal{H}$-matrix hierarchy finishes, rebuild the leaf set of the new $\mathcal{H}$-matrix hierarchy.

References HMatrix< spacedim, Number >::build_leaf_set(), FullMatrixType, HierarchicalMatrixType, HMatrix< spacedim, Number >::leaf_set, and RkMatrixType.

Referenced by HMatrix< spacedim, Number >::convert_between_different_block_cluster_trees(), and main().

8.13.3.38 release()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::release ( )
```

TODO: Construct from a BlockClusterTree and a Sauter quadrature object/functor. Release the memory and status of the $\mathcal{H}$-matrix hierarchy. The deletion of `submatrix` will call the destructor of this sub-HMatrix, which will further recursively call the destructor of the submatrices of this sub-HMatrix. Hence, this destructor is intrinsically recursive.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_index_global_to_local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, HMatrix< spacedim, Number >::leaf_set, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::Tind, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by HMatrix< spacedim, Number >::mmult(), HMatrix< spacedim, Number >::operator=(), and HMatrix< spacedim, Number >::~HMatrix().

8.13.3.39 remove_hmat_pair_from_mm_product_list() [1/2]

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::remove_hmat_pair_from_mm_product_list (
            const HMatrix< spacedim, Number > * M1,
            const HMatrix< spacedim, Number > * M2 )
```

Remove a pair of $\mathcal{H}$-matrix nodes from the list of matrix-matrix product subtasks to be performed, i.e. from the list `HMatrix::Sigma_P`.

References HMatrix< spacedim, Number >::Sigma_P.

Referenced by f_h_mmult_for_h_h_mmult(), h_f_mmult_for_h_h_mmult(), h_rk_mmult_for_h_h_mmult(), and rk↩_h_mmult_for_h_h_mmult().

8.13.3.40 remove_hmat_pair_from_mm_product_list() [2/2]

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::remove_hmat_pair_from_mm_product_list (
            const std::pair< const HMatrix< spacedim, Number > *, const HMatrix< spacedim,
Number > *> & hmat_pair )
```

Remove a pair of $\mathcal{H}$-matrix nodes from the list of matrix-matrix product subtasks to be performed, i.e. from the list `HMatrix::Sigma_P`.

References HMatrix< spacedim, Number >::Sigma_P.

8.13.3.41 truncate_to_rank()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::truncate_to_rank (
            size_type new_rank )
```

Truncate all rank-k matrices in the leaf set of the $\mathcal{H}$-matrix to rank-k matrices with the given `new_rank`, while the full matrices in the leaf set, i.e. those near-field matrices, are kept intact.

Note This method implements the operator $\mathcal{T}^{\mathcal{H}}_{r \leftarrow s}$ in (7.5) in Hackbusch's $\mathcal{H}$-matrix book.

Parameters

new_rank	

Do nothing.

Truncate the RkMatrix in-place.

References FullMatrixType, HierarchicalMatrixType, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, H↩
Matrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by main().

8.13.3.42 Tvmult()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::Tvmult (
            Vector< Number > & y,
            const Vector< Number > & x ) const
```

Calculate matrix-vector multiplication as $y = y + M^T \cdot x$, i.e. the matrix M is transposed.

Because the matrix M is transposed, the roles for `row_indices` and `col_indices` should be swapped. Also refer to HMatrix::vmult.

Parameters

y	
x	

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

References HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrix↩
Type, HierarchicalMatrixType, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix<
spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim,
Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

8.13.3.43 Tvmult_local_vector()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::Tvmult_local_vector (
            Vector< Number > & y,
            const std::map< types::global_dof_index, size_t > & y_index_global_to_local_map,
            const Vector< Number > & x,
            const std::map< types::global_dof_index, size_t > & x_index_global_to_local_map )
const
```

Calculate matrix-vector multiplication as $y = y + M^T \cdot x$, i.e. the matrix M is transposed.

Because the matrix M is transposed, the roles for `row_indices` and `col_indices` should be swapped.

Note The input vectors `x` and `y` are to be accessed via local indices with the assistance of `row_index_`↩
`global_to_local_map` and `col_index_global_to_local_map`.

Also refer to HMatrix::vmult_local_vector.

Parameters

y	
x	

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

References HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrix↩Type, HierarchicalMatrixType, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by f_h_mmult(), and rk_h_mmult().

8.13.3.44 vmult()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::vmult (
            Vector< Number > & y,
            const Vector< Number > & x ) const
```

Calculate matrix-vector multiplication as $y = y + M \cdot x$.

Note 1. The recursive algorithm for $\mathcal{H}$-matrix-vector multiplication needs to collect the results from different components in the leaf set and corresponding vector block in x. More importantly, there will be a series of such results contributing to a same block in the result vector y. Therefore, if the interface of this function is designed with the parameter `add` as that in the `vmult` function of `LAPACKFullMatrix` in deal.ii, in all recursive calls of `vmult` except the first one, this `add` flag should be set to `true`, irrespective of the original flag value passed into the first call of `vmult`. Hence, we do not include the `add` flag in the `vmult` function.

1. The input vectors `x` and `y` are to be accessed via global DoF indices.

Parameters

y	
x	

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

References HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrix↩
Type, HierarchicalMatrixType, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

8.13.3.45 vmult_local_vector()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::vmult_local_vector (
            Vector< Number > & y,
            const std::map< types::global_dof_index, size_t > & y_index_global_to_local_map,
            const Vector< Number > & x,
            const std::map< types::global_dof_index, size_t > & x_index_global_to_local_map )
const
```

Calculate matrix-vector multiplication as $y = y + M \cdot x$.

Note 1. The recursive algorithm for $\mathcal{H}$-matrix-vector multiplication needs to collect the results from different components in the leaf set and corresponding vector block in x. More importantly, there will be a series of such results contributing to a same block in the result vector y. Therefore, if the interface of this function is designed with the parameter `add` as that in the `vmult` function of `LAPACKFullMatrix` in deal.ii, in all recursive calls of `vmult` except the first one, this `add` flag should be set to `true`, irrespective of the original flag value passed into the first call of `vmult`. Hence, we do not include the `add` flag in the `vmult` function.

1. The input vectors `x` and `y` are to be accessed via local indices with the assistance of `row_index_↩
global_to_local_map` and `col_index_global_to_local_map`.

Parameters

y	
x	

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

Restrict vector x to the current matrix block.

Merge back the result vector `local_y` to `y`.

References HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrix↩
Type, HierarchicalMatrixType, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by h_f_mmult(), and h_rk_mmult().

8.13.3.46 write_fullmatrix_leaf_node()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::write_fullmatrix_leaf_node (
            std::ostream & out,
            const Number singular_value_threshold = 0. ) const
```

Write formatted full matrix leaf node to the output stream.

The leaf node is written in the following format:

[list-of-indices-in-cluster-tau],[list-of-indices-in-cluster-sigma],is_near_field,rank

For example,

[1 2 3 ...],[7 8 9 ...],1,1

Parameters

out	
singular_value_threshold	

Print index set of cluster τ.

Print index set of cluster σ.

Print the `is_near_field` flag.

Make a copy of the matrix block and calculate its rank using SVD.

Print the `rank` flag.

References HMatrix< spacedim, Number >::bc_node, FullMatrixType, and HMatrix< spacedim, Number >::type.

Referenced by HMatrix< spacedim, Number >::write_leaf_set().

8.13.3.47 write_leaf_set()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::write_leaf_set (
            std::ostream & out,
            const Number singular_value_threshold = 0. ) const
```

Write formatted leaf set to the output stream as well as the rank of each matrix block by recursion.

Each leaf node is written in the following format:

[list-of-indices-in-cluster-tau],[list-of-indices-in-cluster-sigma],is_near_field,rank

For example,

[1 2 3 ...],[7 8 9 ...],1,1

Parameters

out	
singular_value_threshold	

References FullMatrixType, HierarchicalMatrixType, RkMatrixType, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, HMatrix< spacedim, Number >::write_fullmatrix_leaf_node(), and HMatrix< spacedim, Number >::write_rkmatrix_leaf_node().

Referenced by main().

8.13.3.48 write_leaf_set_by_iteration()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::write_leaf_set_by_iteration (
            std::ostream & out,
            const Number singular_value_threshold = 0. ) const
```

Write formatted leaf set to the output stream as well as the rank of each matrix block by iteration.

Each leaf node is written in the following format:

[list-of-indices-in-cluster-tau],[list-of-indices-in-cluster-sigma],is_near_field,rank

For example,

    [1 2 3 ...],[7 8 9 ...],1,1

Parameters

out	
singular_value_threshold	

References FullMatrixType, HMatrix< spacedim, Number >::leaf_set, RkMatrixType, and HMatrix< spacedim, Number >::type.

Referenced by main().

8.13.3.49 write_rkmatrix_leaf_node()

```
template<int spacedim, typename Number >
void HMatrix< spacedim, Number >::write_rkmatrix_leaf_node (
            std::ostream & out ) const
```

Write formatted rank-k matrix leaf node to the output stream.

The leaf node is written in the following format:

[list-of-indices-in-cluster-tau],[list-of-indices-in-cluster-sigma],is_near_field,rank

For example,

[1 2 3 ...],[7 8 9 ...],1,1

Parameters

out	

Print index set of cluster τ.

Print index set of cluster σ.

Print the `is_near_field` flag.

Print the `rank` flag.

References HMatrix< spacedim, Number >::bc_node, RkMatrixType, and HMatrix< spacedim, Number >::type.

Referenced by HMatrix< spacedim, Number >::write_leaf_set().

8.13.4 Member Data Documentation

8.13.4.1 bc_node

```
template<int spacedim, typename Number = double>
BlockClusterTree<spacedim, Number>::node_pointer_type HMatrix< spacedim, Number >::bc_node
[private]
```

Pointer to the corresponding block cluster node in a BlockClusterTree.

Referenced by HMatrix< spacedim, Number >::clear_hmat_node(), HMatrix< spacedim, Number >::coarsen_to_partition(), convertHMatBlockToRkMatrix(), copy_hmatrix_node(), f_h_mmult_for_h_h_mmult(), h_f_mmult_for_h_h_mmult(), HMatrix< spacedim, Number >::h_h_mmult_cross_split(), HMatrix< spacedim, Number >::h_h_mmult_horizontal_split(), HMatrix< spacedim, Number >::h_h_mmult_reduction(), HMatrix< spacedim, Number >::h_h_mmult_vertical_split(), InitAndCreateHMatrixChildren(), InitHMatrixWrtBlockClusterNode(), HMatrix< spacedim, Number >::print_matrix_info(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), HMatrix< spacedim, Number >::write_fullmatrix_leaf_node(), and HMatrix< spacedim, Number >::write_rkmatrix_leaf_node().

8.13.4.2 col_index_global_to_local_map

```
template<int spacedim, typename Number = double>
std::map<types::global_dof_index, size_t> HMatrix< spacedim, Number >::col_index_global_to_↩
local_map [private]
```

Map from local column indices to global column indices for the cluster σ. The set of local column indices is the range $[0, \#\sigma - 1]$. The corresponding set of global column indices is a subset of J.

Note This mapping is only constructed for H-matrices in the leaf set.

Referenced by HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::clear_hmat_node(), convertHMatBlockToRkMatrix(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), f_h_mmult(), h_f_mmult(), h_rk_mmult(), Init↩ AndCreateHMatrixChildren(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >↩ ::release(), and rk_h_mmult().

8.13.4.3 col_indices

```
template<int spacedim, typename Number = double>
std::vector<types::global_dof_index>* HMatrix< spacedim, Number >::col_indices [private]
```

Pointer to the vector of global column indices, which is stored as the index set in the cluster σ. It is a subset of J. By accessing this vector using indices starting from 0, we actually obtain the mapping from the current matrix's local column indices to the global column indices.

Referenced by HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::_↩ distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::clear_hmat_node(), convertHMatBlock↩ ToRkMatrix(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_↩ r_and_f_to_leaves(), f_h_mmult(), f_h_mmult_for_h_h_mmult(), h_f_mmult(), h_f_mmult_for_h_h_mmult(), h↩ _rk_mmult(), h_rk_mmult_for_h_h_mmult(), InitAndCreateHMatrixChildren(), InitHMatrixWrtBlockClusterNode(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), rk_h_mmult(), rk_h↩ _mmult_for_h_h_mmult(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_↩ local_vector(), HMatrix< spacedim, Number >::vmult(), and HMatrix< spacedim, Number >::vmult_local_vector().

8.13.4.4 fullmatrix

```
template<int spacedim, typename Number = double>
LAPACKFullMatrixExt<Number>* HMatrix< spacedim, Number >::fullmatrix [private]
```

Pointer to the full matrix. It is not null when the current HMatrix object belongs to the near field.

Referenced by HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::↩ _distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::add(), HMatrix< spacedim, Number >::clear_hmat_node(), copy_hmatrix_node(), f_h_mmult_for_h_h_mmult(), HMatrix< spacedim, Number >↩ ::get_fullmatrix(), h_f_mmult_for_h_h_mmult(), h_h_mmult_phase2(), InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::print_formatted(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_local_vector(), HMatrix< spacedim, Number >::vmult(), and HMatrix< spacedim, Number >::vmult_local_vector().

8.13.4.5 leaf_set

```
template<int spacedim, typename Number = double>
std::vector<HMatrix *> HMatrix< spacedim, Number >::leaf_set [private]
```

A list of submatrices in the leaf set.

Referenced by HMatrix< spacedim, Number >::build_leaf_set(), HMatrix< spacedim, Number >::clear_hmat←_node(), copy_hmatrix_node(), HMatrix< spacedim, Number >::find_block_cluster_in_leaf_set(), HMatrix< spacedim, Number >::get_leaf_set(), h_h_mmult_phase2(), HMatrix< spacedim, Number >::refine_to_←supertree(), HMatrix< spacedim, Number >::release(), and HMatrix< spacedim, Number >::write_leaf_set←_by_iteration().

8.13.4.6 m

```
template<int spacedim, typename Number = double>
size_type HMatrix< spacedim, Number >::m [private]
```

Total number of rows in the matrix.

Referenced by HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::clear_←hmat_node(), HMatrix< spacedim, Number >::convertToFullMatrix(), copy_hmatrix_node(), f_h_mmult(), HMatrix< spacedim, Number >::get_m(), h_f_mmult(), h_rk_mmult(), InitAndCreateHMatrixChildren(), InitHMatrixWrtBlock←ClusterNode(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), rk_h←_mmult(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_local_vector(), H←Matrix< spacedim, Number >::vmult(), and HMatrix< spacedim, Number >::vmult_local_vector().

8.13.4.7 n

```
template<int spacedim, typename Number = double>
size_type HMatrix< spacedim, Number >::n [private]
```

Total number of columns in the matrix.

Referenced by HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::clear_←hmat_node(), HMatrix< spacedim, Number >::convertToFullMatrix(), copy_hmatrix_node(), f_h_mmult(), HMatrix< spacedim, Number >::get_n(), h_f_mmult(), h_rk_mmult(), InitAndCreateHMatrixChildren(), InitHMatrixWrtBlock←ClusterNode(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), rk_h←_mmult(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_local_vector(), H←Matrix< spacedim, Number >::vmult(), and HMatrix< spacedim, Number >::vmult_local_vector().

8.13.4.8 rkmatrix

```
template<int spacedim, typename Number = double>
RkMatrix<Number>* HMatrix< spacedim, Number >::rkmatrix [private]
```

Pointer to the rank-k matrix. It is not null when the current HMatrix object belongs to the far field.

Referenced by HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::add(), HMatrix< spacedim, Number >::clear_hmat_node(), convertHMatBlockToRkMatrix(), copy_hmatrix_node(), HMatrix< spacedim, Number >::get_rkmatrix(), h_h_mmult_phase2(), h_rk_mmult_for_h_h_mmult(), InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::print_formatted(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), rk_h_mmult_for_h_h_mmult(), HMatrix< spacedim, Number >::truncate_to_rank(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_local_vector(), HMatrix< spacedim, Number >::vmult(), and HMatrix< spacedim, Number >::vmult_local_vector().

8.13.4.9 row_index_global_to_local_map

```
template<int spacedim, typename Number = double>
std::map<types::global_dof_index, size_t> HMatrix< spacedim, Number >::row_index_global_to_local_map [private]
```

Map from local row indices to global row indices for the cluster τ. The set of local row indices is the range $[0, \#\tau - 1]$. The corresponding set of global row indices is a subset of I.

Note This mapping is only constructed for H-matrices in the leaf set.

Referenced by HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::clear_hmat_node(), convertHMatBlockToRkMatrix(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), f_h_mmult(), h_f_mmult(), h_rk_mmult(), InitAndCreateHMatrixChildren(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), and rk_h_mmult().

8.13.4.10 row_indices

```
template<int spacedim, typename Number = double>
std::vector<types::global_dof_index>* HMatrix< spacedim, Number >::row_indices [private]
```

Pointer to the vector of global row indices, which is stored as the index in the cluster τ. It is a subset of I. By accessing this vector using indices starting from 0, we actually obtain the mapping from the current matrix's local row indices to the global row indices.

Referenced by HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::clear_hmat_node(), convertHMatBlockToRkMatrix(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), f_h_mmult(), f_h_mmult_for_h_h_mmult(), h_f_mmult(), h_f_mmult_for_h_h_mmult(), h_rk_mmult(), h_rk_mmult_for_h_h_mmult(), InitAndCreateHMatrixChildren(), InitHMatrixWrtBlockClusterNode(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), rk_h_mmult(), rk_h_mmult_for_h_h_mmult(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_local_vector(), HMatrix< spacedim, Number >::vmult(), and HMatrix< spacedim, Number >::vmult_local_vector().

8.13.4.11 Sigma_F

```
template<int spacedim, typename Number = double>
std::vector<LAPACKFullMatrixExt<Number> *> HMatrix< spacedim, Number >::Sigma_F  [private]
```

List of full matrix pointers used in $\mathcal{H}$-matrix multiplication.

Referenced by HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::clear_hmat_node(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), f_h_mmult_for_h_h_mmult(), h_f_mmult_for_h_h_mmult(), h_h_mmult_phase2(), h_rk_mmult_for_h_h_mmult(), InitHMatrixWrtBlockClusterNode(), HMatrix< spacedim, Number >::print_matrix_info(), HMatrix< spacedim, Number >::release(), and rk_h_mmult_for_h_h_mmult().

8.13.4.12 Sigma_P

```
template<int spacedim, typename Number = double>
std::vector< std::pair<HMatrix<spacedim, Number> *, HMatrix<spacedim, Number> *> > HMatrix<
spacedim, Number >::Sigma_P  [private]
```

List of pairs of pointers to $\mathcal{H}$-matrix nodes for multiplication.

Referenced by HMatrix< spacedim, Number >::clear_hmat_node(), copy_hmatrix_node(), HMatrix< spacedim, Number >::determine_mm_split_mode_from_Sigma_P(), f_h_mmult_for_h_h_mmult(), h_f_mmult_for_h_h_mmult(), h_h_mmult_phase1_recursion(), h_h_mmult_phase2(), HMatrix< spacedim, Number >::h_h_mmult_reduction(), h_rk_mmult_for_h_h_mmult(), InitHMatrixWrtBlockClusterNode(), HMatrix< spacedim, Number >::print_matrix_info(), HMatrix< spacedim, Number >::release(), HMatrix< spacedim, Number >::remove_hmat_pair_from_mm_product_list(), and rk_h_mmult_for_h_h_mmult().

8.13.4.13 Sigma_R

```
template<int spacedim, typename Number = double>
std::vector<RkMatrix<Number> *> HMatrix< spacedim, Number >::Sigma_R  [private]
```

List of rank-k matrix pointers used in $\mathcal{H}$-matrix multiplication.

Referenced by HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::clear_hmat_node(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), f_h_mmult_for_h_h_mmult(), h_f_mmult_for_h_h_mmult(), h_h_mmult_phase2(), h_rk_mmult_for_h_h_mmult(), InitHMatrixWrtBlockClusterNode(), HMatrix< spacedim, Number >::print_matrix_info(), HMatrix< spacedim, Number >::release(), and rk_h_mmult_for_h_h_mmult().

8.13.4.14 submatrices

```
template<int spacedim, typename Number = double>
std::vector<HMatrix *> HMatrix< spacedim, Number >::submatrices [private]
```

A list of submatrices of type HMatrix.

Referenced by HMatrix< spacedim, Number >::_build_leaf_set(), HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::add(), HMatrix< spacedim, Number >::clear(), HMatrix< spacedim, Number >::clear_hmat_node(), HMatrix< spacedim, Number >::coarsen_to_partition(), convertHMatBlockToRkMatrix(), copy_hmatrix(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::get_submatrices(), InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::print_formatted(), HMatrix< spacedim, Number >::print_matrix_info(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), HMatrix< spacedim, Number >::truncate_to_rank(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_local_vector(), HMatrix< spacedim, Number >::vmult(), HMatrix< spacedim, Number >::vmult_local_vector(), and HMatrix< spacedim, Number >::write_leaf_set().

8.13.4.15 Tind

```
template<int spacedim, typename Number = double>
BlockClusterTree<spacedim, Number> HMatrix< spacedim, Number >::Tind [private]
```

Block cluster tree when this matrix is the product of two $\mathcal{H}$-matrices.

Referenced by HMatrix< spacedim, Number >::clear_hmat_node(), copy_hmatrix_node(), h_h_mmult_phase2(), HMatrix< spacedim, Number >::mmult(), and HMatrix< spacedim, Number >::release().

8.13.4.16 type

```
template<int spacedim, typename Number = double>
HMatrixType HMatrix< spacedim, Number >::type [private]
```

Matrix type.

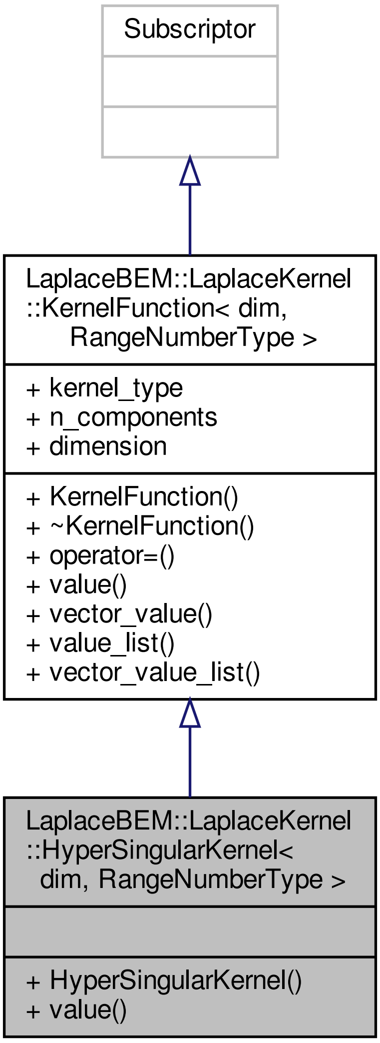
Referenced by HMatrix< spacedim, Number >::_build_leaf_set(), HMatrix< spacedim, Number >::_convertToFullMatrix(), HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::add(), HMatrix< spacedim, Number >::clear(), HMatrix< spacedim, Number >::clear_hmat_node(), convertHMatBlockToRkMatrix(), copy_hmatrix_node(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), f_h_mmult_for_h_h_mmult(), HMatrix< spacedim, Number >::get_type(), h_f_mmult_for_h_h_mmult(), h_h_mmult_phase2(), HMatrix< spacedim, Number >::h_h_mmult_reduction(), h_rk_mmult_for_h_h_mmult(), InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::print_formatted(), RefineHMatrixWrtExtendedBlockClusterTree(), HMatrix< spacedim, Number >::release(), rk_h_mmult_for_h_h_mmult(), HMatrix< spacedim, Number >::truncate_to_rank(), HMatrix< spacedim, Number >::Tvmult(), HMatrix< spacedim, Number >::Tvmult_local_vector(), HMatrix< spacedim, Number >::vmult(), HMatrix< spacedim, Number >::vmult_local_vector(), HMatrix< spacedim, Number >::write_fullmatrix_leaf_node(), HMatrix< spacedim, Number >::write_leaf_set(), HMatrix< spacedim, Number >::write_leaf_set_by_iteration(), and HMatrix< spacedim, Number >::write_rkmatrix_leaf_node().

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

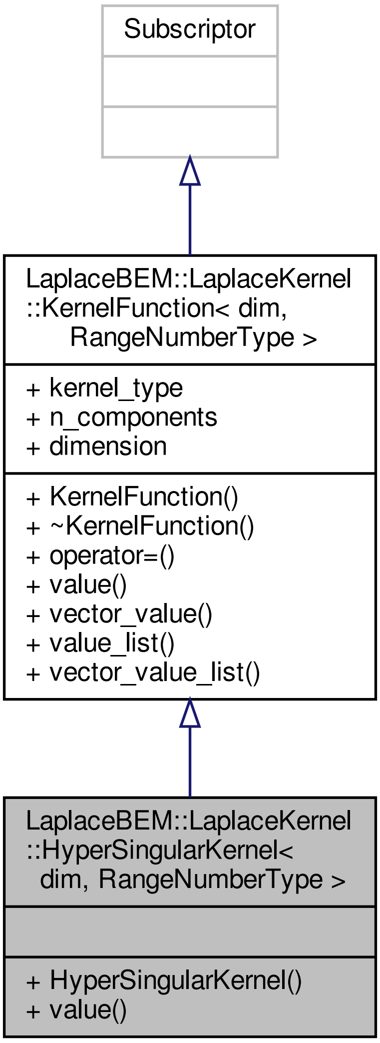
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/hmatrix.h

8.14 LaplaceBEM::LaplaceKernel::HyperSingularKernel< dim, RangeNumberType > Class Template Reference

Inheritance diagram for LaplaceBEM::LaplaceKernel::HyperSingularKernel< dim, RangeNumberType >:



Collaboration diagram for LaplaceBEM::LaplaceKernel::HyperSingularKernel< dim, RangeNumberType >:



Public Member Functions

- virtual RangeNumberType **value** (const Point< dim > &x, const Point< dim > &y, const Tensor< 1, dim > &nx, const Tensor< 1, dim > &ny, const unsigned int component=0) const override

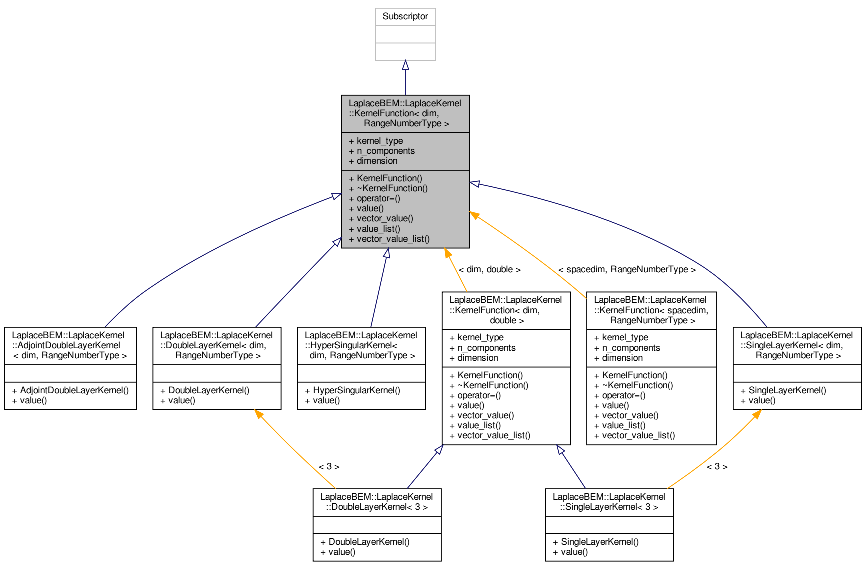
Additional Inherited Members

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

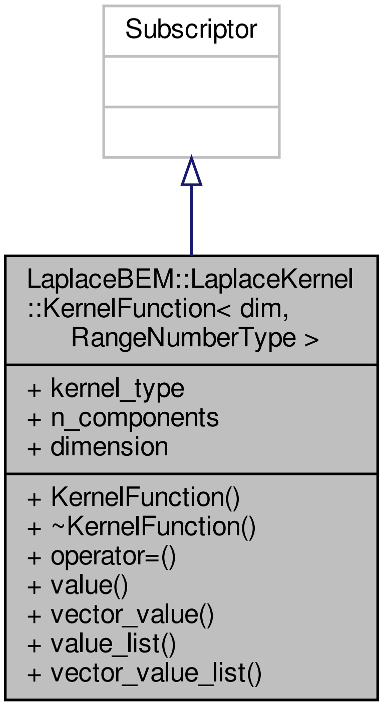
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.15 LaplaceBEM::LaplaceKernel::KernelFunction< dim, RangeNumberType > Class Template Reference

Inheritance diagram for LaplaceBEM::LaplaceKernel::KernelFunction< dim, RangeNumberType >:



Collaboration diagram for LaplaceBEM::LaplaceKernel::KernelFunction< dim, RangeNumberType >:



Public Member Functions

- **KernelFunction** (const KernelType kernel_type=NoneType, const unsigned int n_components=1)
- KernelFunction & **operator=** (const KernelFunction &f)
- virtual RangeNumberType **value** (const Point< dim > &x, const Point< dim > &y, const Tensor< 1, dim > &nx, const Tensor< 1, dim > &ny, const unsigned int component=0) const
- virtual void **vector_value** (const Point< dim > &x, const Point< dim > &y, const Tensor< 1, dim > &nx, const Tensor< 1, dim > &ny, Vector< RangeNumberType > &values) const
- virtual void **value_list** (const std::vector< Point< dim >> &x_points, const std::vector< Point< dim >> &y_points, const std::vector< Tensor< 1, dim >> &nx_list, const std::vector< Tensor< 1, dim >> &ny_list, std::vector< RangeNumberType > &values, const unsigned int component=0) const
- virtual void **vector_value_list** (const std::vector< Point< dim >> &x_points, const std::vector< Point< dim >> &y_points, const std::vector< Tensor< 1, dim >> &nx_list, const std::vector< Tensor< 1, dim >> &ny_list, std::vector< Vector< RangeNumberType >> &values) const

Public Attributes

- const KernelType **kernel_type**
- const unsigned int **n_components**

Static Public Attributes

- static const unsigned int **dimension** = dim

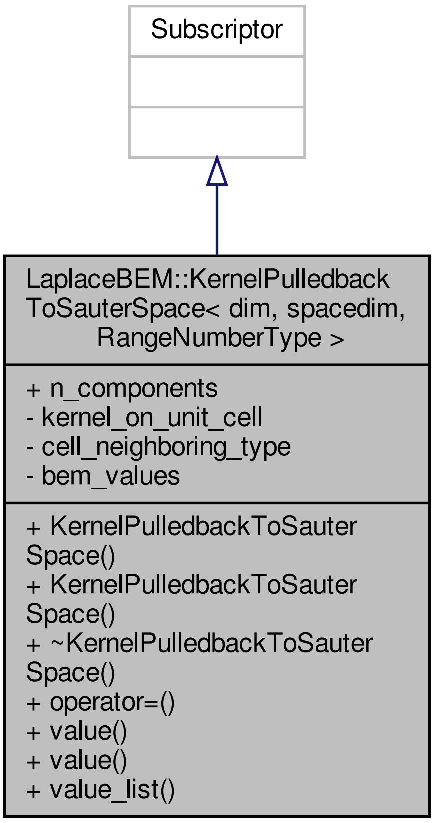
The documentation for this class was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

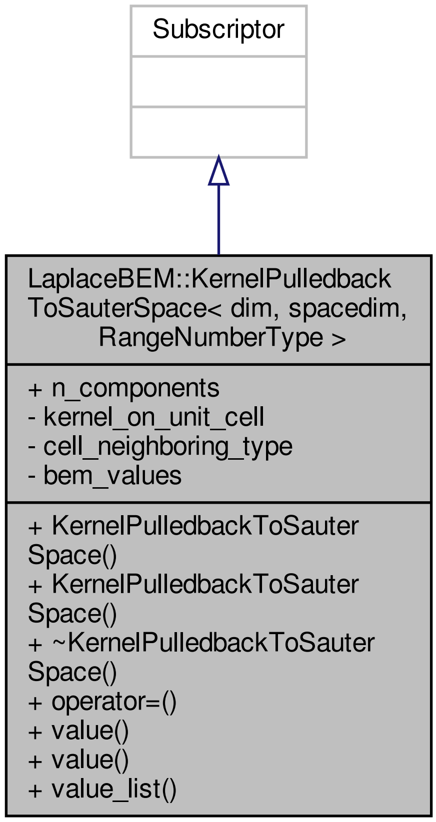
8.16 LaplaceBEM::KernelPulledbackToSauterSpace< dim, spacedim, RangeNumber↩ Type > Class Template Reference

`#include <laplace_bem.h>`

Inheritance diagram for LaplaceBEM::KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType >:



Collaboration diagram for LaplaceBEM::KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType >:



Public Member Functions

- **KernelPulledbackToSauterSpace** (const KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType > &kernel, const CellNeighboringType cell_neighboring_type)
- **KernelPulledbackToSauterSpace** (const KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType > &kernel, const CellNeighboringType cell_neighboring_type, const BEMValues< dim, spacedim, RangeNumberType > *bem_values)
- KernelPulledbackToSauterSpace & **operator=** (const KernelPulledbackToSauterSpace &f)
- RangeNumberType value (const Point< dim *2 > p, const unsigned int component=0) const
- RangeNumberType value (const unsigned int quad_no, const unsigned int component=0) const
- void **value_list** (const std::vector< Point< dim *2 >> &points, std::vector< RangeNumberType > &values, const unsigned int component=0) const

Public Attributes

- const unsigned int **n_components**

Private Attributes

- const KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType > & **kernel_on_unit_cell**
- CellNeighboringType **cell_neighboring_type**
- const BEMValues< dim, spacedim, RangeNumberType > * **bem_values**

8.16.1 Detailed Description

template<int dim, int spacedim, typename RangeNumberType = double>
class LaplaceBEM::KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType >

Class for pullback the kernel function on the product of two unit cells to Sauter's parametric space.

8.16.2 Member Function Documentation

8.16.2.1 value() [1/2]

```
template<int dim, int spacedim, typename RangeNumberType >
RangeNumberType LaplaceBEM::KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType >↩
::value (
            const Point< dim *2 > p,
            const unsigned int component = 0 ) const
```

Evaluate the pullback of kernel function on Sauter's parametric space.

Parameters

p	The coordinates at which the kernel function is to be evaluated. It should be noted that this point has a dimension of dim*2.

8.16.2.2 value() [2/2]

```
template<int dim, int spacedim, typename RangeNumberType >
RangeNumberType LaplaceBEM::KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType >↩
::value (
            const unsigned int quad_no,
            const unsigned int component = 0 ) const
```

Evaluate the pullback of kernel function on Sauter's parametric space at the quad_no'th quadrature point under the given 4D quadrature rule.

Parameters

quad_no	quadrature point index
component	

Returns

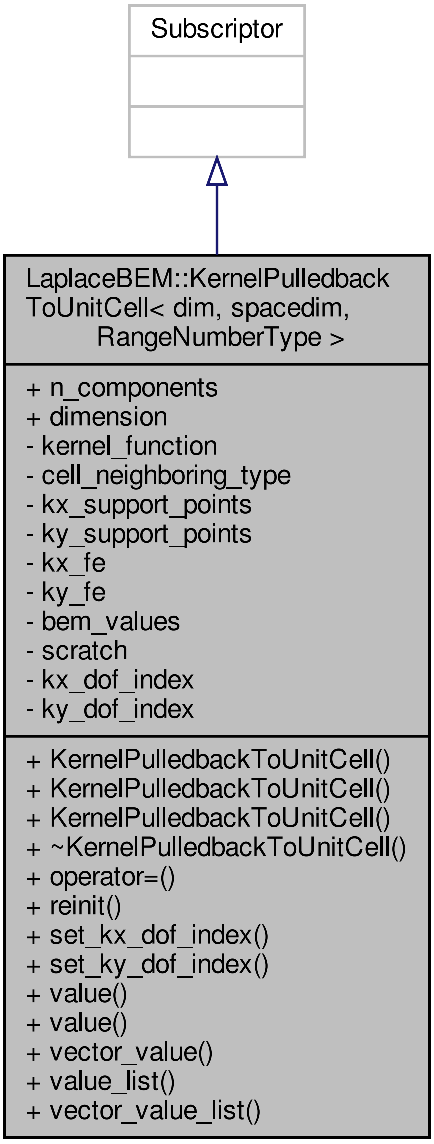
The documentation for this class was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

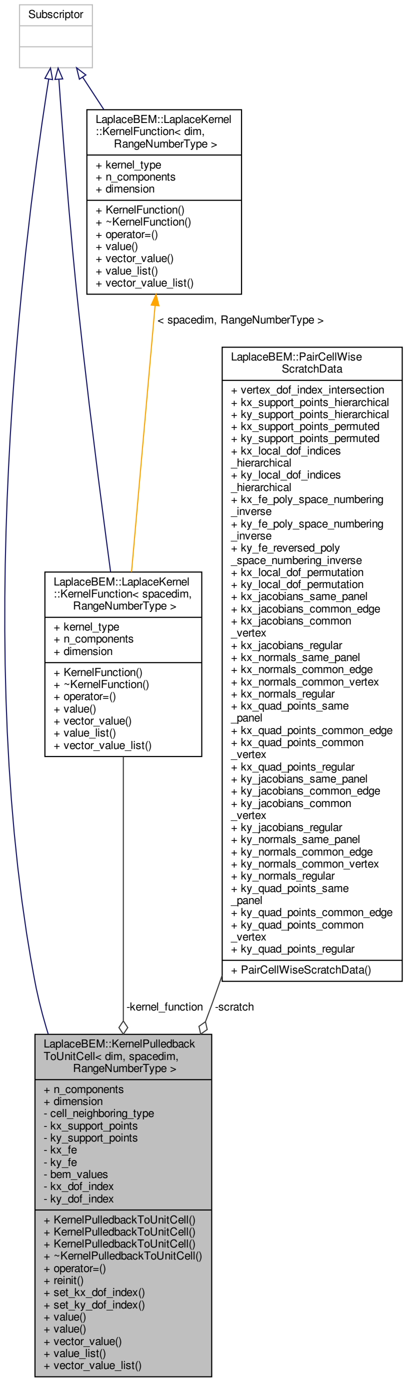
8.17 LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType > Class Template Reference

```
#include <laplace_bem.h>
```

Inheritance diagram for LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >:



Collaboration diagram for LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >:



Public Member Functions

- KernelPulledbackToUnitCell (const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > &kernel_function, const CellNeighboringType &cell_neighboring_type, const std::vector< Point< spacedim >> &kx_support_points, const std::vector< Point< spacedim >> &ky_support_points, const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const unsigned int kx_dof_index=0, const unsigned int ky_dof_index=0)

- KernelPulledbackToUnitCell (const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > &kernel_function, const CellNeighboringType &cell_neighboring_type, const std::vector< Point< spacedim >> &kx_support_points, const std::vector< Point< spacedim >> &ky_support_points, const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const BEMValues< dim, spacedim, RangeNumberType > *bem_values, const unsigned int kx_dof_index=0, const unsigned int ky_dof_index=0)
- KernelPulledbackToUnitCell (const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > &kernel_function, const CellNeighboringType &cell_neighboring_type, const std::vector< Point< spacedim >> &kx_support_points, const std::vector< Point< spacedim >> &ky_support_points, const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const BEMValues< dim, spacedim, RangeNumberType > *bem_values, const PairCellWiseScratchData *scratch, const unsigned int kx_dof_index=0, const unsigned int ky_dof_index=0)
- KernelPulledbackToUnitCell & **operator=** (const KernelPulledbackToUnitCell &f)
- void reinit (const CellNeighboringType &cell_neighboring_type, const std::vector< Point< spacedim >> &kx_support_points, const std::vector< Point< spacedim >> &ky_support_points, const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe)
- void **set_kx_dof_index** (const unsigned int kx_dof_index)
- void **set_ky_dof_index** (const unsigned int ky_dof_index)
- virtual RangeNumberType value (const Point< dim > &x_hat, const Point< dim > &y_hat, const unsigned int component=0) const
- virtual RangeNumberType **value** (const unsigned int k3_index, const unsigned int quad_no, const unsigned int component=0) const
- virtual void **vector_value** (const Point< dim > &x_hat, const Point< dim > &y_hat, Vector< RangeNumberType > &values) const
- virtual void **value_list** (const std::vector< Point< dim >> &x_hat_list, const std::vector< Point< dim >> &y_hat_list, std::vector< RangeNumberType > &values, const unsigned int component=0) const
- virtual void **vector_value_list** (const std::vector< Point< dim >> &x_hat_list, const std::vector< Point< dim >> &y_hat_list, std::vector< Vector< RangeNumberType >> &values) const

Public Attributes

- const unsigned int **n_components**

Static Public Attributes

- static const unsigned int **dimension** = dim

Private Attributes

- const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > & **kernel_function**
- CellNeighboringType **cell_neighboring_type**
- const std::vector< Point< spacedim > > & **kx_support_points**
- const std::vector< Point< spacedim > > & **ky_support_points**
- const FiniteElement< dim, spacedim > & **kx_fe**
- const FiniteElement< dim, spacedim > & **ky_fe**
- const BEMValues< dim, spacedim, RangeNumberType > * **bem_values**
- const PairCellWiseScratchData * **scratch**
- unsigned int kx_dof_index
- unsigned int ky_dof_index

8.17.1 Detailed Description

template<int dim, int spacedim, typename RangeNumberType = double>
class LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >

Kernel function pulled back to unit cell.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 KernelPulledbackToUnitCell() [1/3]

```
template<int dim, int spacedim, typename RangeNumberType >
LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::KernelPulledbackTo↩
UnitCell (
            const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > & kernel_↩
function,
            const CellNeighboringType & cell_neighboring_type,
            const std::vector< Point< spacedim >> & kx_support_points,
            const std::vector< Point< spacedim >> & ky_support_points,
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const unsigned int kx_dof_index = 0,
            const unsigned int ky_dof_index = 0 )
```

Constructor for KernelPulledbackToUnitCell.

Parameters

kx_support_points	Permuted list of support points in K_x.
ky_support_points	Permuted list of support points in K_y.
kx_dof_index	Index for accessing the list of DoFs in tensor product order in K_x.
ky_dof_index	Index for accessing the list of DoFs in tensor product order in K_y.

Referenced by LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::Kernel↩
PulledbackToUnitCell().

8.17.2.2 KernelPulledbackToUnitCell() [2/3]

```
template<int dim, int spacedim, typename RangeNumberType >
LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::KernelPulledbackTo↩
UnitCell (
            const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > & kernel_↩
function,
            const CellNeighboringType & cell_neighboring_type,
            const std::vector< Point< spacedim >> & kx_support_points,
```

```
            const std::vector< Point< spacedim >> & ky_support_points,
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const BEMValues< dim, spacedim, RangeNumberType > * bem_values,
            const unsigned int kx_dof_index = 0,
            const unsigned int ky_dof_index = 0 )
```

Parameters

kernel_function	
cell_neighboring_type	
kx_support_points	
ky_support_points	
kx_fe	
ky_fe	
bem_values	
kx_dof_index	
ky_dof_index	

References LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::Kernel↩
PulledbackToUnitCell().

8.17.2.3 KernelPulledbackToUnitCell() [3/3]

```
template<int dim, int spacedim, typename RangeNumberType >
LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::KernelPulledbackTo↩
UnitCell (
            const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > & kernel_↩
function,
            const CellNeighboringType & cell_neighboring_type,
            const std::vector< Point< spacedim >> & kx_support_points,
            const std::vector< Point< spacedim >> & ky_support_points,
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const BEMValues< dim, spacedim, RangeNumberType > * bem_values,
            const PairCellWiseScratchData * scratch,
            const unsigned int kx_dof_index = 0,
            const unsigned int ky_dof_index = 0 )
```

Parameters

kernel_function	
cell_neighboring_type	
kx_support_points	
ky_support_points	
kx_fe	
ky_fe	
bem_values	
kx_dof_index	
ky_dof_index	

8.17.3 Member Function Documentation

8.17.3.1 reinit()

```
template<int dim, int spacedim, typename RangeNumberType >
void LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::reinit (
            const CellNeighboringType & cell_neighboring_type,
            const std::vector< Point< spacedim >> & kx_support_points,
            const std::vector< Point< spacedim >> & ky_support_points,
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe )
```

Associate the KernelPulledbackToUnitCell with new support point and finite element data.

References LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::kx_dof_index, and LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::ky_dof_index.

8.17.3.2 value()

```
template<int dim, int spacedim, typename RangeNumberType >
RangeNumberType LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >↩
::value (
            const Point< dim > & x_hat,
            const Point< dim > & y_hat,
            const unsigned int component = 0 ) const  [virtual]
```

Evaluation of the function which depends on the kernel type. Different types of kernel function require different normal vector data.

References LaplaceBEM::transform_unit_to_permuted_real_cell().

8.17.4 Member Data Documentation

8.17.4.1 kx_dof_index

```
template<int dim, int spacedim, typename RangeNumberType = double>
unsigned int LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::kx_↩
dof_index  [private]
```

Index for accessing the list of DoFs in tensor product order in K_x.

Referenced by LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::reinit().

8.17.4.2 ky_dof_index

```
template<int dim, int spacedim, typename RangeNumberType = double>
unsigned int LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::ky_←
dof_index [private]
```

Index for accessing the list of DoFs in tensor product order in K_y.

Referenced by LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >::reinit().

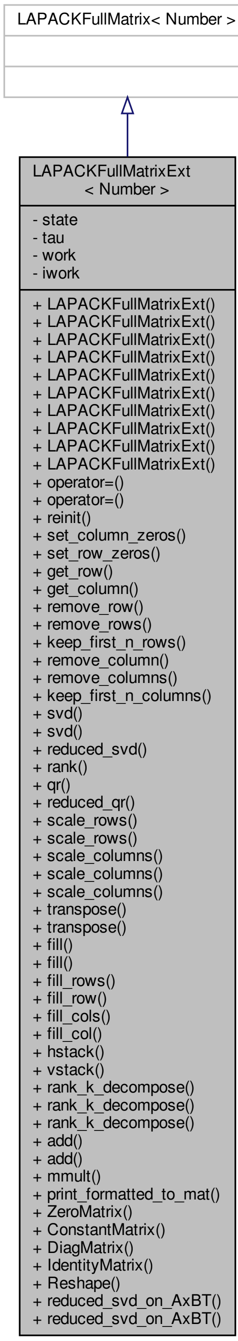
The documentation for this class was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

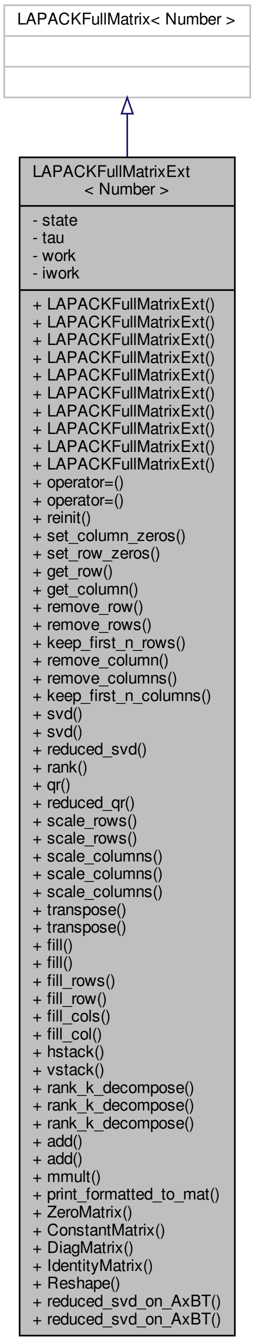
8.18 LAPACKFullMatrixExt< Number > Class Template Reference

```
#include <lapack_full_matrix_ext.h>
```

Inheritance diagram for LAPACKFullMatrixExt< Number >:



Collaboration diagram for LAPACKFullMatrixExt< Number >:



Public Types

- using size_type = std::make_unsigned< types::blas_int >::type

Public Member Functions

- LAPACKFullMatrixExt (const size_type size=0)

- LAPACKFullMatrixExt (const size_type rows, const size_type cols)
- LAPACKFullMatrixExt (const LAPACKFullMatrixExt &mat)
- LAPACKFullMatrixExt (const LAPACKFullMatrix< Number > &mat)
- LAPACKFullMatrixExt (const std::vector< types::global_dof_index > &tau_index_set, const std::vector< types::global_dof_index > &sigma_index_set, const LAPACKFullMatrixExt< Number > &M)
- LAPACKFullMatrixExt (const std::vector< types::global_dof_index > &tau_index_set, const std::vector< types::global_dof_index > &sigma_index_set, const LAPACKFullMatrixExt< Number > &M, const std↵ ::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M)
- LAPACKFullMatrixExt (const LAPACKFullMatrixExt &M1, const LAPACKFullMatrixExt &M2, bool is_↵ horizontal_split)
- LAPACKFullMatrixExt (const std::map< types::global_dof_index, size_t > &row_index_global_to_local↵ _map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M, const LAPACKFullMatrixExt &M1, const std::vector< types::global_dof_index > &M1_tau_index_set, const std::vector< types::global_dof_index > &M1_sigma_index_set, const LAPACKFullMatrixExt &M2, const std::vector< types::global_dof_index > &M2_tau_index_set, const std::vector< types::global_dof_index > &M2_sigma_index_set, bool is_horizontal_split)
- LAPACKFullMatrixExt (const LAPACKFullMatrixExt &M11, const LAPACKFullMatrixExt &M12, const LAP↵ ACKFullMatrixExt &M21, const LAPACKFullMatrixExt &M22)
- LAPACKFullMatrixExt (const std::map< types::global_dof_index, size_t > &row_index_global_to_local↵ _map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_↵ M, const LAPACKFullMatrixExt &M11, const std::vector< types::global_dof_index > &M11_tau_index_set, const std::vector< types::global_dof_index > &M11_sigma_index_set, const LAPACKFullMatrixExt &M12, const std::vector< types::global_dof_index > &M12_tau_index_set, const std::vector< types::global_dof↵ _index > &M12_sigma_index_set, const LAPACKFullMatrixExt &M21, const std::vector< types::global↵ _dof_index > &M21_tau_index_set, const std::vector< types::global_dof_index > &M21_sigma_index_↵ set, const LAPACKFullMatrixExt &M22, const std::vector< types::global_dof_index > &M22_tau_index_set, const std::vector< types::global_dof_index > &M22_sigma_index_set)
- LAPACKFullMatrixExt< Number > & operator= (const LAPACKFullMatrixExt< Number > &matrix)
- LAPACKFullMatrixExt< Number > & operator= (const LAPACKFullMatrix< Number > &matrix)
- void **reinit** (const size_type nrows, const size_type ncols)
- void set_column_zeros (const size_type col_index)
- void set_row_zeros (const size_type row_index)
- void get_row (const size_type row_index, Vector< Number > &row_values) const
- void get_column (const size_type col_index, Vector< Number > &col_values) const
- void remove_row (const size_type row_index)
- void remove_rows (const std::vector< size_type > &row_indices)
- void keep_first_n_rows (const size_type n, bool other_rows_removed=true)
- void remove_column (const size_type column_index)
- void remove_columns (const std::vector< size_type > &column_indices)
- void keep_first_n_columns (const size_type n, bool other_columns_removed=true)
- void svd (LAPACKFullMatrixExt< Number > &U, std::vector< typename numbers::NumberTraits< Number >::real_type > &Sigma_r, LAPACKFullMatrixExt< Number > &VT)
- void svd (LAPACKFullMatrixExt< Number > &U, std::vector< typename numbers::NumberTraits< Number >::real_type > &Sigma_r, LAPACKFullMatrixExt< Number > &VT, const size_type truncation_rank)
- size_type reduced_svd (LAPACKFullMatrixExt< Number > &U, std::vector< typename numbers::Number↵ Traits< Number >::real_type > &Sigma_r, LAPACKFullMatrixExt< Number > &VT, size_type truncation_↵ rank, Number singular_value_threshold=0.)
- size_type rank (Number threshold=0.) const
- void qr (LAPACKFullMatrixExt< Number > &Q, LAPACKFullMatrixExt< Number > &R)
- void reduced_qr (LAPACKFullMatrixExt< Number > &Q, LAPACKFullMatrixExt< Number > &R)
- void scale_rows (const std::vector< typename numbers::NumberTraits< Number >::real_type > &V)
- void scale_rows (LAPACKFullMatrixExt< Number > &A, const std::vector< typename numbers::Number↵ Traits< Number >::real_type > &V) const
- void scale_columns (const std::vector< typename numbers::NumberTraits< Number >::real_type > &V)

- void scale_columns (LAPACKFullMatrixExt< Number > &A, const std::vector< typename numbers::NumberTraits< Number >::real_type > &V) const
- void scale_columns (const Vector< typename numbers::NumberTraits< Number >::real_type > &V)
- void transpose ()
- void transpose (LAPACKFullMatrixExt< Number > &AT) const
- template< typename MatrixType >
  void fill (const MatrixType &src, const size_type dst_offset_i=0, const size_type dst_offset_j=0, const size_type src_offset_i=0, const size_type src_offset_j=0, const Number factor=1., const bool transpose=false)
- void fill (const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map, const LAPACKFullMatrixExt &M, const std::vector< types::global_dof_index > &M_tau_index_set, const std::vector< types::global_dof_index > &M_sigma_index_set)
- void fill_rows (const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map, const LAPACKFullMatrixExt &M, const std::vector< types::global_dof_index > &M_row_global_index_set)
- void fill_row (const size_type row_index, const Vector< Number > &values)
- void fill_cols (const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map, const LAPACKFullMatrixExt &M, const std::vector< types::global_dof_index > &M_col_global_index_set)
- void fill_col (const size_type col_index, const Vector< Number > &values)
- void hstack (LAPACKFullMatrixExt< Number > &C, const LAPACKFullMatrixExt< Number > &B) const
- void vstack (LAPACKFullMatrixExt< Number > &C, const LAPACKFullMatrixExt< Number > &B) const
- size_type rank_k_decompose (const unsigned int k, LAPACKFullMatrixExt< Number > &A, LAPACKFullMatrixExt< Number > &B)
- size_type rank_k_decompose (LAPACKFullMatrixExt< Number > &A, LAPACKFullMatrixExt< Number > &B)
- size_type rank_k_decompose (const unsigned int k, LAPACKFullMatrixExt< Number > &A, LAPACKFullMatrixExt< Number > &B, bool is_left_associative)
- void add (LAPACKFullMatrixExt< Number > &C, const LAPACKFullMatrixExt< Number > &B) const
- void add (const LAPACKFullMatrixExt< Number > &B)
- void mmult (LAPACKFullMatrixExt< Number > &C, const LAPACKFullMatrixExt< Number > &B, const bool adding=false) const
- void print_formatted_to_mat (std::ostream &out, const std::string &name, const unsigned int precision=8, const bool scientific=true, const unsigned int width=0, const char *zero_string="0", const double denominator=1., const double threshold=0.) const

Static Public Member Functions

- static void ZeroMatrix (const size_type rows, const size_type cols, LAPACKFullMatrixExt< Number > &matrix)
- static void ConstantMatrix (const size_type rows, const size_type cols, Number value, LAPACKFullMatrixExt< Number > &matrix)
- static void DiagMatrix (const size_type dim, Number value, LAPACKFullMatrixExt &matrix)
- static void IdentityMatrix (const size_type dim, LAPACKFullMatrixExt &matrix)
- static void Reshape (const size_type rows, const size_type cols, const std::vector< Number > &values, LAPACKFullMatrixExt< Number > &matrix)
- static size_type reduced_svd_on_AxBT (LAPACKFullMatrixExt< Number > &A, LAPACKFullMatrixExt< Number > &B, LAPACKFullMatrixExt< Number > &U, std::vector< typename numbers::NumberTraits< Number >::real_type > &Sigma_r, LAPACKFullMatrixExt< Number > &VT, Number singular_value_threshold=0.)
- static size_type reduced_svd_on_AxBT (LAPACKFullMatrixExt< Number > &A, LAPACKFullMatrixExt< Number > &B, LAPACKFullMatrixExt< Number > &U, std::vector< typename numbers::NumberTraits< Number >::real_type > &Sigma_r, LAPACKFullMatrixExt< Number > &VT, size_type truncation_rank, Number singular_value_threshold=0.)

Private Attributes

- LAPACKSupport::State **state**
- std::vector< typename numbers::NumberTraits< Number >::real_type > tau
- std::vector< Number > work
- std::vector< types::blas_int > iwork

8.18.1 Detailed Description

template<typename Number>
class LAPACKFullMatrixExt< Number >

Extend and expose more of the the functionality of LAPACKFullMatrix.

8.18.2 Member Typedef Documentation

8.18.2.1 size_type

```
template<typename Number>
using LAPACKFullMatrixExt< Number >::size_type = std::make_unsigned<types::blas_int>::type
```

Declare the type for container size.

8.18.3 Constructor & Destructor Documentation

8.18.3.1 LAPACKFullMatrixExt() [1/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const size_type size = 0 )
```

Construct a square matrix by specifying the dimension.

8.18.3.2 LAPACKFullMatrixExt() [2/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const size_type rows,
            const size_type cols )
```

Construct a matrix by specifying the number of rows and columns.

Parameters

rows	
cols	

8.18.3.3 LAPACKFullMatrixExt() [3/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const LAPACKFullMatrixExt< Number > & mat )
```

Copy constructor from an LAPACKFullMatrixExt object.

Parameters

mat	

8.18.3.4 LAPACKFullMatrixExt() [4/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const LAPACKFullMatrix< Number > & mat )
```

Copy constructor from an LAPACKFullMatrix object.

Parameters

mat	

8.18.3.5 LAPACKFullMatrixExt() [5/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const std::vector< types::global_dof_index > & tau_index_set,
            const std::vector< types::global_dof_index > & sigma_index_set,
            const LAPACKFullMatrixExt< Number > & M )
```

Construct a full matrix by restriction to the block cluster $\tau \times \sigma$ from the global full matrix $\mathtt{M}$.

Parameters

tau	
sigma	
M	

Extract the data for the submatrix defined on the block cluster $\tau \times \sigma$ from the full global matrix M.

Because M is global, the indices in τ and σ can be directly used for accessing the elements of M.

8.18.3.6 LAPACKFullMatrixExt() [6/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const std::vector< types::global_dof_index > & tau_index_set,
            const std::vector< types::global_dof_index > & sigma_index_set,
            const LAPACKFullMatrixExt< Number > & M,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M )
```

Construct a full matrix by restriction to the block cluster $\tau \times \sigma$ from the local full matrix M.

Parameters

tau	
sigma	
M	
row_index_global_to_local_map_for↩ _M	
col_index_global_to_local_map_for_M	

Extract the data for the submatrix block $b = \tau \times \sigma$ in the original matrix M.

8.18.3.7 LAPACKFullMatrixExt() [7/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const LAPACKFullMatrixExt< Number > & M1,
            const LAPACKFullMatrixExt< Number > & M2,
            bool is_horizontal_split )
```

Construct a full matrix M from an agglomeration of two full submatrices M_1 and M_2, which have been obtained from either horizontal splitting or vertical splitting.

When the two submatrices have been obtained from horizontal splitting, vstack will be used for the agglomeration. When the two submatrices have been obtained from vertical splitting, hstack will be used for the agglomeration.

Parameters

M1	
M2	
is_horizontal_split	

References LAPACKFullMatrixExt< Number >::hstack(), and LAPACKFullMatrixExt< Number >::vstack().

8.18.3.8 LAPACKFullMatrixExt() [8/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M,
            const LAPACKFullMatrixExt< Number > & M1,
            const std::vector< types::global_dof_index > & M1_tau_index_set,
            const std::vector< types::global_dof_index > & M1_sigma_index_set,
            const LAPACKFullMatrixExt< Number > & M2,
            const std::vector< types::global_dof_index > & M2_tau_index_set,
            const std::vector< types::global_dof_index > & M2_sigma_index_set,
            bool is_horizontal_split )
```

Construct a full matrix M from an agglomeration of two full submatrices M_1 and M_2, which have been obtained from either horizontal splitting or vertical splitting.

When the two submatrices have been obtained from horizontal splitting, `vstack` will be used for the agglomeration. When the two submatrices have been obtained from vertical splitting, `hstack` will be used for the agglomeration.

This method handles the case when the index sets of child clusters are interwoven together into the index set of the parent cluster. This is based on the fact that during DoF support point coordinates based cluster tree partition, the continuity of the index set is not preserved. Make assertions about the submatrix sizes and corresponding index sets.

Perform vertical stacking of the two submatrices.

Perform horizontal stacking of the two submatrices.

References LAPACKFullMatrixExt< Number >::fill().

8.18.3.9 LAPACKFullMatrixExt() [9/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const LAPACKFullMatrixExt< Number > & M11,
            const LAPACKFullMatrixExt< Number > & M12,
            const LAPACKFullMatrixExt< Number > & M21,
            const LAPACKFullMatrixExt< Number > & M22 )
```

Construct a full matrix M from an agglomeration of four full submatrices, $M_{11}, M_{12}, M_{21}, M_{22}$.

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

Note

1. This method implements (7.7) for full matrices in Hackbusch's $\mathcal{H}$-matrix book.
2. This method is only applicable in the case when the cardinality based cluster tree partition is used.

References LAPACKFullMatrixExt< Number >::fill().

8.18.3.10 LAPACKFullMatrixExt() [10/10]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt (
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M,
            const LAPACKFullMatrixExt< Number > & M11,
            const std::vector< types::global_dof_index > & M11_tau_index_set,
            const std::vector< types::global_dof_index > & M11_sigma_index_set,
            const LAPACKFullMatrixExt< Number > & M12,
            const std::vector< types::global_dof_index > & M12_tau_index_set,
            const std::vector< types::global_dof_index > & M12_sigma_index_set,
            const LAPACKFullMatrixExt< Number > & M21,
            const std::vector< types::global_dof_index > & M21_tau_index_set,
            const std::vector< types::global_dof_index > & M21_sigma_index_set,
            const LAPACKFullMatrixExt< Number > & M22,
            const std::vector< types::global_dof_index > & M22_tau_index_set,
            const std::vector< types::global_dof_index > & M22_sigma_index_set )
```

Construct a full matrix M from an agglomeration of four full submatrices, $M_{11}, M_{12}, M_{21}, M_{22}$.

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

Note

1. This method implements (7.7) for full matrices in Hackbusch's $\mathcal{H}$-matrix book.
2. This method handles the case when the index sets of several child clusters are interwoven together into the index set of the parent cluster. This is based on the fact that during DoF support point coordinates based cluster tree partition, the continuity of the index set is not preserved.

Parameters

row_index_global_to_local_map_for↩ _M	
col_index_global_to_local_map_for_M	
M11	
M11_tau_index_set	
M11_sigma_index_set	
M12	
M12_tau_index_set	
M12_sigma_index_set	
M21	
M21_tau_index_set	
M21_sigma_index_set	
M22	
M22_tau_index_set	
M22_sigma_index_set	

Make assertions about the compatibility of row and column numbers of submatrices.

Make further detailed assertions about the cluster index sets of submatrices.

Make assertions about the sizes of row and column index global to local maps for M.

Make assertions about the submatrix sizes and corresponding index sets.

References LAPACKFullMatrixExt< Number >::fill(), and LAPACKFullMatrixExt< Number >::operator=().

8.18.4 Member Function Documentation

8.18.4.1 add() [1/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::add (
            LAPACKFullMatrixExt< Number > & C,
            const LAPACKFullMatrixExt< Number > & B ) const
```

Add two matrix into a new matrix $C = A + B$, where A is the current matrix.

Referenced by main().

8.18.4.2 add() [2/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::add (
            const LAPACKFullMatrixExt< Number > & B )
```

Add the matrix B into the current matrix.

Parameters

B	

8.18.4.3 ConstantMatrix()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::ConstantMatrix (
            const size_type rows,
            const size_type cols,
            Number value,
            LAPACKFullMatrixExt< Number > & matrix )   [static]
```

Create constant valued matrix.

Parameters

rows	
cols	
value	
matrix	

Referenced by main().

8.18.4.4 DiagMatrix()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::DiagMatrix (
            const size_type dim,
            Number value,
            LAPACKFullMatrixExt< Number > & matrix ) [static]
```

Create a constant valued diagonal matrix.

Parameters

dim	
value	
matrix	

8.18.4.5 fill() [1/2]

```
template<typename Number >
template<typename MatrixType >
void LAPACKFullMatrixExt< Number >::fill (
            const MatrixType & src,
            const size_type dst_offset_i = 0,
            const size_type dst_offset_j = 0,
            const size_type src_offset_i = 0,
            const size_type src_offset_j = 0,
            const Number factor = 1.,
            const bool transpose = false )
```

Fill a rectangular block. This function has the similar behavior as FullMatrix::fill.

Parameters

src	
dst_↩ offset_i	
dst_↩ offset_j	

Parameters

src_↩ offset_i	
src_↩ offset_j	
factor	
transpose	

Determine the number of rows and columns to be copied.

Referenced by LAPACKFullMatrixExt< Number >::hstack(), LAPACKFullMatrixExt< Number >::LAPACKFull↩ MatrixExt(), and LAPACKFullMatrixExt< Number >::vstack().

8.18.4.6 fill() [2/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::fill (
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map,
            const LAPACKFullMatrixExt< Number > & M,
            const std::vector< types::global_dof_index > & M_tau_index_set,
            const std::vector< types::global_dof_index > & M_sigma_index_set )
```

Fill the matrix `M` into the current matrix based on the global block cluster indices.

Parameters

row_index_global_to_local_map	
col_index_global_to_local_map	
M	
M_tau_index_set	
M_sigma_index_set	

Make assertions about the sizes of row and column index global to local maps for the current matrix.

8.18.4.7 fill_col()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::fill_col (
            const size_type col_index,
            const Vector< Number > & values )
```

Fill the `values` to the `col_index'th` column of the current matrix.

Parameters

col_index	
values	

Referenced by h_f_mmult().

8.18.4.8 fill_cols()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::fill_cols (
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map,
            const LAPACKFullMatrixExt< Number > & M,
            const std::vector< types::global_dof_index > & M_col_global_index_set )
```

Fill the data in `M` by columns into the current matrix based on the global cluster indices.

Parameters

col_index_global_to_local_map	
M	
M_col_global_index_set	

8.18.4.9 fill_row()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::fill_row (
            const size_type row_index,
            const Vector< Number > & values )
```

Fill the `values` to the `row_index'th` row of the current matrix.

Parameters

row_index	
values	

Referenced by f_h_mmult().

8.18.4.10 fill_rows()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::fill_rows (
```

```
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map,
            const LAPACKFullMatrixExt< Number > & M,
            const std::vector< types::global_dof_index > & M_row_global_index_set )
```

Fill the data in `M` by rows into the current matrix based on the global cluster indices.

Note This function will be used in the RkMatrix constructor via an agglomeration from submatrices.

Parameters

row_index_global_to_local_map	
M	
M_row_global_index_set	

8.18.4.11 get_column()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::get_column (
            const size_type col_index,
            Vector< Number > & col_values ) const
```

Get the values of the column `col_index` into a `Vector`.

Parameters

col_index	
col_values	

Referenced by h_f_mmult().

8.18.4.12 get_row()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::get_row (
            const size_type row_index,
            Vector< Number > & row_values ) const
```

Get the values of the row `row_index` into a `Vector`.

Parameters

row_index	
row_values	

Referenced by f_h_mmult().

8.18.4.13 hstack()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::hstack (
            LAPACKFullMatrixExt< Number > & C,
            const LAPACKFullMatrixExt< Number > & B ) const
```

Horizontally stack two matrices, $C = [A, B]$.

Parameters

C	
B	

References LAPACKFullMatrixExt< Number >::fill().

Referenced by LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt().

8.18.4.14 IdentityMatrix()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::IdentityMatrix (
            const size_type dim,
            LAPACKFullMatrixExt< Number > & matrix )  [static]
```

Create an identity matrix of dimension `dim`.

Parameters

dim	
matrix	

8.18.4.15 keep_first_n_columns()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::keep_first_n_columns (
            const size_type n,
            bool other_columns_removed = true )
```

Keep only the first n columns of the matrix.

When `other_columns_removed` is true, the other columns are deleted from the matrix, otherwise they are set to zero.

Parameters

n	
other_rows_removed	

Do nothing.

Referenced by LAPACKFullMatrixExt< Number >::reduced_qr(), LAPACKFullMatrixExt< Number >::reduced_svd(), LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT(), and LAPACKFullMatrixExt< Number >::svd().

8.18.4.16 keep_first_n_rows()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::keep_first_n_rows (
            const size_type n,
            bool other_rows_removed = true )
```

Keep only the first n rows of the matrix.

When `other_rows_removed` is true, the other rows are deleted from the matrix, otherwise they are set to zero.

Parameters

n	
other_rows_removed	

Do nothing.

Referenced by LAPACKFullMatrixExt< Number >::reduced_qr(), LAPACKFullMatrixExt< Number >::reduced_svd(), LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT(), and LAPACKFullMatrixExt< Number >::svd().

8.18.4.17 mmult()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::mmult (
            LAPACKFullMatrixExt< Number > & C,
            const LAPACKFullMatrixExt< Number > & B,
            const bool adding = false ) const
```

Multiply two matrices: $C = A \cdot B$

Parameters

C	
B	

Referenced by main(), LAPACKFullMatrixExt< Number >::qr(), and LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT().

8.18.4.18 operator=() [1/2]

```
template<typename Number >
LAPACKFullMatrixExt< Number > & LAPACKFullMatrixExt< Number >::operator= (
            const LAPACKFullMatrixExt< Number > & matrix )
```

Overloaded assignment operator.

Parameters

matrix	

Returns

Referenced by LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt().

8.18.4.19 operator=() [2/2]

```
template<typename Number >
LAPACKFullMatrixExt< Number > & LAPACKFullMatrixExt< Number >::operator= (
            const LAPACKFullMatrix< Number > & matrix )
```

Overloaded assignment operator.

Parameters

matrix	

Returns

8.18.4.20 print_formatted_to_mat()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::print_formatted_to_mat (
            std::ostream & out,
```

```
const std::string & name,
const unsigned int precision = 8,
const bool scientific = true,
const unsigned int width = 0,
const char * zero_string = "0",
const double denominator = 1.,
const double threshold = 0. ) const
```

Print a LAPACKFullMatrixExt to Octave mat format.

Parameters

out	
name	
precision	
scientific	
width	
zero_string	
denominator	
threshold	

Referenced by main().

8.18.4.21 qr()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::qr (
            LAPACKFullMatrixExt< Number > & Q,
            LAPACKFullMatrixExt< Number > & R )
```

Perform QR decomposition of the matrix.

Parameters

Q	
R	

Set work space size as 1 and `lwork` as -1 for the determination of optimal work space size.

Make sure that the first entry in the work space is clear, in case the routine does not completely overwrite the memory:

Resize the work space and add one to the size computed by LAPACK to be on the safe side.

Perform the actual QR decomposition.

Collect results for the upper triangular matrix `R`.

Collect results for the orthogonal matrix `Q` with a dimension $m \times m$. It is represented as a product of elementary reflectors (Householder transformation) as

$$Q = H_1 H_2 \cdots H_k,$$

where $k = \min\{m, n\}$.

Construct the vector `v`. Values in `v` before the i'th component are all zeros. The i'th component is 1. Values after the i'th component are stored in the current matrix `A`(i+1:m,i).

Construct the Householder matrix.

Release the work space used.

References LAPACKFullMatrixExt< Number >::mmult(), LAPACKFullMatrixExt< Number >::tau, and LAPACK←FullMatrixExt< Number >::work.

Referenced by main(), and LAPACKFullMatrixExt< Number >::reduced_qr().

8.18.4.22 rank()

```
template<typename Number >
LAPACKFullMatrixExt< Number >::size_type LAPACKFullMatrixExt< Number >::rank (
            Number threshold = 0. ) const
```

Get the rank of the matrix using SVD.

A copy will be made at first for SVD to operate on.

Parameters

threshold	threshold for singular values. The number of singular values larger than this threshold is the matrix rank.

Returns

If the singular value threshold is perfect zero, calculate a threshold value instead.

References LAPACKFullMatrixExt< Number >::svd().

Referenced by LAPACKFullMatrixExt< Number >::reduced_svd().

8.18.4.23 rank_k_decompose() [1/3]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::size_type LAPACKFullMatrixExt< Number >::rank_k_decompose (
            const unsigned int k,
            LAPACKFullMatrixExt< Number > & A,
            LAPACKFullMatrixExt< Number > & B )
```

Decompose the full matrix into the two components of its rank-k representation, the associativity of triple-matrix multiplication is automatically detected.

Note This method will be used for converting a full matrix to a rank-k matrix, which underlies the operator $\mathcal{T}_r^{\mathcal{R}\leftarrow\mathcal{F}}$ in (7.2) in Hackbusch's $\mathcal{H}$-matrix book.

Parameters

k	
A	
B	

Returns

When the matrix is long, apply right associativity.

When the matrix is wide, apply left associativity.

Referenced by main(), LAPACKFullMatrixExt< Number >::rank_k_decompose(), and RkMatrix< Number >::Rk↩ Matrix().

8.18.4.24 rank_k_decompose() [2/3]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::size_type LAPACKFullMatrixExt< Number >::rank_k_decompose (
            LAPACKFullMatrixExt< Number > & A,
            LAPACKFullMatrixExt< Number > & B )
```

Decompose the full matrix into the two components of its rank-k representation, the associativity of triple-matrix multiplication is automatically detected. This version does not have an actual rank truncation but sets the truncation rank to be the minimum matrix dimension $\min\{m,n\}$.

Note This method will be used for converting a full matrix to a rank-k matrix, which implements the operator $\mathcal{T}_r^{\mathcal{R}\leftarrow\mathcal{F}}$ in (7.2) in Hackbusch's $\mathcal{H}$-matrix book.

Parameters

A	
B	

Returns

Effective rank of the matrix.

Use the minimum matrix dimension as the default truncation rank.

References LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.18.4.25 rank_k_decompose() [3/3]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::size_type LAPACKFullMatrixExt< Number >::rank_k_decompose (
            const unsigned int k,
            LAPACKFullMatrixExt< Number > & A,
            LAPACKFullMatrixExt< Number > & B,
            bool is_left_associative )
```

Decompose the full matrix into the two components of its rank-k representation.

Note Because the `reduced_svd` member function is called by `rank_k_decompose`, the effective rank of the matrix will be returned, which can be smaller than the specified fixed rank `k`.

Parameters

k	User specified truncation rank.
is_left_associative	
A	
B	

Returns

Effective rank.

Perform RSVD for the matrix and return U and VT into A and B respectively. N.B. After running this function, B actually holds the transposition of itself at the moment.

Let A = A∗Sigma_r and B = B^T.

Let A = A and B = (Sigma_r ∗ B)^T.

References LAPACKFullMatrixExt< Number >::reduced_svd(), LAPACKFullMatrixExt< Number >::scale_columns(), LAPACKFullMatrixExt< Number >::scale_rows(), and LAPACKFullMatrixExt< Number >::transpose().

8.18.4.26 reduced_qr()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::reduced_qr (
            LAPACKFullMatrixExt< Number > & Q,
            LAPACKFullMatrixExt< Number > & R )
```

Perform reduced QR decomposition of the matrix.

Parameters

Q	
R	

Perform the standard QR decomposition.

Perform the reduced QR decomposition by keeping the first n columns of Q and the first n rows of R.

References LAPACKFullMatrixExt< Number >::keep_first_n_columns(), LAPACKFullMatrixExt< Number >↩
::keep_first_n_rows(), and LAPACKFullMatrixExt< Number >::qr().

Referenced by main(), and LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT().

8.18.4.27 reduced_svd()

```
template<typename Number >
LAPACKFullMatrixExt< Number >::size_type LAPACKFullMatrixExt< Number >::reduced_svd (
            LAPACKFullMatrixExt< Number > & U,
            std::vector< typename numbers::NumberTraits< Number >::real_type > & Sigma_r,
            LAPACKFullMatrixExt< Number > & VT,
            size_type truncation_rank,
            Number singular_value_threshold = 0. )
```

Perform the reduced singular value decomposition (SVD) with rank truncation.

Note Even though an explicit truncation rank is specified, inside this function, after SVD, the effective rank of the matrix is obtained. Because any given truncation rank value larger than the effective matrix rank is meaningless, it will be limited to be the effective rank.

Parameters

U	
Sigma_r	
VT	
truncation_rank	Truncation rank specified by the user.
singular_value_threshold	

Returns

Effective rank.

Work flow Perform the full SVD.

If the singular value threshold is perfect zero, calculate a threshold value instead.

Get the actual rank of the matrix by counting the total number of singular values which are larger than the given threshold `singular_value_threshold`. The actual rank should always be less than or equal to `min_dim`.

Limit the value of `truncation_rank` not larger than the actual rank just obtained by counting effective singular values.

Perform singular value truncation when the given `truncation_rank` (after value limiting wrt. the actual rank) is less than the minimum matrix dimension. The procedures are as below.

1. Keep the first `truncation_rank` singular values, while discarding the others.

2. Keep the first `truncation_rank` columns of `U`, while discarding the others.
3. Keep the first `truncation_rank` rows of `VT`, while discarding the others.

When `truncation_rank` (after value limiting wrt. the actual rank) is equal to the minimum matrix dimension, we only need to adjust the columns of U or the rows of V^T depending on the shape of the matrix M.

For details, if M has a dimension $m \times n$, the dimensions of all matrices obtained from SVD are:

- $U \in \mathbb{R}^{m \times m}$
- $\Sigma_r \in \mathbb{R}^{m \times n}$
- $V \in \mathbb{R}^{n \times n}$

When M is long, $\Sigma_r = \begin{pmatrix} \Sigma_r' \\ 0 \end{pmatrix}$. Therefore, we keep the first `min_dim` columns of U, while deleting others. V^T is kept intact.

When M is wide, $\Sigma_r = \begin{pmatrix} \Sigma_r' & 0 \end{pmatrix}$. Therefore, we keep the first `min_dim` rows of V^T, while deleting others. U is kept intact.

When M is square, do nothing.

Return the value of `truncation_rank`. Instead of its original specified value, it now contains the actual rank of the matrix after truncation.

References LAPACKFullMatrixExt< Number >::keep_first_n_columns(), LAPACKFullMatrixExt< Number >↩ ::keep_first_n_rows(), LAPACKFullMatrixExt< Number >::rank(), and LAPACKFullMatrixExt< Number >::svd().

Referenced by main(), and LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.18.4.28 reduced_svd_on_AxBT() [1/2]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::size_type LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT (
            LAPACKFullMatrixExt< Number > & A,
            LAPACKFullMatrixExt< Number > & B,
            LAPACKFullMatrixExt< Number > & U,
            std::vector< typename numbers::NumberTraits< Number >::real_type > & Sigma_r,
            LAPACKFullMatrixExt< Number > & VT,
            Number singular_value_threshold = 0. )  [static]
```

Perform SVD on the product of two component matrices A and B^T without rank truncation. If the matrix is not of full rank, truncate it to the effective rank. It returns the effective rank.

Parameters

A	
B	
U	
Sigma↩ _r	
VT	

Work flow In a rank-k matrix, the number of columns in the component matrix `A`, which is the formal rank, should match that of `B`. So we make an assertion here.

N.B. The number of columns in the component matrix `A` or `B` is the formal rank of the rank-k matrix, which means the actual rank of the rank-k matrix is less than this formal rank.

When both `A` and `B` are long matrices, i.e. they have more rows than columns, perform the reduced QR decomposition to component matrix `A`, which has a dimension of $m \times r$.

Perform the reduced QR decomposition to component matrix `B`, which has a dimension of $n \times r$.

Perform SVD to the product R of the two upper triangular matrices, i.e. $R = R_A R_B^T$.

N.B. Before LAPACK matrix multiplication, the memory of the result matrix should be reinitialized.

When `A` or `B` is not a long matrix, firstly convert the rank-k matrix to a full matrix, then perform SVD on this full matrix.

If the singular value threshold is perfect zero, calculate a threshold value instead.

Get the actual rank of the matrix by counting the total number of singular values which are larger than the given threshold `singular_value_threshold`. The actual rank should always be less than or equal to `min_dim`. The matrix will be truncated to this effective rank.

When the matrix is not of full rank, keep the first `rank` singular values, while discarding the others.

Keep the first `rank` columns of `U`, while deleting the others.

Keep the first `rank` rows of `VT`, while deleting the others.

When the matrix is of full rank, i.e. `rank` == `min_dim`.

When QR decomposition has been used, if `M` has a dimension $m \times n$, the dimensions of all matrices are:

- $U \in \mathbb{R}^{m \times \mathrm{representationrank}}$
- $\Sigma_r \in \mathbb{R}^{\mathrm{representationrank} \times \mathrm{representationrank}}$
- $V \in \mathbb{R}^{n \times \mathrm{representationrank}}$

When the matrix rank is less than the formal rank, keep the first `rank` singular values, while discarding the others.

Keep the first `rank` columns of `U`, while deleting the others.

Keep the first `rank` rows of `VT`, while deleting the others.

When QR decomposition has not been used, the results `U`, `Sigma_r` and `VT` are obtained from full matrix SVD. And if `M` has a dimension $m \times n$, the dimensions of all matrices obtained from SVD are:

- $U \in \mathbb{R}^{m \times m}$
- $\Sigma_r \in \mathbb{R}^{m \times n}$
- $V \in \mathbb{R}^{n \times n}$

When `M` is a long matrix, $\Sigma_r = \begin{pmatrix} \Sigma_r' \\ 0 \end{pmatrix}$. Therefore, we keep the first `min_dim` columns of `U`, while deleting the others. `VT` is kept intact.

When `M` is a wide matrix, $\Sigma_r = \begin{pmatrix} \Sigma_r' & 0 \end{pmatrix}$. Therefore, we keep the first `min_dim` rows of `VT`, while deleting the others. `U` is kept intact.

When `M` is square, do nothing.

Return the actual rank of the matrix.

References LAPACKFullMatrixExt< Number >::keep_first_n_columns(), LAPACKFullMatrixExt< Number >::keep_first_n_rows(), LAPACKFullMatrixExt< Number >::mmult(), LAPACKFullMatrixExt< Number >::reduced_qr(), and LAPACKFullMatrixExt< Number >::svd().

Referenced by RkMatrix< Number >::truncate_to_rank().

8.18.4.29 reduced_svd_on_AxBT() [2/2]

```
template<typename Number >
LAPACKFullMatrixExt< Number >::size_type LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT (
            LAPACKFullMatrixExt< Number > & A,
            LAPACKFullMatrixExt< Number > & B,
            LAPACKFullMatrixExt< Number > & U,
            std::vector< typename numbers::NumberTraits< Number >::real_type > & Sigma_r,
            LAPACKFullMatrixExt< Number > & VT,
            size_type truncation_rank,
            Number singular_value_threshold = 0. ) [static]
```

Perform SVD on the product of two component matrices A and B^T with truncation by a specified rank.

Parameters

A	
B	
U	
Sigma_r	
VT	
truncation_rank	

In a rank-k matrix, the number of columns in the component matrix `A` should match that of `B`.

N.B. The number of columns in the component matrix `A` or `B` is the representation rank of the rank-k matrix, which means the rank-k matrix may not be of full rank and the actual rank is less than this representation rank.

When both `A` and `B` are long matrices, i.e. they have more rows than columns, perform reduced QR decomposition to component matrix `A`, which has a dimension of $m \times r$.

Perform reduced QR decomposition to component matrix `B`, which has a dimension of $n \times r$.

Perform SVD to the product R of the two upper triangular matrices, i.e. $R = R_A R_B^T$.

N.B. Before LAPACK matrix multiplication, the memory of the result matrix should be reinitialized.

Firstly convert the rank-k matrix to a full matrix, then perform SVD on this full matrix.

If the singular value threshold is perfect zero, calculate a threshold value instead.

Get the actual rank of the matrix and it should always be less than or equal to `min_dim`.

Limit the truncation rank wrt. the actual rank.

Keep the first `truncation_rank` singular values, while discarding others.

Keep the first `truncation_rank` columns of `U`, while deleting others.

Keep the first `truncation_rank` rows of `VT`, while deleting others.

In this case, it could only be `truncation_rank` == `min_dim`.

In this case, the results `U`, `Sigma_r` and `VT` are obtained via QR decomposition. And if `M` has a dimension $m \times n$, the dimensions of all matrices are:

- $U \in \mathbb{R}^{m \times \text{representationrank}}$
- $\Sigma_r \in \mathbb{R}^{\text{representationrank} \times \text{representationrank}}$
- $V \in \mathbb{R}^{n \times \text{representationrank}}$

Keep the first `truncation_rank` singular values, while discarding others.

Keep the first `truncation_rank` columns of `U`, while deleting others.

Keep the first `truncation_rank` rows of `VT`, while deleting others.

In this case, the results `U`, `Sigma_r` and `VT` are obtained from full matrix SVD. And if `M` has a dimension $m \times n$, the dimensions of all matrices obtained from SVD are:

- $U \in \mathbb{R}^{m \times m}$
- $\Sigma_r \in \mathbb{R}^{m \times n}$
- $V \in \mathbb{R}^{n \times n}$

When the original matrix is long, $\Sigma_r = \begin{pmatrix} \Sigma_r' \\ 0 \end{pmatrix}$. Therefore, we keep the first `min_dim` columns of `U`, while deleting others. `VT` is kept intact.

When the original matrix is wide, $\Sigma_r = \begin{pmatrix} \Sigma_r' & 0 \end{pmatrix}$. Therefore, we keep the first `min_dim` rows of `VT`, while deleting others. `U` is kept intact.

When the original matrix is square, do nothing.

References LAPACKFullMatrixExt< Number >::keep_first_n_columns(), LAPACKFullMatrixExt< Number >↩ ::keep_first_n_rows(), LAPACKFullMatrixExt< Number >::mmult(), LAPACKFullMatrixExt< Number >::reduced↩ _qr(), and LAPACKFullMatrixExt< Number >::svd().

8.18.4.30 remove_column()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::remove_column (
            const size_type column_index )
```

Remove column `column_index` from the matrix.

Parameters

column_index	

8.18.4.31 remove_columns()

```
template<typename Number >
```

```
void LAPACKFullMatrixExt< Number >::remove_columns (
            const std::vector< size_type > & column_indices )
```

Remove columns `column_indices` from the matrix.

Parameters

column_indices	

8.18.4.32 remove_row()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::remove_row (
            const size_type row_index )
```

Remove row `row_index` from the matrix.

Parameters

row_index	

8.18.4.33 remove_rows()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::remove_rows (
            const std::vector< size_type > & row_indices )
```

Remove rows `row_indices` from the matrix.

Parameters

row_indices	

8.18.4.34 Reshape()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::Reshape (
            const size_type rows,
            const size_type cols,
            const std::vector< Number > & values,
            LAPACKFullMatrixExt< Number > & matrix )  [static]
```

Reshape a vector of values into a LAPACKFullMatrixExt in column major.

Referenced by main().

8.18.4.35 scale_columns() [1/3]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::scale_columns (
            const std::vector< typename numbers::NumberTraits< Number >::real_type > & V )
```

Right multiply the matrix with a diagonal matrix V, which is stored in a std::vector.

Parameters

V	

Referenced by main(), LAPACKFullMatrixExt< Number >::rank_k_decompose(), and RkMatrix< Number >::truncate_to_rank().

8.18.4.36 scale_columns() [2/3]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::scale_columns (
            LAPACKFullMatrixExt< Number > & A,
            const std::vector< typename numbers::NumberTraits< Number >::real_type > & V )
const
```

Right multiply the matrix with a diagonal matrix V, which is stored in a std::vector. The results are stored in a new matrix.

Parameters

A	
V	

8.18.4.37 scale_columns() [3/3]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::scale_columns (
            const Vector< typename numbers::NumberTraits< Number >::real_type > & V )
```

Right multiply the matrix with a diagonal matrix V, which is stored in a Vector.

Parameters

V	

8.18.4.38 scale_rows() [1/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::scale_rows (
            const std::vector< typename numbers::NumberTraits< Number >::real_type > & V )
```

Left multiply the matrix with a diagonal matrix V, which is stored in a std::vector.

Parameters

V	

Referenced by main(), LAPACKFullMatrixExt< Number >::rank_k_decompose(), and RkMatrix< Number >::truncate_to_rank().

8.18.4.39 scale_rows() [2/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::scale_rows (
            LAPACKFullMatrixExt< Number > & A,
            const std::vector< typename numbers::NumberTraits< Number >::real_type > & V )
const
```

Left multiply the matrix with a diagonal matrix V, which is stored in a std::vector. The results are stored in a new matrix.

Parameters

A	
V	

8.18.4.40 set_column_zeros()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::set_column_zeros (
            const size_type col_index )
```

Set a matrix column as zeros.

8.18.4.41 set_row_zeros()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::set_row_zeros (
            const size_type row_index )
```

Set a matrix row as zeros.

8.18.4.42 svd() [1/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::svd (
            LAPACKFullMatrixExt< Number > & U,
            std::vector< typename numbers::NumberTraits< Number >::real_type > & Sigma_r,
            LAPACKFullMatrixExt< Number > & VT )
```

Perform the standard singular value decomposition (SVD).

Parameters

U	with a dimension $m \times m$.
Sigma_r	the list of singular values, with a dimension $\min(m, n)$.
VT	with a dimension $n \times n$

Allocate memory for result matrices.

Integer array as the work space for pivoting, which is the `IWORK` parameter in `dgesdd`.

Return status.

Determine the optimal work space size, which will be stored into `real_work`. At the moment, `work` is a scalar which will hold the queried optimal `lwork`. Its first element should be initialized as zero.

Store the optimal work space size after inquiry.

Integer work space for pivoting.

lwork = -1, for an inquiry of the optimal work space size.

Update the work space size and resize the `work` vector.

Perform the real SVD.

Integer work space for pivoting.

References LAPACKFullMatrixExt< Number >::work.

Referenced by main(), LAPACKFullMatrixExt< Number >::rank(), LAPACKFullMatrixExt< Number >::reduced_svd(), LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT(), and LAPACKFullMatrixExt< Number >::svd().

8.18.4.43 svd() [2/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::svd (
            LAPACKFullMatrixExt< Number > & U,
            std::vector< typename numbers::NumberTraits< Number >::real_type > & Sigma_r,
            LAPACKFullMatrixExt< Number > & VT,
            const size_type truncation_rank )
```

Perform the standard singular value decomposition (SVD) with rank truncation.

When the given `truncation_rank` is less than the minimum dimension $\min m, n$, `U's` (truncation_rank+1)'th to `m'th` columns are set to zeros; `VT's` (truncation_rank+1)'th to `n'th` rows are set to zeros; `Sigma_r's` (truncation_rank+1)'th to `n'th` values are set to zeros.

Parameters

U	with a dimension $m \times m$.
Sigma←_r	the list of singular values, which has a dimension $\min(m, n)$.
VT	with a dimension $n \times n$.

Perform the full SVD. After the operation,

Perform singular value truncation when the specified rank is less than the minimum dimension.

Keep the first `truncation_rank` number of singular values and clear the remaining ones.

Keep the first `truncation_rank` columns of `U`, while setting other columns to zero.

Keep the first `truncation_rank` rows of `VT`, while setting other rows to zero.

Do not thing when the truncation rank is equal to or larger than `min_dim`. This means the truncation is an identity operator.

References LAPACKFullMatrixExt< Number >::keep_first_n_columns(), LAPACKFullMatrixExt< Number >←::keep_first_n_rows(), and LAPACKFullMatrixExt< Number >::svd().

8.18.4.44 transpose() [1/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::transpose ( )
```

Perform in-place transpose of the matrix.

Referenced by LAPACKFullMatrixExt< Number >::rank_k_decompose(), and RkMatrix< Number >::truncate_←to_rank().

8.18.4.45 transpose() [2/2]

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::transpose (
            LAPACKFullMatrixExt< Number > & AT ) const
```

Get the transpose of the current matrix into a new matrix `AT`.

Parameters

AT	

8.18.4.46 vstack()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::vstack (
            LAPACKFullMatrixExt< Number > & C,
            const LAPACKFullMatrixExt< Number > & B ) const
```

Vertically stack two matrices, $C = [A; B]$.

Parameters

C	
B	

References LAPACKFullMatrixExt< Number >::fill().

Referenced by LAPACKFullMatrixExt< Number >::LAPACKFullMatrixExt().

8.18.4.47 ZeroMatrix()

```
template<typename Number >
void LAPACKFullMatrixExt< Number >::ZeroMatrix (
            const size_type rows,
            const size_type cols,
            LAPACKFullMatrixExt< Number > & matrix )  [static]
```

Create a zero valued matrix.

Parameters

rows	
cols	
matrix	

8.18.5 Member Data Documentation

8.18.5.1 iwork

```
template<typename Number>
std::vector<types::blas_int> LAPACKFullMatrixExt< Number >::iwork  [private]
```

Integer work array used by LAPACK routines.

8.18.5.2 tau

```
template<typename Number>
std::vector<typename numbers::NumberTraits<Number>::real_type> LAPACKFullMatrixExt< Number
>::tau [private]
```

The scalar factors of the elementary reflectors.

Referenced by LAPACKFullMatrixExt< Number >::qr().

8.18.5.3 work

```
template<typename Number>
std::vector<Number> LAPACKFullMatrixExt< Number >::work [private]
```

Work space used by LAPACK routines.

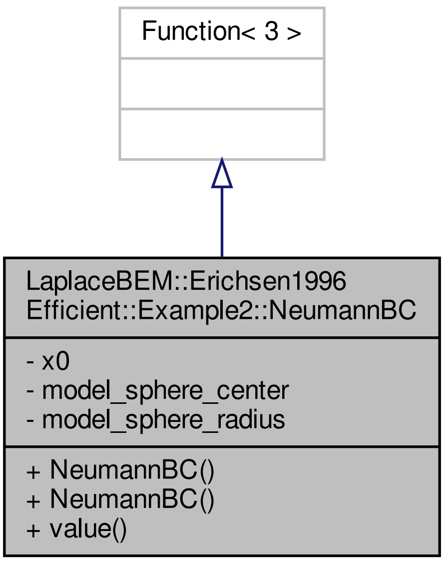
Referenced by LAPACKFullMatrixExt< Number >::qr(), and LAPACKFullMatrixExt< Number >::svd().

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

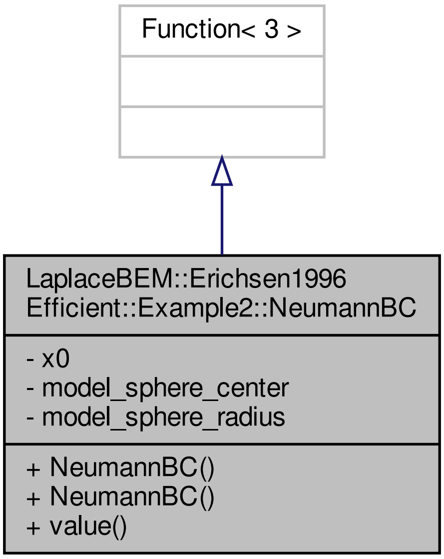
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/lapack_full_matrix_ext.h

8.19 LaplaceBEM::Erichsen1996Efficient::Example2::NeumannBC Class Reference

Inheritance diagram for LaplaceBEM::Erichsen1996Efficient::Example2::NeumannBC:



Collaboration diagram for LaplaceBEM::Erichsen1996Efficient::Example2::NeumannBC:



Public Member Functions

- **NeumannBC** (const Point< 3 > &x0, const Point< 3 > &center, double radius)
- double **value** (const Point< 3 > &p, const unsigned int component=0) const

Private Attributes

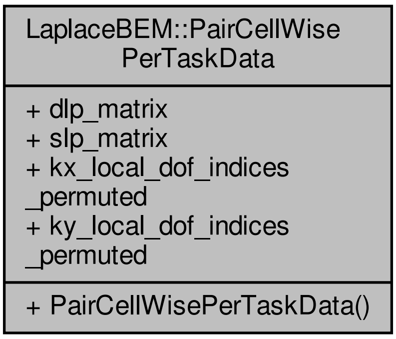
- Point< 3 > **x0**
- Point< 3 > **model_sphere_center**
- double **model_sphere_radius**

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

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/erichsen1996efficient_↩ example2.h

8.20 LaplaceBEM::PairCellWisePerTaskData Struct Reference

Collaboration diagram for LaplaceBEM::PairCellWisePerTaskData:



Public Member Functions

- **PairCellWisePerTaskData** (const FiniteElement< 2, 3 > &kx_fe, const FiniteElement< 2, 3 > &ky_fe)

Public Attributes

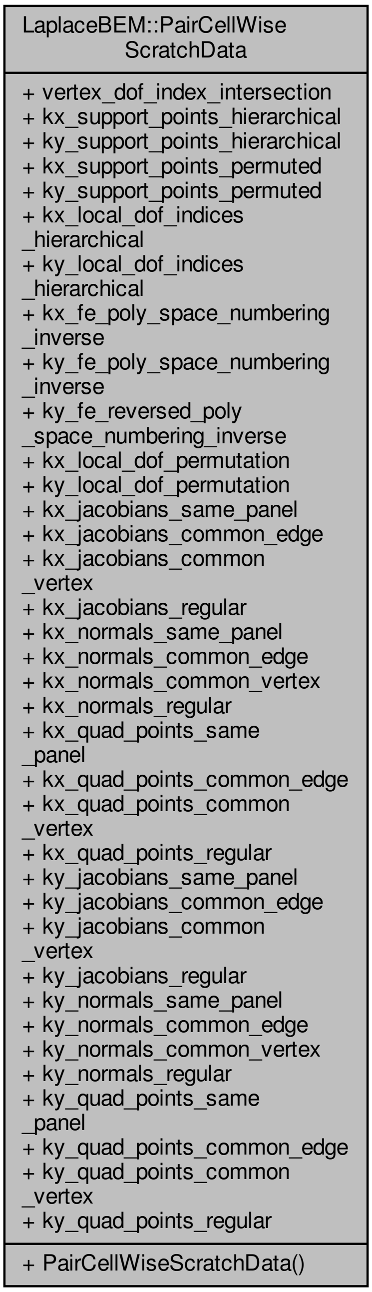
- FullMatrix< double > **dlp_matrix**
- FullMatrix< double > **slp_matrix**
- std::vector< types::global_dof_index > **kx_local_dof_indices_permuted**
- std::vector< types::global_dof_index > **ky_local_dof_indices_permuted**

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

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.21 LaplaceBEM::PairCellWiseScratchData Struct Reference

Collaboration diagram for LaplaceBEM::PairCellWiseScratchData:



Public Member Functions

- **PairCellWiseScratchData** (const FiniteElement< 2, 3 > &kx_fe, const FiniteElement< 2, 3 > &ky_fe, const BEMValues< 2, 3 > &bem_values)

Public Attributes

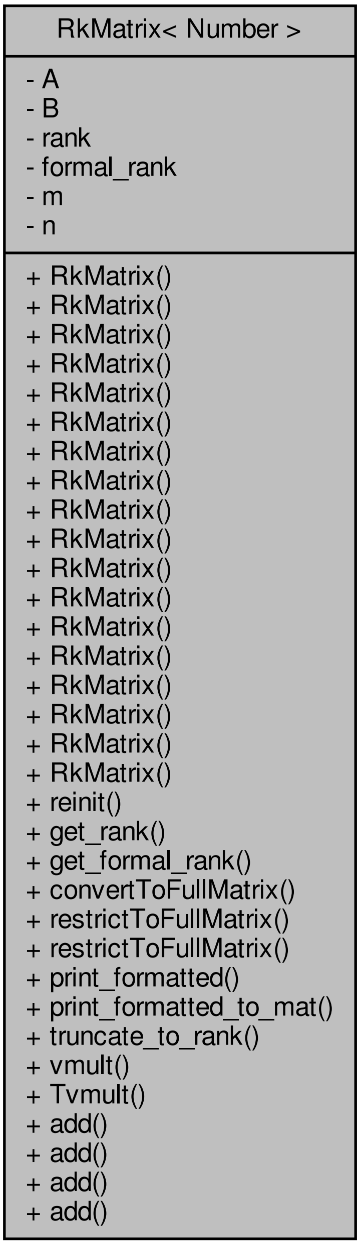
- std::vector< types::global_dof_index > **vertex_dof_index_intersection**
- std::vector< Point< 3 > > **kx_support_points_hierarchical**
- std::vector< Point< 3 > > **ky_support_points_hierarchical**
- std::vector< Point< 3 > > **kx_support_points_permuted**
- std::vector< Point< 3 > > **ky_support_points_permuted**
- std::vector< types::global_dof_index > **kx_local_dof_indices_hierarchical**
- std::vector< types::global_dof_index > **ky_local_dof_indices_hierarchical**
- std::vector< unsigned int > **kx_fe_poly_space_numbering_inverse**
- std::vector< unsigned int > **ky_fe_poly_space_numbering_inverse**
- std::vector< unsigned int > **ky_fe_reversed_poly_space_numbering_inverse**
- std::vector< unsigned int > **kx_local_dof_permutation**
- std::vector< unsigned int > **ky_local_dof_permutation**
- Table< 2, double > **kx_jacobians_same_panel**
- Table< 2, double > **kx_jacobians_common_edge**
- Table< 2, double > **kx_jacobians_common_vertex**
- Table< 2, double > **kx_jacobians_regular**
- Table< 2, Tensor< 1, 3 > > **kx_normals_same_panel**
- Table< 2, Tensor< 1, 3 > > **kx_normals_common_edge**
- Table< 2, Tensor< 1, 3 > > **kx_normals_common_vertex**
- Table< 2, Tensor< 1, 3 > > **kx_normals_regular**
- Table< 2, Point< 3 > > **kx_quad_points_same_panel**
- Table< 2, Point< 3 > > **kx_quad_points_common_edge**
- Table< 2, Point< 3 > > **kx_quad_points_common_vertex**
- Table< 2, Point< 3 > > **kx_quad_points_regular**
- Table< 2, double > **ky_jacobians_same_panel**
- Table< 2, double > **ky_jacobians_common_edge**
- Table< 2, double > **ky_jacobians_common_vertex**
- Table< 2, double > **ky_jacobians_regular**
- Table< 2, Tensor< 1, 3 > > **ky_normals_same_panel**
- Table< 2, Tensor< 1, 3 > > **ky_normals_common_edge**
- Table< 2, Tensor< 1, 3 > > **ky_normals_common_vertex**
- Table< 2, Tensor< 1, 3 > > **ky_normals_regular**
- Table< 2, Point< 3 > > **ky_quad_points_same_panel**
- Table< 2, Point< 3 > > **ky_quad_points_common_edge**
- Table< 2, Point< 3 > > **ky_quad_points_common_vertex**
- Table< 2, Point< 3 > > **ky_quad_points_regular**

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

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.22 RkMatrix$<$ Number $>$ Class Template Reference

Collaboration diagram for RkMatrix$<$ Number $>$:



Public Types

- using size_type = std::make_unsigned$<$ types::blas_int $>$::type

Public Member Functions

- RkMatrix ()
- RkMatrix (const size_type m, const size_type n, const size_type fixed_rank_k)
- RkMatrix (const size_type fixed_rank_k, LAPACKFullMatrixExt< Number > &M)
- RkMatrix (LAPACKFullMatrixExt< Number > &M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const size_type fixed_rank_k, const LAPACKFullMatrixExt< Number > &M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const LAPACKFullMatrixExt< Number > &M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const size_type fixed_rank_k, const LAPACKFullMatrixExt< Number > &M, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const LAPACKFullMatrixExt< Number > &M, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const size_type fixed_rank_k, const RkMatrix< Number > &M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const RkMatrix< Number > &M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const size_type fixed_rank_k, const RkMatrix< Number > &M, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M)
- RkMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const RkMatrix< Number > &M, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M)
- RkMatrix (const LAPACKFullMatrixExt< Number > &A, const LAPACKFullMatrixExt< Number > &B)
- RkMatrix (const size_type fixed_rank_k, const RkMatrix< Number > &M1, const RkMatrix< Number > &M2, bool is_horizontal_split)
- RkMatrix (const size_type fixed_rank_k, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M, const RkMatrix< Number > &M1, const std::vector< types::global_dof_index > &M1_tau_index_set, const std::vector< types::global_dof_index > &M1_sigma_index_set, const RkMatrix< Number > &M2, const std::vector< types::global_dof_index > &M2_tau_index_set, const std::vector< types::global_dof_index > &M2_sigma_index_set, bool is_horizontal_split)
- RkMatrix (const size_type fixed_rank_k, const RkMatrix< Number > &M11, const RkMatrix< Number > &M12, const RkMatrix< Number > &M21, const RkMatrix< Number > &M22)
- RkMatrix (const size_type fixed_rank_k, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M, const RkMatrix< Number > &M11, const std::vector< types::global_dof_index > &M11_tau_index_set, const std::vector< types::global_dof_index > &M11_sigma_index_set, const RkMatrix< Number > &M12, const std::vector< types::global_dof_index > &M12_tau_index_set, const std::vector< types::global_dof_index > &M12_sigma_index_set, const RkMatrix< Number > &M21, const std::vector< types::global_dof_index > &M21_tau_index_set, const std::vector< types::global_dof_index > &M21_sigma_index_set, const RkMatrix< Number > &M22, const std::vector< types::global_dof_index > &M22_tau_index_set, const std::vector< types::global_dof_index > &M22_sigma_index_set)
- RkMatrix (const RkMatrix< Number > &matrix)
- void reinit (const size_type m, const size_type n, const size_type fixed_rank_k)
- size_type get_rank () const
- size_type get_formal_rank () const
- void convertToFullMatrix (LAPACKFullMatrixExt< Number > &matrix) const
- void restrictToFullMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, LAPACKFullMatrixExt< Number > &matrix) const

- void restrictToFullMatrix (const std::vector< types::global_dof_index > &tau, const std::vector< types::global_dof_index > &sigma, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_rk, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_rk, LAPACKFullMatrixExt< Number > &matrix) const
- void print_formatted (std::ostream &out, const unsigned int precision=3, const bool scientific=true, const unsigned int width=0, const char *zero_string="0", const double denominator=1., const double threshold=0.) const
- void print_formatted_to_mat (std::ostream &out, const std::string &name, const unsigned int precision=8, const bool scientific=true, const unsigned int width=0, const char *zero_string="0", const double denominator=1., const double threshold=0.) const
- void truncate_to_rank (size_type new_rank)
- void vmult (Vector< Number > &y, const Vector< Number > &x, const bool adding=false) const
- void Tvmult (Vector< Number > &y, const Vector< Number > &x, const bool adding=false) const
- void add (RkMatrix< Number > &M, const RkMatrix< Number > &M2) const
- void add (const RkMatrix< Number > &M1)
- void add (RkMatrix< Number > &M, const RkMatrix< Number > &M2, const size_type fixed_rank_k) const
- void add (const RkMatrix< Number > &M1, const size_type fixed_rank_k)

Private Attributes

- LAPACKFullMatrixExt< Number > **A**
- LAPACKFullMatrixExt< Number > **B**
- size_type rank
- size_type formal_rank
- size_type m
- size_type n

Friends

- template<int spacedim1, typename Number1 >
  void **h_rk_mmult** (HMatrix< spacedim1, Number1 > &M1, const RkMatrix< Number1 > &M2, RkMatrix< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **h_rk_mmult_for_h_h_mmult** (HMatrix< spacedim1, Number1 > *M1, const HMatrix< spacedim1, Number1 > *M2, HMatrix< spacedim1, Number1 > *M, bool is_M1M2_last_in_M_Sigma_P)
- template<int spacedim1, typename Number1 >
  void **rk_h_mmult** (const RkMatrix< Number1 > &M1, HMatrix< spacedim1, Number1 > &M2, RkMatrix< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **rk_h_mmult_for_h_h_mmult** (const HMatrix< spacedim1, Number1 > *M1, HMatrix< spacedim1, Number1 > *M2, HMatrix< spacedim1, Number1 > *M, bool is_M1M2_last_in_M_Sigma_P)
- template<int spacedim1, typename Number1 >
  void **h_f_mmult** (HMatrix< spacedim1, Number1 > &M1, const LAPACKFullMatrixExt< Number1 > &M2, LAPACKFullMatrixExt< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **h_f_mmult** (HMatrix< spacedim1, Number1 > &M1, const LAPACKFullMatrixExt< Number1 > &M2, RkMatrix< Number1 > &M)
- template<int spacedim1, typename Number1 >
  void **h_f_mmult_for_h_h_mmult** (HMatrix< spacedim1, Number1 > *M1, const HMatrix< spacedim1, Number1 > *M2, HMatrix< spacedim1, Number1 > *M, bool is_M1M2_last_in_M_Sigma_P)
- template<int spacedim1, typename Number1 >
  void **f_h_mmult** (const LAPACKFullMatrixExt< Number1 > &M1, HMatrix< spacedim1, Number1 > &M2, LAPACKFullMatrixExt< Number1 > &M)

- template< int spacedim1, typename Number1 >
  void **f_h_mmult** (const LAPACKFullMatrixExt< Number1 > &M1, HMatrix< spacedim1, Number1 > &M2, RkMatrix< Number1 > &M)
- template< int spacedim1, typename Number1 >
  void **f_h_mmult_for_h_h_mmult** (const HMatrix< spacedim1, Number1 > *M1, HMatrix< spacedim1, Number1 > *M2, HMatrix< spacedim1, Number1 > *M, bool is_M1M2_last_in_M_Sigma_P)
- template< typename Number1 >
  void **print_rkmatrix_to_mat** (std::ostream &out, const std::string &name, const RkMatrix< Number1 > &values, const unsigned int precision, const bool scientific, const unsigned int width, const char *zero_string, const double denominator, const double threshold)

8.22.1 Member Typedef Documentation

8.22.1.1 size_type

```
template<typename Number = double>
using RkMatrix< Number >::size_type = std::make_unsigned<types::blas_int>::type
```

Declare the type for container size.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 RkMatrix() [1/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix ( )
```

Default constructor.

8.22.2.2 RkMatrix() [2/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const size_type m,
            const size_type n,
            const size_type fixed_rank_k )
```

Construct a zero-valued rank-k matrix with the specified matrix dimension and rank.

Parameters

m	
n	
fixed_rank↩_k	

If the given `fixed_rank_k` is larger than the minimum matrix dimension $\min\{m,n\}$, simply set it as this value.

References RkMatrix< Number >::formal_rank, and RkMatrix< Number >::rank.

8.22.2.3 RkMatrix() [3/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const size_type fixed_rank_k,
            LAPACKFullMatrixExt< Number > & M )
```

Construct a rank-k matrix by conversion from a full matrix `M` with rank truncation.

Note This method converts a full matrix to a rank-k matrix, which implements the operator $\mathcal{T}_r^{\mathcal{R}\leftarrow\mathcal{F}}$ in (7.2) in Hackbusch's $\mathcal{H}$-matrix book. The original full matrix `will` be modified since SVD will be applied to it.

Parameters

fixed_rank_k	
M	

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::rank, and LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.22.2.4 RkMatrix() [4/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            LAPACKFullMatrixExt< Number > & M )
```

Construct a rank-k matrix by conversion from a full matrix `M` without rank truncation.

Note The original full matrix `will` be modified since SVD will be applied to it.

Parameters

M	

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::rank, and LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.22.2.5 RkMatrix() [5/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const size_type fixed_rank_k,
            const LAPACKFullMatrixExt< Number > & M )
```

Construct a rank-k matrix by restriction to the block cluster $\tau \times \sigma$ with rank truncation from the full global matrix M defined on the complete block cluster $I \times J$.

Note This operation will not modify the full global matrix M.

Parameters

tau	
sigma	
fixed_rank←_k	
M	

Extract the data for the submatrix defined on the block cluster $\tau \times \sigma$ from the full global matrix M.

Convert the matrix block M_b in full matrix format to rank-k matrix format.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::rank, and LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.22.2.6 RkMatrix() [6/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const LAPACKFullMatrixExt< Number > & M )
```

Construct a rank-k matrix by restriction to the block cluster $\tau \times \sigma$ without rank truncation from the full global matrix M defined on the complete block cluster $I \times J$.

Note This operation will not modify the full global matrix M.

Parameters

tau	
sigma	
M	

Extract the data for the submatrix defined on the block cluster $\tau \times \sigma$ from the full global matrix M.

Convert the matrix block M_b in full matrix format to rank-k matrix format.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::rank, and LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.22.2.7 RkMatrix() [7/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const size_type fixed_rank_k,
            const LAPACKFullMatrixExt< Number > & M,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M )
```

Construct a rank-k matrix from by restriction to the block cluster $\tau \times \sigma$ with rank truncation from the full local matrix M.

Note The original full local matrix M will not be modified.

Parameters

tau	
sigma	
fixed_rank_k	
M	
row_index_global_to_local_map_for↩ _M	The map from the global row indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.
col_index_global_to_local_map_for_M	The map from the global column indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.

Extract the data for the submatrix block $b = \tau \times \sigma$ in the original matrix M.

Convert the matrix block M_b in full matrix format to rank-k format.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::rank, and LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.22.2.8 RkMatrix() [8/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
```

```
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const LAPACKFullMatrixExt< Number > & M,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M )
```

Construct a rank-k matrix from by restriction to the block cluster $\tau \times \sigma$ without rank truncation from the full local matrix M.

Note The original full local matrix M will not be modified.

Parameters

tau	
sigma	
M	
row_index_global_to_local_map_for↩ _M	The map from the global row indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.
col_index_global_to_local_map_for_M	The map from the global column indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.

Extract the data for the submatrix block $b = \tau \times \sigma$ in the original matrix M.

Convert the matrix block M_b in full matrix format to rank-k format.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::rank, and LAPACKFullMatrixExt< Number >::rank_k_decompose().

8.22.2.9 RkMatrix() [9/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const size_type fixed_rank_k,
            const RkMatrix< Number > & M )
```

Construct a rank-k matrix by restriction to the block cluster $\tau \times \sigma$ with rank truncation from the global rank-k matrix M.

Note The original rank-k global matrix M will not be modified.

Parameters

tau	
sigma	
fixed_rank↩ _k	
M	

Restrict the component matrix A of the original global rank-k matrix M to the cluster τ.

Restrict the component matrix B of the original global rank-k matrix M to the cluster σ.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::truncate_to_rank().

8.22.2.10 RkMatrix() [10/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const RkMatrix< Number > & M )
```

Construct a rank-k matrix by restriction to the block cluster $\tau \times \sigma$ without rank truncation from the global rank-k matrix M.

Note The original rank-k global matrix M will not be modified.

Parameters

tau	
sigma	
M	

Restrict the component matrix A of the original global rank-k matrix M to the cluster τ.

Restrict the component matrix B of the original global rank-k matrix M to the cluster σ.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, RkMatrix< Number >::rank, and RkMatrix< Number >::truncate_to_rank().

8.22.2.11 RkMatrix() [11/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const size_type fixed_rank_k,
            const RkMatrix< Number > & M,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M )
```

Construct a rank-k matrix by restriction to the block cluster $\tau \times \sigma$ with rank truncation from the local rank-k matrix M.

Note The original rank-k local matrix M will not be modified.

Parameters

tau	
sigma	
fixed_rank_k	
M	
row_index_global_to_local_map_for←_M	The map from the global row indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.
col_index_global_to_local_map_for_M	The map from the global column indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.

Restrict the component matrix A of the original local rank-k matrix M to the cluster τ.

Restrict the component matrix B of the original local rank-k matrix M to the cluster σ.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, and Rk←Matrix< Number >::truncate_to_rank().

8.22.2.12 RkMatrix() [12/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const RkMatrix< Number > & M,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_←
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_←
map_for_M )
```

Construct a rank-k matrix by restriction to the block cluster $\tau \times \sigma$ without rank truncation from the local rank-k matrix M. The rank of the rank-k matrix to be constructed is initialized to be the minimum of its minimum matrix dimension and the rank of M.

Note The original rank-k matrix M will not be modified.

Parameters

tau	
sigma	
M	
row_index_global_to_local_map_for←_M	The map from the global row indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.
col_index_global_to_local_map_for_M	The map from the global column indices to the local indices of the matrix associated the H-matrix when first calling this recursive function.

Restrict the component A of the original local rank-k matrix M to the cluster τ.

Restrict the component B of the original local rank-k matrix M to the cluster σ.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, RkMatrix< Number >::rank, and RkMatrix< Number >::truncate_to_rank().

8.22.2.13 RkMatrix() [13/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const LAPACKFullMatrixExt< Number > & A,
            const LAPACKFullMatrixExt< Number > & B )
```

Construct a rank-k matrix from two component matrices.

Note The formal rank of the rank-k matrix is set to the number of columns of matrix A or B. The rank of the rank-k matrix will not be calculated but temporarily set to the minimum dimension of the matrix. Hence we have actual rank $\leq$ rank $\leq$ formal rank.

Parameters

A	
B	

The formal rank of the rank-k matrix is equal to the number of columns of A or B. Hence, we make an assertion about their equality.

8.22.2.14 RkMatrix() [14/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const size_type fixed_rank_k,
            const RkMatrix< Number > & M1,
            const RkMatrix< Number > & M2,
            bool is_horizontal_split )
```

Construct a rank-k matrix M from an agglomeration of two rank-k submatrices, M_1 and M_2, which have been obtained from either horizontal splitting or vertical splitting.

Parameters

fixed_rank←_k	
M1	
M2	

Perform formatted addition of the two embedded matrices.

Perform formatted addition of the two embedded matrices.

References RkMatrix< Number >::add(), RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::rank.

8.22.2.15 RkMatrix() [15/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const size_type fixed_rank_k,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_←
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_←
map_for_M,
            const RkMatrix< Number > & M1,
            const std::vector< types::global_dof_index > & M1_tau_index_set,
            const std::vector< types::global_dof_index > & M1_sigma_index_set,
            const RkMatrix< Number > & M2,
            const std::vector< types::global_dof_index > & M2_tau_index_set,
            const std::vector< types::global_dof_index > & M2_sigma_index_set,
            bool is_horizontal_split )
```

Construct a rank-k matrix M from an agglomeration of two rank-k submatrices, M_1 and M_2, which have been obtained from either horizontal splitting or vertical splitting.

This method handles the case when the index sets of several child clusters are interwoven together into the index set of the parent cluster. This is based on the fact that during DoF support point coordinates based cluster tree partition, the continuity of the index set is not preserved.

Parameters

fixed_rank_k	
row_index_global_to_local_map_for←_M	
col_index_global_to_local_map_for_M	
M1	
M1_tau_index_set	
M1_sigma_index_set	
M2	
M2_tau_index_set	
M2_sigma_index_set	
is_horizontal_split	

Note Because the assembly of the two submatrices into the larger matrix depends on the global DoF indices, there is no need for a manual control of the assembly location and the procedures for both horizontal and vertical stacking cases are the same.

Make assertions about the sizes of row and column index global to local maps for `M`.

Perform formatted addition of the two embedded matrices.

References RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::rank.

8.22.2.16 RkMatrix() [16/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
             const size_type fixed_rank_k,
             const RkMatrix< Number > & M11,
             const RkMatrix< Number > & M12,
             const RkMatrix< Number > & M21,
             const RkMatrix< Number > & M22 )
```

Construct a rank-k matrix M from an agglomeration of four rank-k submatrices, $M_{11}, M_{12}, M_{21}, M_{22}$.

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

Note This method implements the operator $\mathcal{T}_{r,\mathrm{pairw}}^{\mathcal{R}}$ in (2.13) in Hackbusch's $\mathcal{H}$-matrix book.

Create a rank-k matrix which is an embedding of the submatrix. N.B. The embedding of a matrix does not change its rank.

At first, each matrix block is embedded into the large matrix by padding zeros to the component matrices `A` and `B` in a rank-k matrix.

Next, pairwise formatted addition will be applied successively to the four embedded matrices to achieve agglomeration.

References RkMatrix< Number >::add(), RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::rank.

8.22.2.17 RkMatrix() [17/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
             const size_type fixed_rank_k,
             const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
             const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M,
             const RkMatrix< Number > & M11,
             const std::vector< types::global_dof_index > & M11_tau_index_set,
             const std::vector< types::global_dof_index > & M11_sigma_index_set,
             const RkMatrix< Number > & M12,
             const std::vector< types::global_dof_index > & M12_tau_index_set,
             const std::vector< types::global_dof_index > & M12_sigma_index_set,
             const RkMatrix< Number > & M21,
             const std::vector< types::global_dof_index > & M21_tau_index_set,
             const std::vector< types::global_dof_index > & M21_sigma_index_set,
             const RkMatrix< Number > & M22,
             const std::vector< types::global_dof_index > & M22_tau_index_set,
             const std::vector< types::global_dof_index > & M22_sigma_index_set )
```

Construct a rank-k matrix M from an agglomeration of four rank-k submatrices, $M_{11}, M_{12}, M_{21}, M_{22}$.

$$M = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}$$

Note

1. This method implements the operator $\mathcal{T}_{r,\mathrm{pairw}}^{\mathcal{R}}$ in (2.13) in Hackbusch's $\mathcal{H}$-matrix book.
2. This method handles the case when the index sets of several child clusters are interwoven together into the index set of the parent cluster. This is based on the fact that during DoF support point coordinates based cluster tree partition, the continuity of the index set is not preserved.

Make assertions about the compatibility of number of rows and columns of the submatrices.

Make assertions about the equality of cluster index sets of submatrices.

Make assertions about the sizes of row and column index global to local maps for `M`.

Make assertions about the submatrix sizes and associated index sets.

Create a rank-k matrix which is an embedding of the submatrix. N.B. The embedding of a matrix does not change its rank.

At first, each matrix block is embedded into the large matrix by padding zeros to the component matrices `A` and `B` in a rank-k matrix.

Next, pairwise formatted addition will be applied successively to the four embedded matrices to achieve agglomeration.

References RkMatrix< Number >::add(), RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::rank.

8.22.2.18 RkMatrix() [18/18]

```
template<typename Number >
RkMatrix< Number >::RkMatrix (
            const RkMatrix< Number > & matrix )
```

Copy constructor.

8.22.3 Member Function Documentation

8.22.3.1 add() [1/4]

```
template<typename Number >
void RkMatrix< Number >::add (
            RkMatrix< Number > & M,
            const RkMatrix< Number > & M2 ) const
```

Perform the addition of two rank-k matrices $M = M_1 + M_2$ by juxtaposition without rank truncation, where M_1 is the current matrix.

Let $M_1 = A_1 B_1^T$ and $M_2 = A_2 B_2^T$. Their addition without truncation is a juxtaposition of the components of M_1 and M_2. Assume $M = AB^T$, then $A = [A_1 A_2]$ and $B = [B_1 B_2]$. Its rank $r \leq r_1 + r_2$. Also note that the value of the class member `M` may be just formal, i.e. the actual matrix rank is smaller than it.

Parameters

M	
M2	
fixed_rank←↩ _k	

Stack the components of A and B.

Referenced by RkMatrix< Number >::add(), main(), and RkMatrix< Number >::RkMatrix().

8.22.3.2 add() [2/4]

```
template<typename Number >
void RkMatrix< Number >::add (
            const RkMatrix< Number > & M1 )
```

Perform the addition of two rank-k matrices $M = M + M_1$ by juxtaposition without rank truncation, where M is the current matrix.

Parameters

M1	

Stack the components of A and B.

8.22.3.3 add() [3/4]

```
template<typename Number >
void RkMatrix< Number >::add (
            RkMatrix< Number > & M,
            const RkMatrix< Number > & M2,
            const size_type fixed_rank_k ) const
```

Perform the formatted addition of two rank-k matrices, $M = M_1 + M_2$. The resulted rank-k matrix M will be truncated to the fixed rank fixed_rank_k. Perform the addition via a simple juxtaposition of matrix components. Then rank truncation is carried out.

References RkMatrix< Number >::add(), and RkMatrix< Number >::truncate_to_rank().

8.22.3.4 add() [4/4]

```
template<typename Number >
void RkMatrix< Number >::add (
            const RkMatrix< Number > & M1,
            const size_type fixed_rank_k )
```

Perform the formatted addition of two rank-k matrices, $M = M + M_1$. `M_1` is added to the current matrix itself. The resulted rank-k matrix `M` will be truncated to the fixed rank `fixed_rank_k`. Perform the addition via a simple juxtaposition of matrix components. Then rank truncation is carried out.

References RkMatrix< Number >::add(), and RkMatrix< Number >::truncate_to_rank().

8.22.3.5 convertToFullMatrix()

```
template<typename Number >
void RkMatrix< Number >::convertToFullMatrix (
            LAPACKFullMatrixExt< Number > & matrix ) const
```

Convert an rank-k matrix to a full matrix.

Parameters

matrix	

References RkMatrix< Number >::m, and RkMatrix< Number >::n.

Referenced by main(), and RkMatrix< Number >::restrictToFullMatrix().

8.22.3.6 get_formal_rank()

```
template<typename Number >
RkMatrix< Number >::size_type RkMatrix< Number >::get_formal_rank ( ) const
```

Get the formal rank of the rank-k matrix.

Returns

References RkMatrix< Number >::formal_rank.

8.22.3.7 get_rank()

```
template<typename Number >
RkMatrix< Number >::size_type RkMatrix< Number >::get_rank ( ) const
```

Get the rank of the rank-k matrix.

Returns

References RkMatrix< Number >::rank.

8.22.3.8 print_formatted()

```
template<typename Number >
void RkMatrix< Number >::print_formatted (
            std::ostream & out,
            const unsigned int precision = 3,
            const bool scientific = true,
            const unsigned int width = 0,
            const char * zero_string = "0",
            const double denominator = 1.,
            const double threshold = 0. )  const
```

Print a RkMatrix.

Parameters

out	
precision	
scientific	
width	
zero_string	
denominator	
threshold	

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::rank.

Referenced by main().

8.22.3.9 print_formatted_to_mat()

```
template<typename Number >
void RkMatrix< Number >::print_formatted_to_mat (
            std::ostream & out,
            const std::string & name,
            const unsigned int precision = 8,
            const bool scientific = true,
            const unsigned int width = 0,
            const char * zero_string = "0",
            const double denominator = 1.,
            const double threshold = 0. )  const
```

Print a RkMatrix into Octave mat format.

Parameters

out	
name	
precision	
scientific	
width	
zero_string	
denominator	
threshold	

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::rank.

Referenced by main().

8.22.3.10 reinit()

```
template<typename Number >
void RkMatrix< Number >::reinit (
            const size_type m,
            const size_type n,
            const size_type fixed_rank_k )
```

Reinitialize a rank-k matrix with specified dimension and rank. By default, all matrix entries are initialized to zero.

Parameters

m	
n	
fixed_rank_k	
omit_zeroing_entries	

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, and RkMatrix< Number >::rank.

Referenced by h_rk_mmult(), and rk_h_mmult().

8.22.3.11 restrictToFullMatrix() [1/2]

```
template<typename Number >
void RkMatrix< Number >::restrictToFullMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            LAPACKFullMatrixExt< Number > & matrix ) const
```

Restrict a global rank-k matrix to a full matrix defined on the block cluster $\tau \times \sigma$.

References RkMatrix< Number >::convertToFullMatrix().

Referenced by HMatrix< spacedim, Number >::_distribute_sigma_r_and_f_to_leaves(), and main().

8.22.3.12 restrictToFullMatrix() [2/2]

```
template<typename Number >
void RkMatrix< Number >::restrictToFullMatrix (
            const std::vector< types::global_dof_index > & tau,
            const std::vector< types::global_dof_index > & sigma,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_rk,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_rk,
            LAPACKFullMatrixExt< Number > & matrix ) const
```

Restrict a local rank-k matrix to a full matrix defined on the block cluster $\tau \times \sigma$.

Parameters

tau	
sigma	
row_index_global_to_local_map_for↩_rk	
col_index_global_to_local_map_for_rk	
matrix	

References RkMatrix< Number >::convertToFullMatrix().

8.22.3.13 truncate_to_rank()

```
template<typename Number >
void RkMatrix< Number >::truncate_to_rank (
            size_type new_rank )
```

Truncate the RkMatrix to `new_rank`.

Note This method implements the operator $\mathcal{T}^{\mathcal{R}}_{r \leftarrow s}$ in (7.4) in Hackbusch's $\mathcal{H}$-matrix book.

Parameters

new_rank	

Work flow introduction: Use QR decomposition to perform the rank truncation.

Adopt right associativity when the matrix is long.

Adopt left associativity when the matrix is wide.

Update the formal rank.

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, RkMatrix< Number >::rank, LAPACKFullMatrixExt< Number >::reduced_svd_on_AxBT(), LAPACKFullMatrixExt< Number

>::scale_columns(), LAPACKFullMatrixExt< Number >::scale_rows(), and LAPACKFullMatrixExt< Number >↩
::transpose().

Referenced by RkMatrix< Number >::add(), main(), and RkMatrix< Number >::RkMatrix().

8.22.3.14 Tvmult()

```
template<typename Number >
void RkMatrix< Number >::Tvmult (
            Vector< Number > & y,
            const Vector< Number > & x,
            const bool adding = false ) const
```

Calculate matrix-vector multiplication as $y = BA^T x$ or $y = y + BA^T x$.

Parameters

y	
x	
adding	

The vector storing $B^T x$

References RkMatrix< Number >::formal_rank.

8.22.3.15 vmult()

```
template<typename Number >
void RkMatrix< Number >::vmult (
            Vector< Number > & y,
            const Vector< Number > & x,
            const bool adding = false ) const
```

Calculate matrix-vector multiplication as $y = AB^T x$ or $y = y + AB^T x$.

Parameters

y	
x	
adding	

The vector storing $B^T x$

References RkMatrix< Number >::formal_rank.

8.22.4 Member Data Documentation

8.22.4.1 formal_rank

```
template<typename Number = double>
size_type RkMatrix< Number >::formal_rank [private]
```

Formal matrix rank, which is equal to the number of columns of A or B, i.e. A.n or B.n.

Referenced by RkMatrix< Number >::get_formal_rank(), h_rk_mmult(), RkMatrix< Number >::print_formatted(), RkMatrix< Number >::print_formatted_to_mat(), print_rkmatrix_to_mat(), RkMatrix< Number >::reinit(), rk_h_mmult(), RkMatrix< Number >::RkMatrix(), RkMatrix< Number >::truncate_to_rank(), RkMatrix< Number >::Tvmult(), and RkMatrix< Number >::vmult().

8.22.4.2 m

```
template<typename Number = double>
size_type RkMatrix< Number >::m [private]
```

Total number of rows.

Referenced by RkMatrix< Number >::convertToFullMatrix(), h_rk_mmult(), RkMatrix< Number >::print_formatted(), RkMatrix< Number >::print_formatted_to_mat(), print_rkmatrix_to_mat(), RkMatrix< Number >::reinit(), rk_h_mmult(), RkMatrix< Number >::RkMatrix(), and RkMatrix< Number >::truncate_to_rank().

8.22.4.3 n

```
template<typename Number = double>
size_type RkMatrix< Number >::n [private]
```

Total number of columns

Referenced by RkMatrix< Number >::convertToFullMatrix(), h_rk_mmult(), RkMatrix< Number >::print_formatted(), RkMatrix< Number >::print_formatted_to_mat(), print_rkmatrix_to_mat(), RkMatrix< Number >::reinit(), rk_h_mmult(), RkMatrix< Number >::RkMatrix(), and RkMatrix< Number >::truncate_to_rank().

8.22.4.4 rank

```
template<typename Number = double>
size_type RkMatrix< Number >::rank [private]
```

Matrix rank, which is either the actual matrix rank or the minimum of m, n and A.n (or B.n). Actually, this is an upper bound of the actual matrix rank.

Referenced by RkMatrix< Number >::get_rank(), RkMatrix< Number >::print_formatted(), RkMatrix< Number >::print_formatted_to_mat(), print_rkmatrix_to_mat(), RkMatrix< Number >::reinit(), RkMatrix< Number >::RkMatrix(), and RkMatrix< Number >::truncate_to_rank().

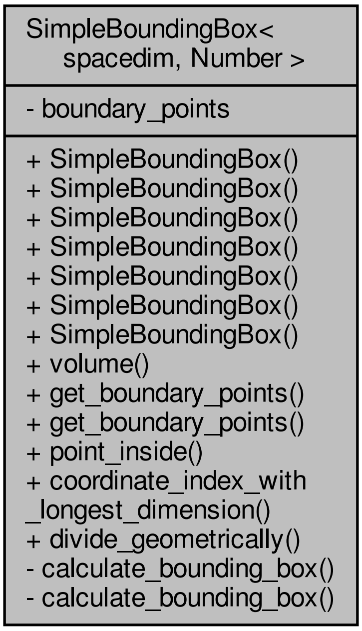
The documentation for this class was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/rkmatrix.h

8.23 SimpleBoundingBox< spacedim, Number > Class Template Reference

```
#include <simple_bounding_box.h>
```

Collaboration diagram for SimpleBoundingBox< spacedim, Number >:



Public Member Functions

- SimpleBoundingBox ()
- SimpleBoundingBox (const Point< spacedim, Number > &corner1, const Point< spacedim, Number > &corner2)
- SimpleBoundingBox (const std::pair< Point< spacedim, Number >, Point< spacedim, Number >> &point_pair)
- SimpleBoundingBox (const std::vector< Point< spacedim, Number >> &points)
- template<int dim>
  SimpleBoundingBox (const Mapping< dim, spacedim > &mapping, const DoFHandler< dim, spacedim > &dof_handler)
- SimpleBoundingBox (const std::vector< types::global_dof_index > &dof_indices, const std::vector< Point< spacedim, Number >> &all_support_points)
- SimpleBoundingBox (const SimpleBoundingBox< spacedim, Number > &bbox)
- Number volume () const
- std::pair< Point< spacedim, Number >, Point< spacedim, Number > > & get_boundary_points ()
- const std::pair< Point< spacedim, Number >, Point< spacedim, Number > > & get_boundary_points () const
- bool point_inside (const Point< spacedim, Number > &p) const
- unsigned int coordinate_index_with_longest_dimension () const
- std::pair< SimpleBoundingBox< spacedim, Number >, SimpleBoundingBox< spacedim, Number > > divide_geometrically () const

Private Member Functions

- void calculate_bounding_box (const std::vector< Point< spacedim, Number >> &points)
- void calculate_bounding_box (const std::vector< types::global_dof_index > &dof_indices, const std::vector< Point< spacedim, Number >> &all_support_points)

Private Attributes

- std::pair< Point< spacedim, Number >, Point< spacedim, Number > > boundary_points

Friends

- template<int spacedim1, typename Number1 >
  std::ostream & **operator<<** (std::ostream &out, const SimpleBoundingBox< spacedim1, Number1 > &bbox)

8.23.1 Detailed Description

template<int spacedim, typename Number = double>
class SimpleBoundingBox< spacedim, Number >

Class implementing a simple axis-parallel bounding box. N.B. This class only has the `spacedim` template argument but without the `dim` template argument, which means an axis-parallel bounding box always has the same dimension as the specified space dimension. Therefore, the bounding box for a 2D surface in 3D space is still a 3D box.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 SimpleBoundingBox() [1/7]

```
template<int spacedim, typename Number >
SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox ( )
```

Default constructor with two boundary points being both zeros.

8.23.2.2 SimpleBoundingBox() [2/7]

```
template<int spacedim, typename Number >
SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox (
            const Point< spacedim, Number > & corner1,
            const Point< spacedim, Number > & corner2 )
```

Constructor from two corner points.

8.23.2.3 SimpleBoundingBox() [3/7]

```
template<int spacedim, typename Number >
SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox (
            const std::pair< Point< spacedim, Number >, Point< spacedim, Number >> & point←
_pair )
```

Constructor from a pair of corner points.

Parameters

point_pair	

8.23.2.4 SimpleBoundingBox() [4/7]

```
template<int spacedim, typename Number >
SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox (
            const std::vector< Point< spacedim, Number >> & points )
```

Constructor from a vector of points.

References SimpleBoundingBox< spacedim, Number >::calculate_bounding_box().

8.23.2.5 SimpleBoundingBox() [5/7]

```
template<int spacedim, typename Number >
template<int dim>
SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox (
            const Mapping< dim, spacedim > & mapping,
            const DoFHandler< dim, spacedim > & dof_handler )
```

Constructor from a `Mapping` and a `DoFHandler`. N.B. This is a template constructor with the argument dim as manifold dimension. The resulted bounding box contains the support points associated with all DoFs within the DoFHandler.

References SimpleBoundingBox< spacedim, Number >::calculate_bounding_box().

8.23.2.6 SimpleBoundingBox() [6/7]

```
template<int spacedim, typename Number >
SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox (
            const std::vector< types::global_dof_index > & dof_indices,
            const std::vector< Point< spacedim, Number >> & all_support_points )
```

Constructor from a specified DoF index set. The support point coordinates for all DoFs have been placed into `all_support_points`.

Parameters

dof_indices	The DoF index set, for which the bounding box is to be generated.
all_support_points	The const reference to the list of all support points associated with a DoFHandler.

References SimpleBoundingBox< spacedim, Number >::calculate_bounding_box().

8.23.2.7 SimpleBoundingBox() [7/7]

```
template<int spacedim, typename Number >
SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox (
            const SimpleBoundingBox< spacedim, Number > & bbox )
```

Copy constructor

8.23.3 Member Function Documentation

8.23.3.1 calculate_bounding_box() [1/2]

```
template<int spacedim, typename Number >
void SimpleBoundingBox< spacedim, Number >::calculate_bounding_box (
            const std::vector< Point< spacedim, Number >> & points )  [private]
```

Calculate the bounding box from a list of points. Initialize the two boundary points to be the first point in the list.

Calculate the minimum and the maximum coordinate for each dimension.

References SimpleBoundingBox< spacedim, Number >::boundary_points.

Referenced by SimpleBoundingBox< spacedim, Number >::SimpleBoundingBox().

8.23.3.2 calculate_bounding_box() [2/2]

```
template<int spacedim, typename Number >
void SimpleBoundingBox< spacedim, Number >::calculate_bounding_box (
            const std::vector< types::global_dof_index > & dof_indices,
            const std::vector< Point< spacedim, Number >> & all_support_points )  [private]
```

Calculate the bounding box from a list of DoF indices.

Parameters

dof_indices	The DoF index set.
all_support_points	The const reference to the list of all support points associated with a DoFHandler.

Initialize the two boundary points to be the point associated with the first DoF index.

Calculate the minimum and the maximum coordinate for each dimension.

References SimpleBoundingBox< spacedim, Number >::boundary_points.

8.23.3.3 coordinate_index_with_longest_dimension()

```
template<int spacedim, typename Number >
unsigned int SimpleBoundingBox< spacedim, Number >::coordinate_index_with_longest_dimension (
) const
```

Get the index to the coordinate component, which has the longest dimension.

Returns

Index to the coordinate component, which should be in the range $[0, \mathrm{spacedim})$.

References SimpleBoundingBox< spacedim, Number >::boundary_points.

Referenced by SimpleBoundingBox< spacedim, Number >::divide_geometrically().

8.23.3.4 divide_geometrically()

```
template<int spacedim, typename Number >
std::pair< SimpleBoundingBox< spacedim, Number >, SimpleBoundingBox< spacedim, Number > >
SimpleBoundingBox< spacedim, Number >::divide_geometrically ( ) const
```

Bisect the bounding box along the longest coordinate direction.

Returns

Calculate the coordinate index which has the longest dimension.

Modify the `coordinate_index` 'th coordinate of the top corner point in the first box.

Modify the `coordinate_index` 'th coordinate of the bottom corner point in the second box.

References SimpleBoundingBox< spacedim, Number >::coordinate_index_with_longest_dimension(), and SimpleBoundingBox< spacedim, Number >::get_boundary_points().

8.23.3.5 get_boundary_points() [1/2]

```
template<int spacedim, typename Number >
std::pair< Point< spacedim, Number >, Point< spacedim, Number > > & SimpleBoundingBox< spacedim,
Number >::get_boundary_points ( )
```

Get the two corner points of the bounding box (mutable version).

Returns

The reference to the pair of corner points. The point with a smaller coordinate in each dimension is the first.

References SimpleBoundingBox< spacedim, Number >::boundary_points.

Referenced by SimpleBoundingBox< spacedim, Number >::divide_geometrically().

8.23.3.6 get_boundary_points() [2/2]

```
template<int spacedim, typename Number >
const std::pair< Point< spacedim, Number >, Point< spacedim, Number > > & SimpleBoundingBox<
spacedim, Number >::get_boundary_points ( ) const
```

Get the two corner points of the bounding box (const version).

Returns

The const reference to the pair of corner points. The point with a smaller coordinate in each dimension is the first.

References SimpleBoundingBox< spacedim, Number >::boundary_points.

8.23.3.7 point_inside()

```
template<int spacedim, typename Number >
bool SimpleBoundingBox< spacedim, Number >::point_inside (
            const Point< spacedim, Number > & p ) const
```

Determine if a given point lies within the bounding box. Make a predicate on if the point lies outside the bounding box.

References SimpleBoundingBox< spacedim, Number >::boundary_points.

8.23.3.8 volume()

```
template<int spacedim, typename Number >
Number SimpleBoundingBox< spacedim, Number >::volume ( ) const
```

Calculate the volume of the bounding box.

Returns

Volume of the bounding box.

References SimpleBoundingBox< spacedim, Number >::boundary_points.

8.23.4 Member Data Documentation

8.23.4.1 boundary_points

```
template<int spacedim, typename Number = double>
std::pair<Point<spacedim, Number>, Point<spacedim, Number> > SimpleBoundingBox< spacedim,
Number >::boundary_points [private]
```

Two corner points of the bounding box. The point in the pair is the bottom corner, i.e. it has a smaller component coordinate in each dimension; while the second point is the top corner.

Referenced by SimpleBoundingBox< spacedim, Number >::calculate_bounding_box(), SimpleBoundingBox< spacedim, Number >::coordinate_index_with_longest_dimension(), SimpleBoundingBox< spacedim, Number >↵ ::get_boundary_points(), SimpleBoundingBox< spacedim, Number >::point_inside(), and SimpleBoundingBox< spacedim, Number >::volume().

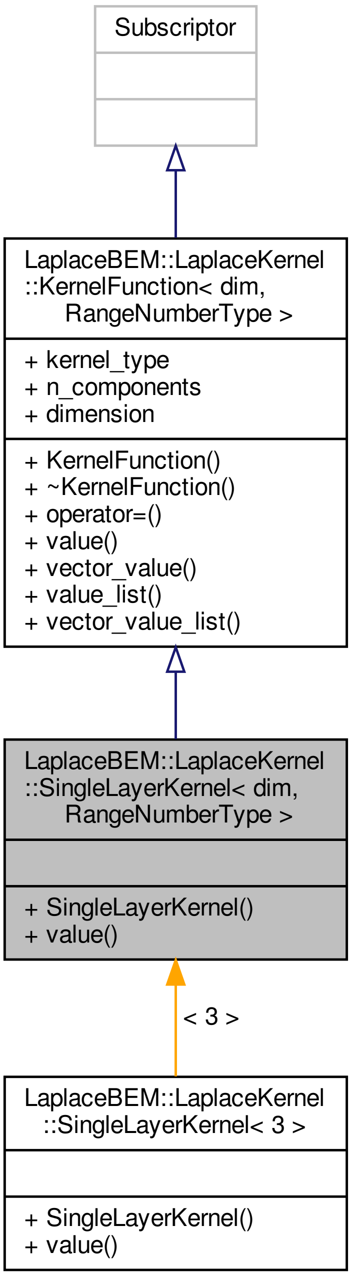
The documentation for this class was generated from the following file:

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/simple_bounding_box.h

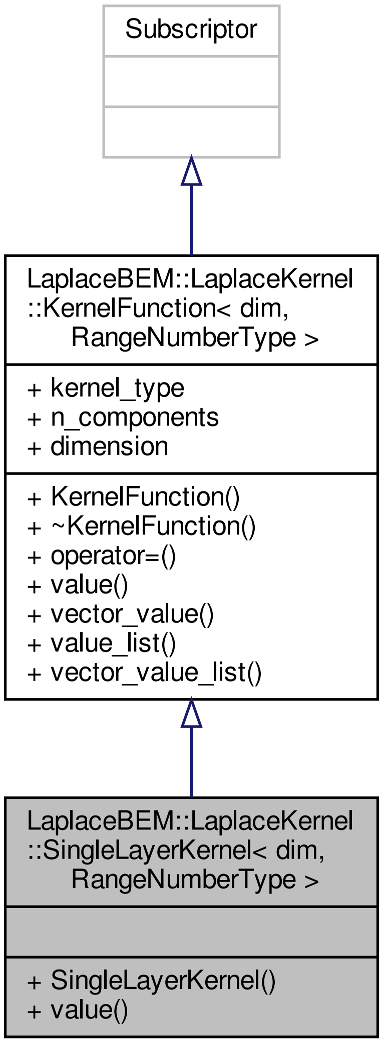
8.24 LaplaceBEM::LaplaceKernel::SingleLayerKernel< dim, RangeNumberType > Class Template Reference

```
#include <laplace_bem.h>
```

Inheritance diagram for LaplaceBEM::LaplaceKernel::SingleLayerKernel< dim, RangeNumberType >:



Collaboration diagram for LaplaceBEM::LaplaceKernel::SingleLayerKernel< dim, RangeNumberType >:



Public Member Functions

- virtual RangeNumberType **value** (const Point< dim > &x, const Point< dim > &y, const Tensor< 1, dim > &nx, const Tensor< 1, dim > &ny, const unsigned int component=0) const override

Additional Inherited Members

8.24.1 Detailed Description

template< int dim, typename RangeNumberType = double>
class LaplaceBEM::LaplaceKernel::SingleLayerKernel< dim, RangeNumberType >

Single layer kernel.

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

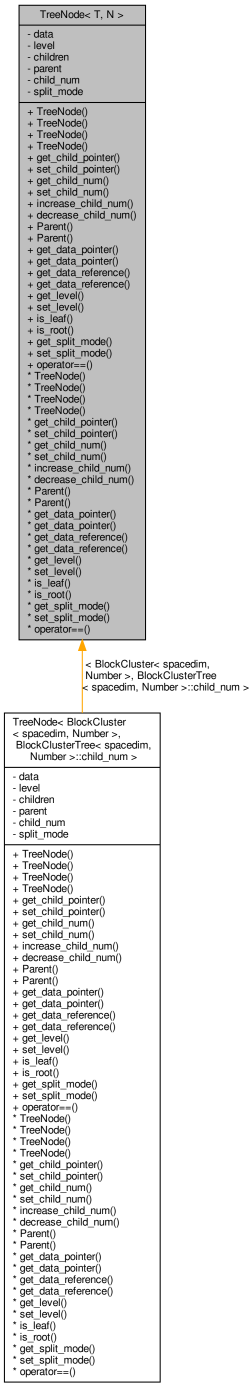
- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h

8.25 TreeNode< T, N > Class Template Reference

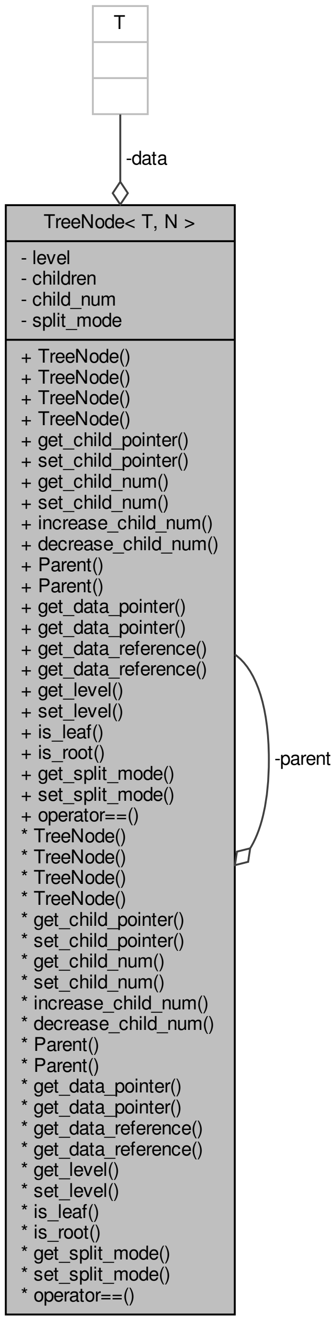
Class for general tree node.

`#include <tree.h>`

Inheritance diagram for TreeNode< T, N >:



Collaboration diagram for TreeNode< T, N >:



Public Member Functions

- TreeNode ()
- TreeNode (const T &data)

- TreeNode (const T &data, unsigned int level, const std::array< TreeNode ∗, N > &children, TreeNode ∗parent=nullptr, TreeNodeSplitMode split_mode=UnsplitMode)
- TreeNode (const TreeNode &node)
- TreeNode ∗ get_child_pointer (std::size_t index) const
- void set_child_pointer (std::size_t index, const TreeNode ∗pointer)
- unsigned int get_child_num () const
- void set_child_num (const unsigned int child_num)
- void increase_child_num (const unsigned int incr_num=1)
- void decrease_child_num (const unsigned int decr_num=1)
- TreeNode ∗ Parent (void) const
- void Parent (const TreeNode ∗node_pointer)
- T ∗ get_data_pointer ()
- const T ∗ get_data_pointer () const
- T & get_data_reference ()
- const T & get_data_reference () const
- unsigned int get_level () const
- void set_level (const unsigned int level)
- bool is_leaf () const
- bool is_root () const
- TreeNodeSplitMode get_split_mode () const
- void set_split_mode (TreeNodeSplitMode split_mode)
- bool operator== (const TreeNode< T, N > &node) const

Private Attributes

- T **data**
- unsigned int **level**
- std::array< TreeNode ∗, N > **children**
- TreeNode ∗ **parent**
- unsigned int child_num
- TreeNodeSplitMode **split_mode**

8.25.1 Detailed Description

template<typename T, std::size_t N>
class TreeNode< T, N >

Class for general tree node.

This general tree node can contains any but *fixed* number of children. `T` is the type of the data held by the node. `N` is the number of children belong to the node.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 TreeNode() [1/4]

```
template<typename T , std::size_t N>
TreeNode< T, N >::TreeNode ( )
```

Default constructor

8.25.2.2 TreeNode() [2/4]

```
template<typename T, std::size_t N>
TreeNode< T, N >::TreeNode (
            const T & data )
```

Constructor from the given data without children.

8.25.2.3 TreeNode() [3/4]

```
template<typename T, std::size_t N>
TreeNode< T, N >::TreeNode (
            const T & data,
            unsigned int level,
            const std::array< TreeNode< T, N > *, N > & children,
            TreeNode< T, N > * parent = nullptr,
            TreeNodeSplitMode split_mode = UnsplitMode )
```

Constructor from the given data.

N.B. The number of children of the parent node will automatically be incremented, because the current node is associated with this parent. Count the total number of nonempty children.

When the tree node is split into four children, this is a cross splitting mode. Otherwise, `split_mode` is temporarily kept at `UnsplitMode`, which will further be set from outside.

Increment the `child_num` of the parent node.

8.25.2.4 TreeNode() [4/4]

```
template<typename T, std::size_t N>
TreeNode< T, N >::TreeNode (
            const TreeNode< T, N > & node )
```

Copy constructor.

8.25.3 Member Function Documentation

8.25.3.1 decrease_child_num()

```
template<typename T , std::size_t N>
void TreeNode< T, N >::decrease_child_num (
            const unsigned int decr_num = 1 )
```

Decrease the total number of children.

Returns

8.25.3.2 get_child_num()

```
template<typename T , std::size_t N>
unsigned int TreeNode< T, N >::get_child_num ( ) const
```

Get the total number of nonempty children.

Referenced by CopyTree(), GetTreeLeaves(), InitAndCreateHMatrixChildren(), and PrintTreeNode().

8.25.3.3 get_child_pointer()

```
template<typename T , std::size_t N>
TreeNode< T, N > * TreeNode< T, N >::get_child_pointer (
            std::size_t index ) const
```

Get the pointer to the `index'th` child.

Parameters

index	Index of the child

Returns

Referenced by calc_depth(), CopyTree(), CountTreeNodes(), GetTreeLeaves(), InitAndCreateHMatrixChildren(), Postorder(), and Preorder().

8.25.3.4 get_data_pointer() [1/2]

```
template<typename T , std::size_t N>
T * TreeNode< T, N >::get_data_pointer ( )
```

Get the pointer to the node data.

Referenced by BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product_from_block←_cluster_node(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product_from_block←_cluster_node(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product_from_block_←cluster_node_N(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product_from_block_←cluster_node_Nstar(), BlockClusterTree< spacedim, Number >::partition_from_block_cluster_node(), and Block←ClusterTree< spacedim, Number >::partition_tensor_product_from_block_cluster_node().

8.25.3.5 get_data_pointer() [2/2]

```
template<typename T , std::size_t N>
const T * TreeNode< T, N >::get_data_pointer ( ) const
```

Get the pointer to the node data (const version).

8.25.3.6 get_data_reference() [1/2]

```
template<typename T , std::size_t N>
T & TreeNode< T, N >::get_data_reference ( )
```

Get the reference to the node data.

Referenced by CopyTree(), BlockClusterTree< spacedim, Number >::extend_finer_than_partition(), BlockCluster↩ Tree< spacedim, Number >::extend_to_finer_partition(), BlockClusterTree< spacedim, Number >::find_leaf↩ _bc_node_not_in_partition(), BlockClusterTree< spacedim, Number >::find_leaf_bc_node_not_subset_of_bc_↩ nodes_in_partition(), InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::mmult(), and PrintTree↩ Node().

8.25.3.7 get_data_reference() [2/2]

```
template<typename T , std::size_t N>
const T & TreeNode< T, N >::get_data_reference ( ) const
```

Get the reference to the node data (const version).

8.25.3.8 get_level()

```
template<typename T , std::size_t N>
unsigned int TreeNode< T, N >::get_level ( ) const
```

Get the level of the node in the tree.

Referenced by CopyTree(), and PrintTreeNode().

8.25.3.9 get_split_mode()

```
template<typename T , std::size_t N>
TreeNodeSplitMode TreeNode< T, N >::get_split_mode ( ) const
```

Return the split mode of the tree node.

Returns

Referenced by CopyTree(), and PrintTreeNode().

8.25.3.10 increase_child_num()

```
template<typename T , std::size_t N>
void TreeNode< T, N >::increase_child_num (
            const unsigned int incr_num = 1 )
```

Increase the total number of children.

Referenced by CopyTree().

8.25.3.11 is_leaf()

```
template<typename T , std::size_t N>
bool TreeNode< T, N >::is_leaf ( ) const
```

Return whether the current tree node is a leaf.

Returns

8.25.3.12 is_root()

```
template<typename T , std::size_t N>
bool TreeNode< T, N >::is_root ( ) const
```

Return whether the current tree node is the root.

Returns

8.25.3.13 operator==()

```
template<typename T, std::size_t N>
bool TreeNode< T, N >::operator== (
            const TreeNode< T, N > & node ) const
```

Check the equality of two tree nodes by comparing the contained data.

Parameters

node	

Returns

8.25.3.14 Parent() [1/2]

```
template<typename T , std::size_t N>
TreeNode< T, N > * TreeNode< T, N >::Parent (
            void ) const
```

Get the pointer to the parent tree node.

Returns

Referenced by CopyTree(), and PrintTreeNode().

8.25.3.15 Parent() [2/2]

```
template<typename T , std::size_t N>
void TreeNode< T, N >::Parent (
            const TreeNode< T, N > * node_pointer )
```

Set the pointer to the parent tree node.

Parameters

node_pointer	

8.25.3.16 set_child_num()

```
template<typename T , std::size_t N>
void TreeNode< T, N >::set_child_num (
            const unsigned int child_num )
```

Set the total number of nonempty children.

Parameters

child_num	

8.25.3.17 set_child_pointer()

```
template<typename T , std::size_t N>
void TreeNode< T, N >::set_child_pointer (
            std::size_t index,
            const TreeNode< T, N > * pointer )
```

Set the pointer to the `index'th` child.

N.B. The const pointer type in the argument will be converted to non-const pointer type. In this way, even a const node can be added to the tree.

Parameters

index	Index of the child
pointer	New pointer value

Referenced by CopyTree().

8.25.3.18 set_level()

```
template<typename T , std::size_t N>
void TreeNode< T, N >::set_level (
            const unsigned int level )
```

Set the level of the node.

Parameters

level	

8.25.3.19 set_split_mode()

```
template<typename T , std::size_t N>
void TreeNode< T, N >::set_split_mode (
            TreeNodeSplitMode split_mode )
```

Set the split mode of the tree node.

Parameters

split_mode	

Referenced by MakeIntExampleTree(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor↩_product_from_block_cluster_node(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_↩product_from_block_cluster_node(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_↩

product_from_block_cluster_node_N(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor↩
_product_from_block_cluster_node_Nstar(), BlockClusterTree< spacedim, Number >::partition_from_block_↩
cluster_node(), and BlockClusterTree< spacedim, Number >::partition_tensor_product_from_block_cluster_↩
node().

8.25.4 Member Data Documentation

8.25.4.1 child_num

```
template<typename T, std::size_t N>
unsigned int TreeNode< T, N >::child_num  [private]
```

Total number of nonempty children.

Referenced by TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child↩
_num >::decrease_child_num(), TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num >::get_child_num(), TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num >::increase_child_num(), TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num >::is_leaf(), TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num >::set_child_num(), and TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num >::TreeNode().

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

- /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/tree.h

Chapter 9

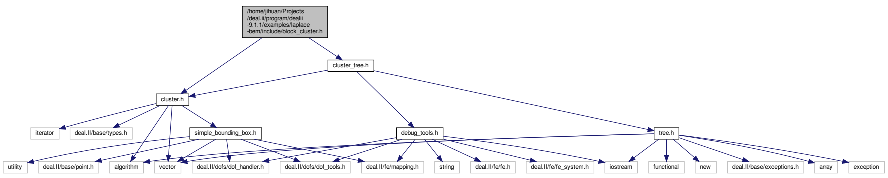
File Documentation

9.1 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/block↩ _cluster.h File Reference

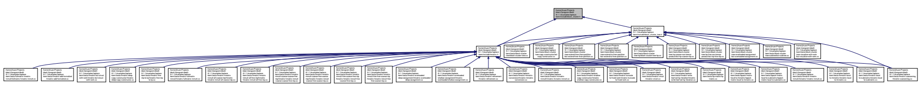
Implementation of the class BlockCluster.

```
#include "cluster.h"
#include "cluster_tree.h"
```

Include dependency graph for block_cluster.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BlockCluster< spacedim, Number >

    Class for block cluster.

Functions

- template<int spacedim, typename Number >
  std::ostream & **operator**<< (std::ostream &out, const BlockCluster< spacedim, Number > &block_cluster)
- template<int spacedim, typename Number >
  bool operator== (const BlockCluster< spacedim, Number > &block_cluster1, const BlockCluster< spacedim, Number > &block_cluster2)
- template<int spacedim, typename Number >
  bool is_equal (const BlockCluster< spacedim, Number > &block_cluster1, const BlockCluster< spacedim, Number > &block_cluster2)

9.1.1 Detailed Description

Implementation of the class BlockCluster.

Date

2021-04-20

Author

Jihuan Tian

9.1.2 Function Documentation

9.1.2.1 is_equal()

```
template<int spacedim, typename Number >
bool is_equal (
            const BlockCluster< spacedim, Number > & block_cluster1,
            const BlockCluster< spacedim, Number > & block_cluster2 )
```

Check the equality of two block cluster by comparing the contents of block cluster's index sets. Compared to `BlockCluster<spacedim`, Number>operator==, this can be considered as deep comparison.

Parameters

block_cluster1	
block_cluster2	

Returns

Referenced by BlockClusterTree< spacedim, Number >::find_leaf_bc_node_not_in_partition().

9.1.2.2 operator==()

```
template<int spacedim, typename Number >
bool operator== (
            const BlockCluster< spacedim, Number > & block_cluster1,
            const BlockCluster< spacedim, Number > & block_cluster2 )
```

Check the equality of two block clusters by shallow comparison.

The comparison is based on the pointer addresses to the tau cluster node and the sigma cluster node. Therefore, this is a "shallow" comparison for performance issue.

Note This method is valid in the following two cases.

1. The two block clusters to be compared belong a same block cluster tree. In this scenario, because in either cluster tree, $T(I)$ or $T(J)$, comprising the block cluster tree, the cluster nodes contained are created on the heap, hence each of them has an address in the memory different from all the others. Therefore, the equality of two block clusters, i.e. the equality of the index sets in the τ cluster nodes and the equality of the index sets in the σ cluster nodes, is equivalent to the equality of the pointer addresses of τ cluster nodes and the equality of the pointer addresses of σ cluster nodes.
2. The two block clusters to be compared belong to two different block cluster tress, each of which is built from the two cluster trees $T(I)$ and $T(J)$.

Parameters

block_cluster1	
block_cluster2	

Returns

References BlockCluster< spacedim, Number >::sigma_node, and BlockCluster< spacedim, Number >::tau_↩ node.

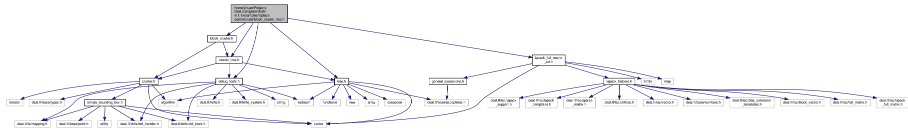
9.2 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/block↩ _cluster_tree.h File Reference

Implementation of the class BlockClusterTree.

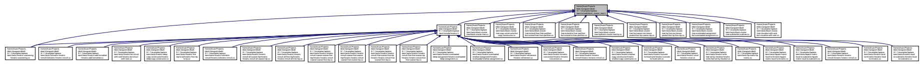
```
#include "block_cluster.h"
#include "cluster_tree.h"
#include "debug_tools.h"
#include "lapack_full_matrix_ext.h"
```

`#include "tree.h"`

Include dependency graph for block_cluster_tree.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BlockClusterTree< spacedim, Number >

  Class for block cluster tree.

Functions

- template<int spacedim, typename Number >
  std::ostream & **operator**<< (std::ostream &out, const BlockClusterTree< spacedim, Number > &block_cluster_tree)
- template<int spacedim, typename Number >
  void split_block_cluster_node (TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > *bc_node, BlockClusterTree< spacedim, Number > &bc_tree, const TreeNodeSplitMode split_mode, const bool if_add_child_nodes_to_leaf_set=true)
- template<int spacedim, typename Number >
  TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > * find_bc_node_in_partition_intersect_current_bc_node (TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > *current_bc_node, const std::vector< TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > *> &partition)
- template<int spacedim, typename Number >
  TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > * **find_bc_node_in_partition_proper_subset_of_current_bc_node** (TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > *current_bc_node, const std::vector< TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number >::child_num > *> &partition)

9.2.1 Detailed Description

Implementation of the class BlockClusterTree.

Date

2021-04-20

Author

Jihuan Tian

9.2.2 Function Documentation

9.2.2.1 find_bc_node_in_partition_intersect_current_bc_node()

```
template<int spacedim, typename Number >
TreeNode<BlockCluster<spacedim, Number>, BlockClusterTree<spacedim, Number>::child_num>*
find_bc_node_in_partition_intersect_current_bc_node (
            TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number
>::child_num > * current_bc_node,
            const std::vector< TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree<
spacedim, Number >::child_num > *> & partition )
```

Find a block cluster node in the given partition which has nonempty intersection with the current block cluster node.

Returns

References BlockClusterTree< spacedim, Number >::partition().

9.2.2.2 split_block_cluster_node()

```
template<int spacedim, typename Number >
void split_block_cluster_node (
            TreeNode< BlockCluster< spacedim, Number >, BlockClusterTree< spacedim, Number
>::child_num > * bc_node,
            BlockClusterTree< spacedim, Number > & bc_tree,
            const TreeNodeSplitMode split_mode,
            const bool if_add_child_nodes_to_leaf_set = true )
```

Split a block cluster node in a block cluster tree with the given split mode.

After the splitting, the total number of block cluster nodes in the block cluster tree will be updated, while the leaf set of the block cluster tree will not be rebuilt.

Parameters

bc_node	
bc_tree	
split_mode	

Work flow

- Horizontal split mode

1. Set the split mode of the current block cluster node.
2. Create a new block cluster node by constructing its index set as $\tau^* \times \sigma$ with $\tau^* \in S(\tau)$. Its parent node is set to the current block cluster node.
3. Check if the block cluster node is small.

4. Append this new node as a child of the current block cluster node.
5. Add this new node to the leaf set if necessary.
6. Increase the total number of nodes in the block cluster tree.

- Vertical split mode
  1. Set the split mode of the current block cluster node.
  2. Create a new block cluster node by constructing its index set as $\tau \times \sigma^*$ with $\sigma^* \in S(\sigma)$. Its parent node is set to the current block cluster node.
  3. Check if the block cluster node is small.
  4. Append this new node as a child of the current block cluster node.
  5. Add this new node to the leaf set if necessary.
  6. Increase the total number of nodes in the block cluster tree.
- Cross split mode
  1. Set the split mode of the current block cluster node.
  2. Create a new block cluster node by constructing its index set as $\tau^* \times \sigma^*$ with $\tau^* \in S(\tau)$ and $\sigma^* \in S(\sigma)$. Its parent node is set to the current block cluster node.
  3. Check if the block cluster node is small.
  4. Append this new node as one of the children of the current block cluster node.
  5. Add this new node to the leaf set if necessary.
  6. Increase the total number of nodes in the block cluster tree.

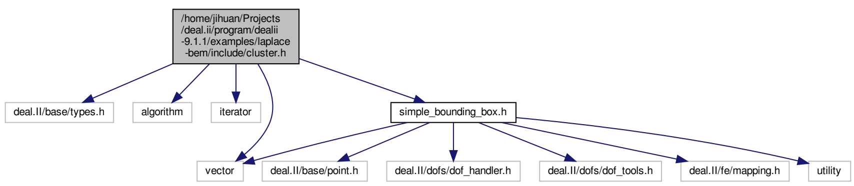
Referenced by HMatrix< spacedim, Number >::h_h_mmult_cross_split(), HMatrix< spacedim, Number >::h_h_mmult_horizontal_split(), and HMatrix< spacedim, Number >::h_h_mmult_vertical_split().

9.3 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/cluster.h File Reference

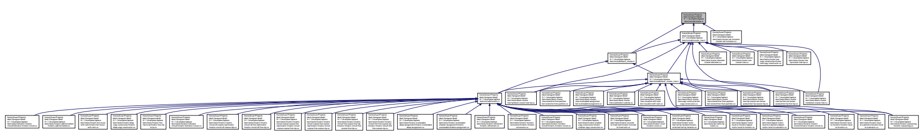
Implementation of the class Cluster.

```
#include <deal.II/base/types.h>
#include <algorithm>
#include <iterator>
#include <vector>
#include "simple_bounding_box.h"
```

Include dependency graph for cluster.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Cluster< spacedim, Number >

    Class for an index cluster.

Functions

- template< int dim, int spacedim, typename Number = double>
  void map_dofs_to_average_cell_size (const DoFHandler< dim, spacedim > &dof_handler, std::vector< Number > &dof_average_cell_size)
- template< typename DoFHandlerType , typename Number = double>
  void map_dofs_to_max_cell_size (const DoFHandlerType &dof_handler, std::vector< Number > &dof_max_cell_size)
- template< typename DoFHandlerType , typename Number = double>
  void map_dofs_to_min_cell_size (const DoFHandlerType &dof_handler, std::vector< Number > &dof_min_cell_size)
- template< int spacedim, typename Number >
  std::ostream & operator<< (std::ostream &out, const Cluster< spacedim, Number > &cluster)
- template< int spacedim, typename Number = double>
  Number calc_cluster_distance (const Cluster< spacedim, Number > &cluster1, const Cluster< spacedim, Number > &cluster2, const std::vector< Point< spacedim, Number >> &all_support_points)
- template< int spacedim, typename Number = double>
  Number calc_cluster_distance (const Cluster< spacedim, Number > &cluster1, const Cluster< spacedim, Number > &cluster2, const std::vector< Point< spacedim, Number >> &all_support_points, const std::vector< Number > &cell_size_at_dofs)
- template< int spacedim, typename Number >
  bool **operator==** (const Cluster< spacedim, Number > &cluster1, const Cluster< spacedim, Number > &cluster2)

9.3.1 Detailed Description

Implementation of the class Cluster.

Date

2021-04-18

Author

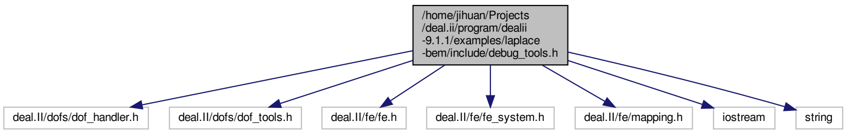
Jihuan Tian

9.4 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/debug_tools.h File Reference

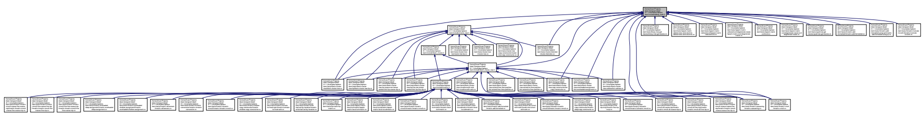
This file includes a bunch of helper functions for printing out and visualizing information about grid, DoFs, map, etc.

```
#include <deal.II/dofs/dof_handler.h>
#include <deal.II/dofs/dof_tools.h>
#include <deal.II/fe/fe.h>
#include <deal.II/fe/fe_system.h>
#include <deal.II/fe/mapping.h>
#include <iostream>
#include <string>
```

Include dependency graph for debug_tools.h:



This graph shows which files directly or indirectly include this file:



Functions

- template<typename VectorType >
  void **print_vector_values** (std::ostream &out, const VectorType &values, const std::string &sep=std::string(","), bool has_newline=true)
- template<typename VectorType >
  void **print_vector_indices** (std::ostream &out, const VectorType &values, const std::string &sep, bool index_starting_from_zero, bool has_newline=true)
- template<typename Number >
  void **print_scalar_to_mat** (std::ostream &out, const std::string &name, const Number value)
- template<typename VectorType >
  void **print_vector_to_mat** (std::ostream &out, const std::string &name, const VectorType &values)
- template<typename MatrixType >
  void **print_matrix_to_mat** (std::ostream &out, const std::string &name, const MatrixType &values, const unsigned int precision=8, const bool scientific=true, const unsigned int width=0, const char *zero_string="0", const double denominator=1., const double threshold=0.)
- template<typename node_pointer_type >
  void **print_vector_of_tree_node_pointer_values** (std::ostream &out, const std::vector< node_pointer_type > &tree_node_pointers, const std::string &sep=std::string(","))
- template<int dim, int spacedim>
  void print_support_point_info (const FiniteElement< dim, spacedim > &fe, const Mapping< dim, spacedim > &mapping, const DoFHandler< dim, spacedim > &dof_handler, const std::string &base_name)

  Generate a table of DoF indices associated with each support point.
- template<int dim, int spacedim>
  void print_support_point_info (const FESystem< dim, spacedim > &fe_system, const Mapping< dim, spacedim > &mapping, const DoFHandler< dim, spacedim > &dof_handler, const std::string &base_name)

  Generate a table of DoF indices associated with each support point.

9.4.1 Detailed Description

This file includes a bunch of helper functions for printing out and visualizing information about grid, DoFs, map, etc.

Date

2021-04-25

Author

Jihuan Tian

9.4.2 Function Documentation

9.4.2.1 print_support_point_info() [1/2]

```
template<int dim, int spacedim>
void print_support_point_info (
            const FiniteElement< dim, spacedim > & fe,
            const Mapping< dim, spacedim > & mapping,
            const DoFHandler< dim, spacedim > & dof_handler,
            const std::string & base_name )
```

Generate a table of DoF indices associated with each support point.

The information in the table is defined in the given Mapping and DoFHandler objects, which can be then visualized in Gnuplot by executing the following command.

1. For spacedim=2:

```
plot "./data_file.gpl" using 1:2:3 with labels offset 1,1 point pt 1 lc rgb \
"red" notitle
```

2. For spacedim=3:

```
splot "./data_file.gpl" using 1:2:3:4 with labels offset 1,1 point pt 1 lc \
rgb "red"
```

Parameters

fe_system	The given FiniteElement object will be checked if it has support points.
mapping	
dof_handler	
base_name	

Allocate memory for the vector storing support points.

Get the list of support point coordinates for all DoFs. The DoFs are in the default numbering starting from 0.

Write the table of DoF indices for each support point into file.

Referenced by main().

9.4.2.2 print_support_point_info() [2/2]

```
template<int dim, int spacedim>
void print_support_point_info (
            const FESystem< dim, spacedim > & fe_system,
            const Mapping< dim, spacedim > & mapping,
            const DoFHandler< dim, spacedim > & dof_handler,
            const std::string & base_name )
```

Generate a table of DoF indices associated with each support point.

The information in the table is defined in the given Mapping and DoFHandler objects, which can be then visualized in Gnuplot by executing the following command.

1. For spacedim=2:

```
plot "./data_file.gpl" using 1:2:3 with labels offset 1,1 point pt 1 lc rgb \
"red" notitle
```

2. For spacedim=3:

```
splot "./data_file.gpl" using 1:2:3:4 with labels offset 1,1 point pt 1 lc \
rgb "red"
```

Parameters

fe_system	The given FESystem object will be checked if it has support points.
mapping	
dof_handler	
base_name	

Allocate memory for the vector storing support points.

Get the list of support point coordinates for all DoFs. The DoFs are in the default numbering starting from 0.

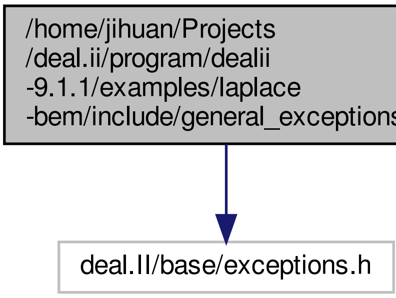
Write the table of DoF indices for each support point into file.

9.5 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/general_exceptions.h File Reference

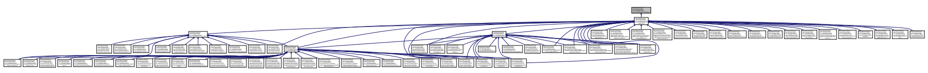
Definition of self-defined general exceptions.

```
#include <deal.II/base/exceptions.h>
```

Include dependency graph for general_exceptions.h:



This graph shows which files directly or indirectly include this file:



Functions

- **DeclException3** (ExcOpenIntervalRange, double, double, double,<< arg1<< " is not in the range ("<< arg2<< ", "<< arg3<< ").")
- **DeclException3** (ExcLeftOpenIntervalRange, double, double, double,<< arg1<< " is not in the range ("<< arg2<< ", "<< arg3<< "].")
- **DeclException3** (ExcRightOpenIntervalRange, double, double, double,<< arg1<< " is not in the range ["<< arg2<< ", "<< arg3<< ").")
- **DeclException3** (ExcClosedIntervalRange, double, double, double,<< arg1<< " is not in the range ["<< arg2<< ", "<< arg3<< "].")

9.5.1 Detailed Description

Definition of self-defined general exceptions.

Date

2021-06-10

Author

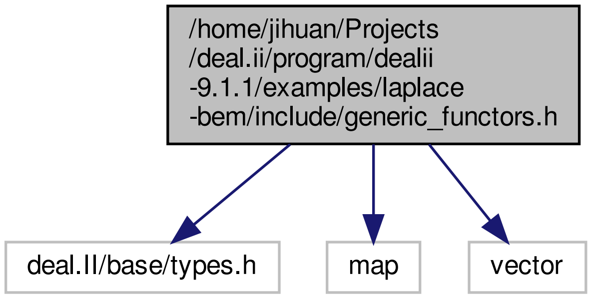
Jihuan Tian

9.6 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/generic_functors.h File Reference

This header file contains a set of self-defined generic functors.

```
#include <deal.II/base/types.h>
#include <map>
#include <vector>
```

Include dependency graph for generic_functors.h:



This graph shows which files directly or indirectly include this file:

Functions

- template<typename InputIterator , typename T >
  InputIterator find_pointer_data (InputIterator first, InputIterator last, const T ∗value_pointer)
- void build_index_set_global_to_local_map (const std::vector< dealii::types::global_dof_index > &index_set_as_local_to_global_map, std::map< dealii::types::global_dof_index, size_t > &global_to_local_map)

9.6.1 Detailed Description

This header file contains a set of self-defined generic functors.

Date

2021-07-20

Author

Jihuan Tian

9.6.2 Function Documentation

9.6.2.1 build_index_set_global_to_local_map()

```
void build_index_set_global_to_local_map (
             const std::vector< dealii::types::global_dof_index > & index_set_as_local_to_↩
global_map,
             std::map< dealii::types::global_dof_index, size_t > & global_to_local_map )
```

Build a map from global indices to local indices based on the given index set, which is actually a map from local indices to global indices.

Parameters

index_set_as_local_to_global_map	
global_to_local_map	

Referenced by convertHMatBlockToRkMatrix(), f_h_mmult(), find_pointer_data(), h_f_mmult(), h_rk_mmult(), Init↩ AndCreateHMatrixChildren(), main(), and rk_h_mmult().

9.6.2.2 find_pointer_data()

```
template<typename InputIterator , typename T >
InputIterator find_pointer_data (
             InputIterator first,
             InputIterator last,
             const T * value_pointer )
```

Find the pointer in a list of pointers of the same type. The comparison is based on the data being pointed instead of the pointer addresses.

Parameters

first	
last	
value_pointer	

Returns

N.B. Here we use (∗first) to get the pointer stored in the container. Then use (∗(∗first)) to get the data associated with this pointer.

References build_index_set_global_to_local_map().

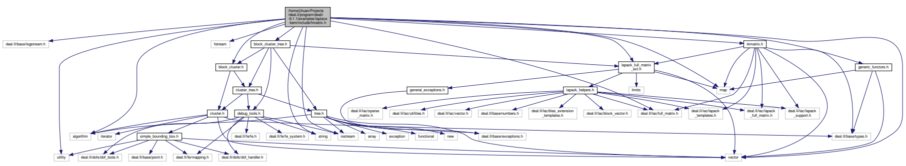
Referenced by HMatrix< spacedim, Number >::coarsen_to_partition().

9.7 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/hmatrix.h File Reference

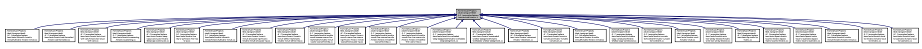
Definition of hierarchical matrix.

```
#include <deal.II/base/logstream.h>
#include <deal.II/lac/full_matrix.h>
#include <algorithm>
#include <fstream>
#include <iostream>
#include <map>
#include <string>
#include <utility>
#include <vector>
#include "block_cluster.h"
#include "block_cluster_tree.h"
#include "generic_functors.h"
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for hmatrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class HMatrix< spacedim, Number >

Enumerations

- enum HMatrixType { FullMatrixType, RkMatrixType, HierarchicalMatrixType, UndefinedMatrixType }

Functions

- **DeclException1** (ExcInvalidHMatrixType, HMatrixType,<< "Invalid HMatrix type "<< arg1)
- template<int spacedim, typename Number = double>
  void InitHMatrixWrtBlockClusterNode (HMatrix< spacedim, Number > &hmat, typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node)
- template<int spacedim, typename Number = double>
  void InitHMatrixWrtBlockClusterNode (HMatrix< spacedim, Number > &hmat, typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, const std::vector< std::pair< HMatrix< spacedim, Number > ∗, HMatrix< spacedim, Number > ∗>> &Sigma_P)

- template< int spacedim, typename Number = double >
  void InitHMatrixWrtBlockClusterNode (HMatrix< spacedim, Number > &hmat, typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, const std::pair< HMatrix< spacedim, Number > *, HMatrix< spacedim, Number > *> &hmat_pair)
- template< int spacedim, typename Number = double >
  void InitAndCreateHMatrixChildren (HMatrix< spacedim, Number > *hmat, typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, const unsigned int fixed_rank_k, bool is_build_index_set_global_to_local_map=true)
- template< int spacedim, typename Number = double >
  void InitAndCreateHMatrixChildren (HMatrix< spacedim, Number > *hmat, typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, const unsigned int fixed_rank_k, const LAPACKFullMatrixExt< Number > &M, bool is_build_index_set_global_to_local_map=true)
- template< int spacedim, typename Number = double >
  void InitAndCreateHMatrixChildren (HMatrix< spacedim, Number > *hmat, typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, const unsigned int fixed_rank_k, const LAPACKFullMatrixExt< Number > &M, const std::map< types::global_dof_index, size_t > &row_index_global_to_local_map_for_M, const std::map< types::global_dof_index, size_t > &col_index_global_to_local_map_for_M, bool is_build_index_set_global_to_local_map=true)
- template< int spacedim, typename Number = double >
  void InitAndCreateHMatrixChildren (HMatrix< spacedim, Number > *hmat, typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node, HMatrix< spacedim, Number > &&H)
- template< int spacedim, typename Number >
  void RefineHMatrixWrtExtendedBlockClusterTree (HMatrix< spacedim, Number > *starting_hmat, HMatrix< spacedim, Number > *current_hmat)
- template< int spacedim, typename Number = double >
  void convertHMatBlockToRkMatrix (HMatrix< spacedim, Number > *hmat_block, const unsigned int fixed_rank_k, const HMatrix< spacedim, Number > *hmat_root_block=nullptr, size_t *calling_counter=nullptr, const std::string &output_file_base_name=std::string("hmat-bct"))
- template< int spacedim, typename Number = double >
  void h_rk_mmult (HMatrix< spacedim, Number > &M1, const RkMatrix< Number > &M2, RkMatrix< Number > &M)
- template< int spacedim, typename Number = double >
  void h_rk_mmult_for_h_h_mmult (HMatrix< spacedim, Number > *M1, const HMatrix< spacedim, Number > *M2, HMatrix< spacedim, Number > *M, bool is_M1M2_last_in_M_Sigma_P=true)
- template< int spacedim, typename Number = double >
  void rk_h_mmult (const RkMatrix< Number > &M1, HMatrix< spacedim, Number > &M2, RkMatrix< Number > &M)
- template< int spacedim, typename Number = double >
  void rk_h_mmult_for_h_h_mmult (const HMatrix< spacedim, Number > *M1, HMatrix< spacedim, Number > *M2, HMatrix< spacedim, Number > *M, bool is_M1M2_last_in_M_Sigma_P=true)
- template< int spacedim, typename Number = double >
  void h_f_mmult (HMatrix< spacedim, Number > &M1, const LAPACKFullMatrixExt< Number > &M2, LAPACKFullMatrixExt< Number > &M)
- template< int spacedim, typename Number = double >
  void h_f_mmult (HMatrix< spacedim, Number > &M1, const LAPACKFullMatrixExt< Number > &M2, RkMatrix< Number > &M)
- template< int spacedim, typename Number = double >
  void h_f_mmult_for_h_h_mmult (HMatrix< spacedim, Number > *M1, const HMatrix< spacedim, Number > *M2, HMatrix< spacedim, Number > *M, bool is_M1M2_last_in_M_Sigma_P=true)
- template< int spacedim, typename Number = double >
  void f_h_mmult (const LAPACKFullMatrixExt< Number > &M1, HMatrix< spacedim, Number > &M2, LAPACKFullMatrixExt< Number > &M)
- template< int spacedim, typename Number >
  void **f_h_mmult** (const LAPACKFullMatrixExt< Number > &M1, HMatrix< spacedim, Number > &M2, RkMatrix< Number > &M)
- template< int spacedim, typename Number = double >
  void f_h_mmult_for_h_h_mmult (const HMatrix< spacedim, Number > *M1, HMatrix< spacedim, Number > *M2, HMatrix< spacedim, Number > *M, bool is_M1M2_last_in_M_Sigma_P=true)

- template<int spacedim, typename Number = double>
  void h_h_mmult_phase1_recursion (HMatrix< spacedim, Number > *M, BlockClusterTree< spacedim, Number > &Tind)
- template<int spacedim, typename Number = double>
  void h_h_mmult_phase2 (HMatrix< spacedim, Number > &M, BlockClusterTree< spacedim, Number > &target_bc_tree, const unsigned int fixed_rank)
- template<int spacedim, typename Number = double>
  void copy_hmatrix_node (HMatrix< spacedim, Number > &hmat_dst, const HMatrix< spacedim, Number > &hmat_src)
- template<int spacedim, typename Number = double>
  void copy_hmatrix_node (HMatrix< spacedim, Number > &hmat_dst, HMatrix< spacedim, Number > &&hmat_src)
- template<int spacedim, typename Number = double>
  void copy_hmatrix (HMatrix< spacedim, Number > &hmat_dst, const HMatrix< spacedim, Number > &hmat_src)
- template<int spacedim, typename Number = double>
  void **print_h_submatrix_accessor** (std::ostream &out, const std::string &name, const HMatrix< spacedim, Number > &M)
- template<int spacedim, typename Number = double>
  void **print_h_h_submatrix_mmult_accessor** (std::ostream &out, const std::string &name1, const HMatrix< spacedim, Number > &M1, const std::string &name2, const HMatrix< spacedim, Number > &M2)

9.7.1 Detailed Description

Definition of hierarchical matrix.

Date

2021-06-06

Author

Jihuan Tian

9.7.2 Enumeration Type Documentation

9.7.2.1 HMatrixType

`enum HMatrixType`

Matrix type of an HMaxtrix, which can be full matrix in the near field, rank-k matrix in the far field and hierarchical matrix which does not belong to the leaf set of a block cluster tree.

Enumerator

FullMatrixType	FullMatrixType.
RkMatrixType	RkMatrixType.
HierarchicalMatrixType	HierarchicalType.
UndefinedMatrixType	UndefinedMatrixType.

9.7.3 Function Documentation

9.7.3.1 convertHMatBlockToRkMatrix()

```
template<int spacedim, typename Number = double>
void convertHMatBlockToRkMatrix (
            HMatrix< spacedim, Number > * hmat_block,
            const unsigned int fixed_rank_k,
            const HMatrix< spacedim, Number > * hmat_root_block = nullptr,
            size_t * calling_counter = nullptr,
            const std::string & output_file_base_name = std::string("hmat-bct") )
```

Convert an $\mathcal{H}$-matrix block `hmat_block` recursively into a rank-k matrix or a full matrix, which depends on whether the block cluster associated with `hmat_block` is large or not.

Generally speaking, this method can be considered as the agglomeration of all descendants of `hmat_block`.

Note This method implements the operator $\mathcal{T}_r^{\mathcal{R}\leftarrow\mathcal{H}}$, i.e. the algorithm $Convert\_H$ in (7.8) in Hackbusch's $\mathcal{H}$-matrix book.

This $\mathcal{H}$-matrix block is implemented as a node in a whole $\mathcal{H}$-matrix hierarchy. This conversion algorithm will recursively descend in the hierarchical matrices for processing:

1. when the current matrix block belongs to the near field set P^{-}, it is represented as a full matrix and no operations will be applied to it;
2. when it belongs to the far field set P^{+}, it is already a rank-k matrix, which will then be truncated to the given `fixed_rank_k`;
3. when it is not a leaf, i.e. it is a hierarchical matrix, this function will be called recursively for each of its children. After that,

   a. if the block cluster related to the current matrix is large, pairwise agglomeration for rank-k matrices will be performed and a rank-k matrix will be obtained with the given rank `fixed_rank_k`;

   b. if the block cluster related to the current matrix is small, agglomeration of full matrices will be performed and a full matrix will be obtained.

Parameters

hmat_block	the pointer to the current matrix block from which the recursion will start.
fixed_rank_k	the fixed rank to which the rank-k matrices in the far field set will be truncated.
hmat_root_block	the pointer to the root $\mathcal{H}$-matrix block, which is only used for exporting matrix partition structure for further visualization.
calling_counter	the pointer to the counter which records the current total number of calling times of this function. Its value will be used to construct the name of the output file, which stores the matrix partition structure.
output_file_base_name	the based name of the output file which stores the matrix partition structure.

Work flow When the current $\mathcal{H}$-matrix block belongs to the leaf set.

When the current $\mathcal{H}$-matrix belongs to the near field set, it should be of a full matrix type. Therefore, we make an assertion here. After that we do nothing, since a near field node should always be represented as a full matrix, thus the rank truncation should not be applied.

When the current $\mathcal{H}$-matrix belongs to the far field set, it should be of a rank-k matrix type. Therefore, we make an assertion here. After that the rank-k matrix block is truncated to the specified rank.

When the current $\mathcal{H}$-matrix block does not belong to the leaf set, recursively convert each child of it to rank-k matrix if possible.

When the current $\mathcal{H}$-matrix block belongs to the near field set, we perform the operation of full matrix agglomeration.

Note Normally, this case cannot happen because when an $\mathcal{H}$-matrix block belongs to the near field, it is represented as a full matrix and belongs to the leaf set. However, this contradicts the precondition that the current $\mathcal{H}$-matrix block does not belong to the leaf set.

But still this situation may happen during the conversion of an $\mathcal{H}$-matrix to a different block cluster tree.

The general work flow for the agglomeration of a set of full matrix blocks is as below.

1. Create a large full matrix on the heap and assemble all submatrices into it which depends on the split mode of the block cluster.
   a. When it is `CrossSplitMode`, apply agglomeration of four full submatrices.
   b. When the split mode is `HorizontalSplitMode`, apply agglomeration of two full submatrices via vertical stacking.
   c. When the split mode is `VerticalSplitMode`, apply agglomeration of two full submatrices via horizontal stacking.
2. Delete all submatrices associated with the current $\mathcal{H}$-matrix and clear the `std::vector` storing submatrix pointers.
3. Associate the new large full matrix with the current $\mathcal{H}$-matrix.
4. Update the $\mathcal{H}$-matrix type as `FullMatrix`.

About matrix assembly for `CrossSplitMode`

Let the block cluster associated with the current $\mathcal{H}$-matrix is $\tau \times \sigma$. Assume the clusters are partitioned as $\tau = [\tau_1, \tau_2]$ and $\sigma = [\sigma_1, \sigma_2]$. Then the ordering of the child block clusters are $\tau_1 \times \sigma_1, \tau_1 \times \sigma_2, \tau_2 \times \sigma_1, \tau_2 \times \sigma_2$.

Build the map from the global DoF indices to the local row indices of the current $\mathcal{H}$-matrix node, if necessary.

Build the map from the global DoF indices to the local column indices of the current $\mathcal{H}$-matrix node, if necessary.

When the current $\mathcal{H}$-matrix block belongs to the far field set, perform the pairwise matrix agglomeration of rank-k submatrices or full submatrices, which has been implemented into the constructor of `RkMatrix`.

Build the map from the global DoF indices to the local row indices of the current $\mathcal{H}$-matrix node, if necessary.

Build the map from the global DoF indices to the local column indices of the current $\mathcal{H}$-matrix node, if necessary.

If the children of the current $\mathcal{H}$-matrix block are rank-k matrices, perform the pairwise rank-k matrix agglomeration directly.

If the children of the current $\mathcal{H}$-matrix block are full matrices, firstly convert all of them into rank-k matrices, then perform the pairwise rank-k matrix agglomeration.

Other cases are invalid.

If the children of the current $\mathcal{H}$-matrix block are rank-k matrices, perform the pairwise rank-k matrix agglomeration directly.

If the children of the current $\mathcal{H}$-matrix block are full matrices, firstly convert all of them into rank-k matrices, then perform the pairwise rank-k matrix agglomeration.

Other cases are invalid.

If the children of the current $\mathcal{H}$-matrix block are rank-k matrices, perform the pairwise rank-k matrix agglomeration directly.

If the children of the current $\mathcal{H}$-matrix block are full matrices, firstly convert all of them into rank-k matrices, then perform the pairwise rank-k matrix agglomeration.

Other cases are invalid.

Visualize the partition structure if a not-null pointer to the root $\mathcal{H}$-matrix node and a a not-null pointer to a `calling_counter` are provided.

References HMatrix< spacedim, Number >::bc_node, build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local_map, HMatrix< spacedim, Number >::col_indices, FullMatrix↩Type, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_index_↩global_to_local_map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, HMatrix< spacedim, Number >::type, and UndefinedMatrixType.

Referenced by main().

9.7.3.2 copy_hmatrix()

```
template<int spacedim, typename Number = double>
void copy_hmatrix (
            HMatrix< spacedim, Number > & hmat_dst,
            const HMatrix< spacedim, Number > & hmat_src )
```

Recursively copy an $\mathcal{H}$-matrix into the target matrix.

Parameters

M_dst	
M_src	

Copy the current $\mathcal{H}$-matrix node.

Recursively copy child $\mathcal{H}$-matrix nodes.

Create a corresponding child $\mathcal{H}$-matrix node on the heap and push it back into the `submatrices` list of the current $\mathcal{H}$-matrix node.

References copy_hmatrix_node(), and HMatrix< spacedim, Number >::submatrices.

9.7.3.3 copy_hmatrix_node() [1/2]

```
template<int spacedim, typename Number = double>
void copy_hmatrix_node (
            HMatrix< spacedim, Number > & hmat_dst,
            const HMatrix< spacedim, Number > & hmat_src )
```

Shallow copy an $\mathcal{H}$-matrix node into the target node, i.e. the copy is limited within the current node without recursion into its descendants. This function will be called by `copy_hmatrix`.

N.B. Do not copy the list `submatrices` from the source submatrix, because newly created child matrices will be pushed back into this list.

Do not copy the list `leaf_set`. After the whole $\mathcal{H}$-matrix hierarchy has been constructed, the leaf set will be built in the constructor.

Do not copy the working data: `Sigma_P`, `Sigma_F`, `Sigma_R` and `Tind`.

Parameters

hmat_dst	
hmat_src	

Copy the rank-k matrix in the source submatrix if it is not `NULL`.

Copy the full matrix in the source submatrix if it is not `NULL`.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_index_global_to_↩ local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, and H↩ Matrix< spacedim, Number >::type.

Referenced by copy_hmatrix().

9.7.3.4 copy_hmatrix_node() [2/2]

```
template<int spacedim, typename Number = double>
void copy_hmatrix_node (
            HMatrix< spacedim, Number > & hmat_dst,
            HMatrix< spacedim, Number > && hmat_src )
```

Deep copy an $\mathcal{H}$-matrix node into the target node, i.e. the copy is limited within the current node without recursion into its descendants.

Parameters

hmat_dst	
hmat_src	

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_index_global_to_↩ local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, HMatrix< spacedim, Number >::leaf_set, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, H↩ Matrix< spacedim, Number >::rkmatrix, HMatrix< spacedim, Number >::row_index_global_to_local_map, H↩ Matrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, HMatrix< spacedim, Number >::submatrices, H↩ Matrix< spacedim, Number >::Tind, and HMatrix< spacedim, Number >::type.

9.7.3.5 f_h_mmult()

```
template<int spacedim, typename Number = double>
void f_h_mmult (
            const LAPACKFullMatrixExt< Number > & M1,
            HMatrix< spacedim, Number > & M2,
            LAPACKFullMatrixExt< Number > & M )
```

Build the map from global DoF indices to local matrix indices if necessary.

References build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local↩_map, HMatrix< spacedim, Number >::col_indices, LAPACKFullMatrixExt< Number >::fill_row(), LAPACKFull↩MatrixExt< Number >::get_row(), HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, rk_h_↩mmult(), HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row↩_indices, and HMatrix< spacedim, Number >::Tvmult_local_vector().

Referenced by f_h_mmult_for_h_h_mmult(), and main().

9.7.3.6 f_h_mmult_for_h_h_mmult()

```
template<int spacedim, typename Number = double>
void f_h_mmult_for_h_h_mmult (
            const HMatrix< spacedim, Number > * M1,
            HMatrix< spacedim, Number > * M2,
            HMatrix< spacedim, Number > * M,
            bool is_M1M2_last_in_M_Sigma_P = true )
```

Full matrix is returned.

Rank-k matrix is returned.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_indices, f_h_mmult(), HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, HMatrix< spacedim, Number >::remove_hmat_pair↩_from_mm_product_list(), RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, and H↩Matrix< spacedim, Number >::type.

9.7.3.7 h_f_mmult() [1/2]

```
template<int spacedim, typename Number = double>
void h_f_mmult (
            HMatrix< spacedim, Number > & M1,
            const LAPACKFullMatrixExt< Number > & M2,
            LAPACKFullMatrixExt< Number > & M )
```

Calculate the product of two $\mathcal{H}$-matrix nodes, where the second one is a full matrix and the result is also represented as a full matrix because the associated block cluster node $\tau \times \rho$ is small.

Parameters

M1	
M2	
M	

Build the map from global DoF indices to local matrix indices if necessary.

References build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local↩_map, HMatrix< spacedim, Number >::col_indices, LAPACKFullMatrixExt< Number >::fill_col(), LAPACKFull↩

MatrixExt< Number >::get_column(), HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, H↵ Matrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, and HMatrix< spacedim, Number >::vmult_local_vector().

Referenced by h_f_mmult_for_h_h_mmult(), and main().

9.7.3.8 h_f_mmult() [2/2]

```
template<int spacedim, typename Number = double>
void h_f_mmult (
            HMatrix< spacedim, Number > & M1,
            const LAPACKFullMatrixExt< Number > & M2,
            RkMatrix< Number > & M )
```

Calculate the product of two $\mathcal{H}$-matrix nodes, where the second one is a full matrix and the result is represented as a rank-k matrix because the associated block cluster is large.

The second matrix `M2` will be firstly converted to a rank-k matrix. Then its multiplication with `M1` will be carried by calling `h_rk_mmult`. Since the conversion from a full matrix to a rank-k matrix will modify the original data, a copy of `M2` will be created.

Parameters

M1	
M2	
M	

Create a local copy of the full matrix `M2`.

Convert the full matrix `M2` to a rank-k matrix.

References h_rk_mmult(), and HMatrix< spacedim, Number >::n.

9.7.3.9 h_f_mmult_for_h_h_mmult()

```
template<int spacedim, typename Number = double>
void h_f_mmult_for_h_h_mmult (
            HMatrix< spacedim, Number > * M1,
            const HMatrix< spacedim, Number > * M2,
            HMatrix< spacedim, Number > * M,
            bool is_M1M2_last_in_M_Sigma_P = true )
```

Full matrix is returned.

Rank-k matrix is returned.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, h_f_mmult(), HMatrix< spacedim, Number >::remove_hmat_↵ pair_from_mm_product_list(), RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, and HMatrix< spacedim, Number >::type.

9.7.3.10 h_h_mmult_phase1_recursion()

```
template<int spacedim, typename Number = double>
void h_h_mmult_phase1_recursion (
            HMatrix< spacedim, Number > * M,
            BlockClusterTree< spacedim, Number > & Tind )
```

There are still multiplication subtasks stored in `Sigma_P` to be handled recursively.

After previous reduction and splitting, the matrix multiplication for the current $\mathcal{H}$-matrix node should be replaced by the multiplication subtasks for submatrices. These subtasks are recorded as $\mathcal{H}$-matrix node pairs which are stored in Σ_b^P of the submatrices.

References HMatrix< spacedim, Number >::determine_mm_split_mode_from_Sigma_P(), HMatrix< spacedim, Number >::h_h_mmult_reduction(), and HMatrix< spacedim, Number >::Sigma_P.

9.7.3.11 h_h_mmult_phase2()

```
template<int spacedim, typename Number = double>
void h_h_mmult_phase2 (
            HMatrix< spacedim, Number > & M,
            BlockClusterTree< spacedim, Number > & target_bc_tree,
            const unsigned int fixed_rank )
```

Collect terms in `Sigma_R` and `Sigma_F` for the leaf nodes.

Here we make sure that $\mathcal{H}$-matrix pairs in the list Σ_b^P have all been processed and erased.

Perform pairwise formatted addition for the list of rank-k matrices.

Distribute matrices stored in Σ_b^R and Σ_b^F of each non-leaf node to its leaf nodes.

Convert the calculated product matrix to the specified matrix structure.

References HMatrix< spacedim, Number >::convert_between_different_block_cluster_trees(), HMatrix< spacedim, Number >::distribute_all_non_leaf_nodes_sigma_r_and_f_to_leaves(), HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, HMatrix< spacedim, Number >::leaf_set, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, HMatrix< spacedim, Number >::Tind, and HMatrix< spacedim, Number >::type.

9.7.3.12 h_rk_mmult()

```
template<int spacedim, typename Number = double>
void h_rk_mmult (
            HMatrix< spacedim, Number > & M1,
            const RkMatrix< Number > & M2,
            RkMatrix< Number > & M )
```

Calculate the product of two $\mathcal{H}$-matrix nodes, where the second one `M2` has `RkMatrixType` and the result will also be a rank-k matrix.

The arithmetic operation to be performed is

$$M = M_1 \cdot M_2 = M_1(AB^T) = (M_1 A)B^T = A'B^T,$$

where $A' = M_1 A$ is calculated as a series of $\mathcal{H}$-matrix-vector multiplications. For details,

$$M_1 A = M_1 \begin{bmatrix} a_{\sigma,1} & \cdots & a_{\sigma,r} \end{bmatrix} = \begin{bmatrix} M_1 a_{\sigma,1} & \cdots & M_1 a_{\sigma,r} \end{bmatrix} = \begin{bmatrix} a'_{\tau,1} & \cdots & a'_{\tau,r} \end{bmatrix}.$$

It can be seen that the formal rank r of the result matrix `M` is the same as that of `M2`.

Parameters

M1	
M2	
M	

Create a temporary `Vector` storing a column $a_{\sigma,j}$ in the `A` component of `M2` and another `Vector` $a'_{\tau,j}$ storing the matrix-vector product $M_1 \cdot a_{\sigma,j}$.

Initialize the result rank-k matrix `M` with the formal rank of `M2`. Its `B` component matrix is the same as that of `M2`.

Build the map from global DoF indices to local matrix indices if necessary.

Then we calculate the `A` component matrix of `M`, which is `M1*M2`.A.

Fill the result vector into the `A` component matrix of `M`.

References build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local↩
_map, HMatrix< spacedim, Number >::col_indices, RkMatrix< Number >::formal_rank, RkMatrix< Number >↩
::m, HMatrix< spacedim, Number >::m, RkMatrix< Number >::n, HMatrix< spacedim, Number >::n, RkMatrix< Number >::reinit(), HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, and HMatrix< spacedim, Number >::vmult_local_vector().

Referenced by h_f_mmult(), h_rk_mmult_for_h_h_mmult(), and main().

9.7.3.13 h_rk_mmult_for_h_h_mmult()

```
template<int spacedim, typename Number = double>
void h_rk_mmult_for_h_h_mmult (
            HMatrix< spacedim, Number > * M1,
            const HMatrix< spacedim, Number > * M2,
            HMatrix< spacedim, Number > * M,
            bool is_M1M2_last_in_M_Sigma_P = true )
```

Calculate the product of two $\mathcal{H}$-matrix nodes, where the second one `M2` has `RkMatrixType` and the result will also be a rank-k matrix. This function is to be called by the matrix-matrix multiplication function.

Parameters

M1	
M2	
M	

References HMatrix< spacedim, Number >::col_indices, h_rk_mmult(), HMatrix< spacedim, Number >↩
::remove_hmat_pair_from_mm_product_list(), HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, HMatrix< spacedim, Number >::Sigma_R, and HMatrix< spacedim, Number >::type.

9.7.3.14 InitAndCreateHMatrixChildren() [1/4]

```
template<int spacedim, typename Number = double>
void InitAndCreateHMatrixChildren (
            HMatrix< spacedim, Number > * hmat,
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            const unsigned int fixed_rank_k,
            bool is_build_index_set_global_to_local_map = true )
```

Recursively construct the children of an $\mathcal{H}$-matrix with respect to a block cluster tree by starting from a tree node which is associated with the current $\mathcal{H}$-matrix.

The matrices in the leaf set are initialized with zero values. The rank of the near field matrices are predefined fixed values.

Parameters

hmat	Pointer to the current $\mathcal{H}$-matrix node, **which has already been created on the heap but with its internal data left empty.**
bc_node	Pointer to a TreeNode in a BlockClusterTree, which is to be associated with `hmat`.

Link `hmat` with `bc_node`.

Link row and column indices stored in the clusters τ and σ respectively.

Update the matrix dimension of `hmat`.

When the block cluster node `bc_node` has children, set the current `hmat` type as `HierarchicalMatrix↩Type`.

Then we will continue constructing its hierarchical submatrices.

Create an empty HMatrix on the heap.

Append the initialized child to the list of submatrices of `hmat`.

Build the maps from global row and column indices respectively to local indices.

Update the current matrix type according to the identity of the block cluster node. When the block cluster belongs to the near field, `hmat` should be represented as a `LAPACKFullMatrixExt`. When the block cluster belongs to the far field, `hmat` should be represented as an `RkMatrix`. Correspondingly, new matrices, either full matrix or rank-k matrix will be created on the heap and assigned to the current $\mathcal{H}$-matrix.

References HMatrix< spacedim, Number >::bc_node, build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, TreeNode< T, N >::get_child_num(), TreeNode< T, N >::get↩_child_pointer(), TreeNode< T, N >::get_data_reference(), HierarchicalMatrixType, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

Referenced by InitAndCreateHMatrixChildren().

9.7.3.15 InitAndCreateHMatrixChildren() [2/4]

```
template<int spacedim, typename Number = double>
void InitAndCreateHMatrixChildren (
            HMatrix< spacedim, Number > * hmat,
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            const unsigned int fixed_rank_k,
            const LAPACKFullMatrixExt< Number > & M,
            bool is_build_index_set_global_to_local_map = true )
```

Recursively construct the children of an $\mathcal{H}$-matrix with respect to a block cluster tree by starting from a tree node which is associated with the current $\mathcal{H}$-matrix.

The matrices in the leaf set are initialized with the data in the given global full matrix `M`, which is created on the complete block cluster index set $I \times J$ and whose elements should be accessed via indices stored in the block cluster. The rank of the near field matrices are predefined fixed values.

During the recursive calling of this function, the source data matrix `M` is kept intact, which will not be restricted to small matrix blocks.

Parameters

hmat	Pointer to the current $\mathcal{H}$-matrix node, **which has already been created on the heap but with its internal data left empty.**
bc_node	Pointer to a TreeNode in a BlockClusterTree, which is to be associated with `hmat`.
M	The global full matrix containing all the data required to initialize the $\mathcal{H}$-matrix.

Link `hmat` with `bc_node`.

Link row and column indices.

Update the matrix dimension of `hmat`.

When the block cluster node `bc_node` has children, set the current `hmat` type as `HierarchicalMatrix↩Type`. Then we will continue constructing hierarchical submatrices.

Create an empty HMatrix on the heap.

Append the initialized child to the list of submatrices of `hmat`.

Build the maps from global row and column indices respectively to local indices.

Update the current matrix type according to the identity of the block cluster node. When the block cluster belongs to the near field, `hmat` should be represented as a `LAPACKFullMatrixExt`. When the block cluster belongs to the far field, `hmat` should be represented as an `RkMatrix`. Correspondingly, new matrices, either full matrix or rank-k matrix will be created on the heap and assigned to the current $\mathcal{H}$-matrix.

Assign matrix values from `M` to the current HMatrix.

References HMatrix< spacedim, Number >::bc_node, build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, TreeNode< T, N >::get_child_num(), TreeNode< T, N >::get_↩child_pointer(), TreeNode< T, N >::get_data_reference(), HierarchicalMatrixType, InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

9.7.3.16 InitAndCreateHMatrixChildren() [3/4]

```
template<int spacedim, typename Number = double>
void InitAndCreateHMatrixChildren (
            HMatrix< spacedim, Number > * hmat,
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            const unsigned int fixed_rank_k,
            const LAPACKFullMatrixExt< Number > & M,
            const std::map< types::global_dof_index, size_t > & row_index_global_to_local_↩
map_for_M,
            const std::map< types::global_dof_index, size_t > & col_index_global_to_local_↩
map_for_M,
            bool is_build_index_set_global_to_local_map = true )
```

Recursively construct the children of an $\mathcal{H}$-matrix with respect to a block cluster tree by starting from a tree node which is associated with the current $\mathcal{H}$-matrix.

The matrices in the leaf set are initialized with the data in the given full matrix `M`, which is created on the block cluster index set $\tau \times \sigma$ associated with the current $\mathcal{H}$-matrix. Hence, this full matrix is just a block of the original global full matrix created on the block cluster index set $I \times J$. The rank of the near field matrices are predefined fixed values.

During the recursive calling of this function, the source data matrix `M` is kept intact, which will not be restricted to small matrix blocks.

Parameters

hmat	Pointer to the current $\mathcal{H}$-matrix node, **which has already been created on the heap but with its internal data left empty.**
bc_node	Pointer to a TreeNode in a BlockClusterTree, which is to be associated with `hmat`.
M	The full matrix, as a submatrix of the global full matrix, containing all the data required to initialize the $\mathcal{H}$-matrix.
row_index_global_to_local_map_for↩ _M	The map from the global row indices to the local indices of the matrix associated the $\mathcal{H}$-matrix when first calling this recursive function.
col_index_global_to_local_map_for_M	The map from the global column indices to the local indices of the matrix associated the $\mathcal{H}$-matrix when first calling this recursive function.

Link `hmat` with `bc_node`.

Link row and column indices.

Update the matrix dimension of `hmat`.

When the block cluster node `bc_node` has children, set the current `hmat` type as `HierarchicalMatrix↩ Type`. Then we will continue constructing hierarchical submatrices.

Create an empty HMatrix on the heap.

Append the initialized child to the list of submatrices of `hmat`.

Build the maps from global row and column indices respectively to local indices.

Update the current matrix type according to the identity of the block cluster node. When the block cluster belongs to the near field, `hmat` should be represented as a `LAPACKFullMatrixExt`. When the block cluster belongs to the far field, `hmat` should be represented as an `RkMatrix`. Correspondingly, new matrices, either full matrix or rank-k matrix will be created on the heap and assigned to the current $\mathcal{H}$-matrix.

Assign matrix values from `M` to the current HMatrix.

References HMatrix< spacedim, Number >::bc_node, build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrixType, TreeNode< T, N >::get_child_num(), TreeNode< T, N >::get_child_pointer(), TreeNode< T, N >::get_data_reference(), HierarchicalMatrixType, InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

9.7.3.17 InitAndCreateHMatrixChildren() [4/4]

```
template<int spacedim, typename Number = double>
void InitAndCreateHMatrixChildren (
            HMatrix< spacedim, Number > * hmat,
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            HMatrix< spacedim, Number > && H )
```

Recursively construct the children of an $\mathcal{H}$-matrix node with respect to a block cluster tree by starting from a tree node which is associated with the current $\mathcal{H}$-matrix node.

The matrices in the leaf set take the data migrated from the leaf set of the given $\mathcal{H}$-matrix `M`.

Parameters

hmat	
M	

Link `hmat` with `bc_node`.

Link row and column indices stored in the clusters τ and σ respectively..

Update the matrix dimension of `hmat`.

When the block cluster node `bc_node` has children, set the current `hmat` type as `HierarchicalMatrixType`.

Then we will continue constructing its hierarchical submatrices.

Create an empty HMatrix on the heap.

Append the initialized child to the list of submatrices of `hmat`.

When the current $\mathcal{H}$-matrix node is a leaf, migrate the data from the leaf set of `H` to it.

Shallow copy the found $\mathcal{H}$-matrix node in the leaf set to the current $\mathcal{H}$-matrix node.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_indices, TreeNode< T, N >::get_child_num(), TreeNode< T, N >::get_child_pointer(), HierarchicalMatrixType, InitAndCreateHMatrixChildren(), HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

9.7.3.18 InitHMatrixWrtBlockClusterNode() [1/3]

```
template<int spacedim, typename Number = double>
void InitHMatrixWrtBlockClusterNode (
            HMatrix< spacedim, Number > & hmat,
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node )
```

Initialize an $\mathcal{H}$-matrix node with respect to a block cluster node. The list Σ_b^P is set to empty.

Parameters

hmat	
bc_node	
Sigma↩_P	

Link `hmat` with `bc_node`.

Link row and column indices stored in the clusters τ and σ respectively.

Update the matrix dimension of `hmat`.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::row_indices, H↩Matrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, and HMatrix< spacedim, Number >::Sigma_R.

9.7.3.19 InitHMatrixWrtBlockClusterNode() [2/3]

```
template<int spacedim, typename Number = double>
void InitHMatrixWrtBlockClusterNode (
            HMatrix< spacedim, Number > & hmat,
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            const std::vector< std::pair< HMatrix< spacedim, Number > *, HMatrix< spacedim,
Number > *>> & Sigma_P )
```

Initialize an $\mathcal{H}$-matrix node with respect to a block cluster node. Its member list Σ_b^P will be merged with the given `Sigma_P`.

Parameters

hmat	
bc_node	
Sigma↩_P	

Link `hmat` with `bc_node`.

Link row and column indices stored in the clusters τ and σ respectively.

Update the matrix dimension of `hmat`.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, and HMatrix< spacedim, Number >::Sigma_R.

9.7.3.20 InitHMatrixWrtBlockClusterNode() [3/3]

```
template<int spacedim, typename Number = double>
void InitHMatrixWrtBlockClusterNode (
            HMatrix< spacedim, Number > & hmat,
            typename BlockClusterTree< spacedim, Number >::node_const_pointer_type bc_node,
            const std::pair< HMatrix< spacedim, Number > *, HMatrix< spacedim, Number > *>
& hmat_pair )
```

Initialize an $\mathcal{H}$-matrix node with respect to a block cluster node. The given `hmat_pair` will be appended to the list Σ_b^P.

Parameters

hmat	
bc_node	
hmat_pair	

Link `hmat` with `bc_node`.

Link row and column indices stored in the clusters τ and σ respectively.

Update the matrix dimension of `hmat`.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::Sigma_P, and HMatrix< spacedim, Number >::Sigma_R.

9.7.3.21 RefineHMatrixWrtExtendedBlockClusterTree()

```
template<int spacedim, typename Number >
void RefineHMatrixWrtExtendedBlockClusterTree (
            HMatrix< spacedim, Number > * starting_hmat,
            HMatrix< spacedim, Number > * current_hmat )
```

Refine an $\mathcal{H}$-matrix node with respect to its associated block cluster tree which has already been extended to be finer than the original tree. The $\mathcal{H}$-matrix node should be of either `FullMatrixType` or `RkMatrixType`, i.e. it belongs to the leaf set of the block cluster tree before extension.

Parameters

starting_hmat	The pointer to the initial $\mathcal{H}$-matrix node from which this recursive function is called for the first time, i.e. the $\mathcal{H}$-matrix node from which the refinement begins.
current_hmat	The pointer to the current $\mathcal{H}$-matrix node being handled during the recursion. For the first time of calling this function, `current_hmat` is the same as `starting_hmat`.

Work flow Because the $\mathcal{H}$-matrix node from which the refinement begins belongs to the leaf set of the original block cluster tree, its $\mathcal{H}$-matrix type can only be `FullMatrixType` or `RkMatrixType`. Therefore, we make an assertion here.

Determine the total number of children of the current $\mathcal{H}$-matrix node by querying its associated block cluster node. We do it like this is because the block cluster tree has already been extended which contains a set of child node, while the hierarchy of H-matrices has still not been extended yet.

If the associated block cluster node of the current $\mathcal{H}$-matrix node has children, we firstly update the $\mathcal{H}$-matrix type for the current $\mathcal{H}$-matrix node as `HierarchicalMatrix` and this is only performed when the current $\mathcal{H}$-matrix node is not the starting $\mathcal{H}$-matrix node, because the original matrix type of the starting $\mathcal{H}$-matrix node will be used later during restriction operations to the current block cluster.

For each of the children, create an empty $\mathcal{H}$-matrix node on the heap and append it to the list of submatrices of the current $\mathcal{H}$-matrix.

Link the child $\mathcal{H}$-matrix node with the corresponding block cluster node.

Link row and column indices of the child $\mathcal{H}$-matrix node to those index sets stored in clusters.

Update the matrix dimension of the child $\mathcal{H}$-matrix node.

Recursively call the function.

When the current $\mathcal{H}$-matrix node has no children, i.e. it belongs to the leaf set of the extended block cluster tree.

If the current $\mathcal{H}$-matrix node is still the same as the starting $\mathcal{H}$-matrix node, there is no actual refinement work to be done.

```
If the current \form#17-matrix node is not the
```

starting $\mathcal{H}$-matrix node, we firstly build the maps from global row and column indices to their respective local indices. When there will be further refinement from those nodes, these maps will be used for matrix restriction.

Note These two global to local maps are only needed for $\mathcal{H}$-matrix nodes in the leaf set, because only these $\mathcal{H}$-matrix nodes contain the actual full matrix data or rank-k matrix data.

Update the current $\mathcal{H}$-matrix node type according to the identity of the block cluster node: when the block cluster belongs to the near field, `current_hmat` should be represented as a full matrix `LAPACKFull↩ MatrixExt`; when the block cluster belongs to the far field, `current_hmat` should be represented as a rank-k matrix `RkMatrix`. Correspondingly, new matrices, either full matrix or rank-k matrix will be created on the heap and assigned to the corresponding field of the current $\mathcal{H}$-matrix.

Fill the current full matrix with the data extracted from the starting $\mathcal{H}$-matrix node. This is actually a restriction of the starting $\mathcal{H}$-matrix node to the current $\mathcal{H}$-matrix node.

Fill the current rank-k matrix with the data extracted from the starting $\mathcal{H}$-matrix node.

References HMatrix< spacedim, Number >::bc_node, HMatrix< spacedim, Number >::col_index_global_to_↩ local_map, HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::fullmatrix, FullMatrix↩ Type, HierarchicalMatrixType, HMatrix< spacedim, Number >::m, HMatrix< spacedim, Number >::n, HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_index_global_to_local_↩ map, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::submatrices, and HMatrix< spacedim, Number >::type.

9.7.3.22 rk_h_mmult()

```
template<int spacedim, typename Number = double>
void rk_h_mmult (
            const RkMatrix< Number > & M1,
```

```
HMatrix< spacedim, Number > & M2,
RkMatrix< Number > & M )
```

Calculate the product of two $\mathcal{H}$-matrix nodes, where the first one `M1` has `RkMatrixType` and the result is also a rank-k matrix.

The arithmetic operation to be performed is

$$M = M_1 \cdot M_2 = (AB^T)M_2 = A(B^T M_2) = AB'^T,$$

where $B' = M_2^T B$ is calculated as a series of transposed $\mathcal{H}$-matrix-vector multiplications. For details,

$$M_2^T B = M_2^T \begin{bmatrix} b_{\sigma,1} & \cdots & b_{\sigma,r} \end{bmatrix} = \begin{bmatrix} M_2^T b_{\sigma,1} & \cdots & M_2^T b_{\sigma,r} \end{bmatrix} = \begin{bmatrix} b'_{\rho,1} & \cdots & b'_{\rho,r} \end{bmatrix}.$$

It can be seen that the formal rank r of the result matrix `M` is the same as that of `M1`.

Parameters

M1	
M2	
M	

Create a temporary `Vector` storing a column $b_{\sigma,j}$ in the `B` component of `M1` and another `Vector` $b'_{\rho,j}$ storing the matrix-vector product $M_2^T \cdot b_{\sigma,j}$.

Initialize the result rank-k matrix `M` with the formal rank of `M1_rk`. Its `A` component matrix is the same as that of `M1_rk`.

Build the map from global DoF indices to local matrix indices if necessary.

Then we calculate the `B` component matrix of `M`, which is `M2^T*M1_rk`.B.

Fill the result vector into the `B` component matrix of `M`.

References build_index_set_global_to_local_map(), HMatrix< spacedim, Number >::col_index_global_to_local↩_map, HMatrix< spacedim, Number >::col_indices, RkMatrix< Number >::formal_rank, RkMatrix< Number >↩::m, HMatrix< spacedim, Number >::m, RkMatrix< Number >::n, HMatrix< spacedim, Number >::n, RkMatrix< Number >::reinit(), HMatrix< spacedim, Number >::row_index_global_to_local_map, HMatrix< spacedim, Number >::row_indices, and HMatrix< spacedim, Number >::Tvmult_local_vector().

Referenced by f_h_mmult(), main(), and rk_h_mmult_for_h_h_mmult().

9.7.3.23 rk_h_mmult_for_h_h_mmult()

```
template<int spacedim, typename Number = double>
void rk_h_mmult_for_h_h_mmult (
            const HMatrix< spacedim, Number > * M1,
            HMatrix< spacedim, Number > * M2,
            HMatrix< spacedim, Number > * M,
            bool is_M1M2_last_in_M_Sigma_P = true )
```

Calculate the product of two $\mathcal{H}$-matrix nodes, where the first one `M1` has `RkMatrixType` and the result will also be a rank-k matrix. This function is to be called by the matrix-matrix multiplication function.

Parameters

M1	
M2	
M	

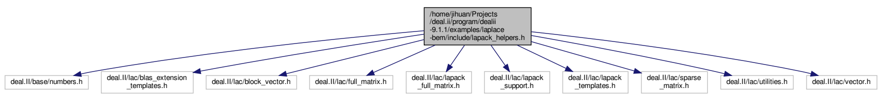
References HMatrix< spacedim, Number >::col_indices, HMatrix< spacedim, Number >::remove_hmat_↩ pair_from_mm_product_list(), rk_h_mmult(), HMatrix< spacedim, Number >::rkmatrix, RkMatrixType, HMatrix< spacedim, Number >::row_indices, HMatrix< spacedim, Number >::Sigma_F, HMatrix< spacedim, Number >::↩ Sigma_P, HMatrix< spacedim, Number >::Sigma_R, and HMatrix< spacedim, Number >::type.

9.8 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/lapack↩ _helpers.h File Reference

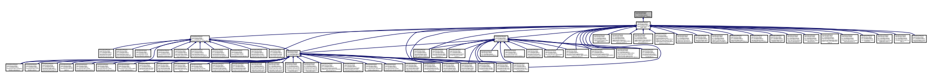
Exposes LAPACK helper functions defined in lapack_full_matrix.cc and define new ones by following them as examples.

```
#include <deal.II/base/numbers.h>
#include <deal.II/lac/blas_extension_templates.h>
#include <deal.II/lac/block_vector.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/lac/lapack_full_matrix.h>
#include <deal.II/lac/lapack_support.h>
#include <deal.II/lac/lapack_templates.h>
#include <deal.II/lac/sparse_matrix.h>
#include <deal.II/lac/utilities.h>
#include <deal.II/lac/vector.h>
```

Include dependency graph for lapack_helpers.h:



This graph shows which files directly or indirectly include this file:



Functions

- template<typename T >
  void **internal::LAPACKFullMatrixImplementation::geev_helper** (const char vl, const char vr, Aligned↩ Vector< T > &matrix, const types::blas_int n_rows, std::vector< T > &real_part_eigenvalues, std::vector< T > &imag_part_eigenvalues, std::vector< T > &left_eigenvectors, std::vector< T > &right_eigenvectors, std::vector< T > &real_work, std::vector< T > &, const types::blas_int work_flag, types::blas_int &info)

- template<typename T >
  void **internal::LAPACKFullMatrixImplementation::geev_helper** (const char vl, const char vr, AlignedVector< std::complex< T >> &matrix, const types::blas_int n_rows, std::vector< T > &, std::vector< std::complex< T >> &eigenvalues, std::vector< std::complex< T >> &left_eigenvectors, std::vector< std::complex< T >> &right_eigenvectors, std::vector< std::complex< T >> &complex_work, std::vector< T > &real_work, const types::blas_int work_flag, types::blas_int &info)
- template<typename T >
  void internal::LAPACKFullMatrixImplementation::gesdd_helper (const char job, const types::blas_int n_rows, const types::blas_int n_cols, AlignedVector< T > &matrix, std::vector< T > &singular_values, AlignedVector< T > &left_vectors, AlignedVector< T > &right_vectors, std::vector< T > &real_work, std::vector< T > &, std::vector< types::blas_int > &integer_work, const types::blas_int work_flag, types::blas_int &info)
- template<typename T >
  void internal::LAPACKFullMatrixImplementation::gesdd_helper (const char job, const types::blas_int n_rows, const types::blas_int n_cols, AlignedVector< std::complex< T >> &matrix, std::vector< T > &singular_values, AlignedVector< std::complex< T >> &left_vectors, AlignedVector< std::complex< T >> &right_vectors, std::vector< std::complex< T >> &work, std::vector< T > &real_work, std::vector< types::blas_int > &integer_work, const types::blas_int &work_flag, types::blas_int &info)
- template<typename T >
  void internal::LAPACKFullMatrixImplementation::geqrf_helper (const types::blas_int n_rows, const types::blas_int n_cols, AlignedVector< T > &matrix, std::vector< T > &tau, std::vector< T > &work, const types::blas_int work_flag, types::blas_int &info)

9.8.1 Detailed Description

Exposes LAPACK helper functions defined in lapack_full_matrix.cc and define new ones by following them as examples.

Date

2021-06-09

Author

Jihuan Tian

9.8.2 Function Documentation

9.8.2.1 geqrf_helper()

```
template<typename T >
void internal::LAPACKFullMatrixImplementation::geqrf_helper (
            const types::blas_int n_rows,
            const types::blas_int n_cols,
            AlignedVector< T > & matrix,
            std::vector< T > & tau,
            std::vector< T > & work,
            const types::blas_int work_flag,
            types::blas_int & info )
```

Helper function for real valued QR decomposition.

References internal::LAPACKFullMatrixImplementation::geqrf_helper().

Referenced by internal::LAPACKFullMatrixImplementation::geqrf_helper().

9.8.2.2 gesdd_helper() [1/2]

```
template<typename T >
void internal::LAPACKFullMatrixImplementation::gesdd_helper (
            const char job,
            const types::blas_int n_rows,
            const types::blas_int n_cols,
            AlignedVector< T > & matrix,
            std::vector< T > & singular_values,
            AlignedVector< T > & left_vectors,
            AlignedVector< T > & right_vectors,
            std::vector< T > & real_work,
            std::vector< T > & ,
            std::vector< types::blas_int > & integer_work,
            const types::blas_int work_flag,
            types::blas_int & info )
```

Helper function for real valued SVD.

Its implementation has already existed in the deal.ii library.

Parameters

job	
n_rows	
n_cols	
matrix	
singular_values	
left_vectors	
right_vectors	
real_work	

9.8.2.3 gesdd_helper() [2/2]

```
template<typename T >
void internal::LAPACKFullMatrixImplementation::gesdd_helper (
            const char job,
            const types::blas_int n_rows,
            const types::blas_int n_cols,
            AlignedVector< std::complex< T >> & matrix,
            std::vector< T > & singular_values,
            AlignedVector< std::complex< T >> & left_vectors,
            AlignedVector< std::complex< T >> & right_vectors,
            std::vector< std::complex< T >> & work,
            std::vector< T > & real_work,
            std::vector< types::blas_int > & integer_work,
            const types::blas_int & work_flag,
            types::blas_int & info )
```

Helper function for complex valued SVD.

Its implementation has already existed in the deal.ii library.

Parameters

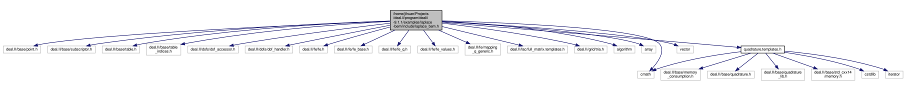
job	
n_rows	
n_cols	
matrix	
singular_values	
left_vectors	
right_vectors	
work	
real_work	
integer_work	
work_flag	
info	

9.9 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/laplace_bem.h File Reference

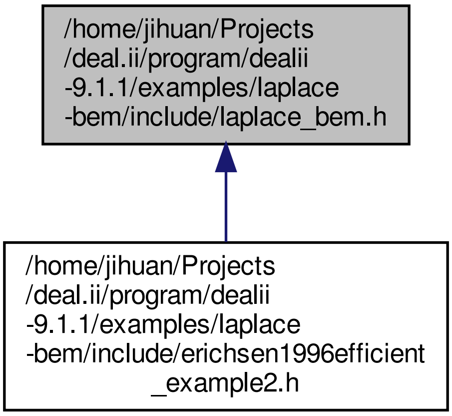
Implementation of BEM involving kernel functions and singular numerical quadratures.

```
#include <deal.II/base/point.h>
#include <deal.II/base/subscriptor.h>
#include <deal.II/base/table.h>
#include <deal.II/base/table_indices.h>
#include <deal.II/dofs/dof_accessor.h>
#include <deal.II/dofs/dof_handler.h>
#include <deal.II/fe/fe.h>
#include <deal.II/fe/fe_base.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/fe/fe_values.h>
#include <deal.II/fe/mapping_q_generic.h>
#include <deal.II/lac/full_matrix.templates.h>
#include <deal.II/grid/tria.h>
#include <algorithm>
#include <array>
#include <cmath>
#include <vector>
#include "quadrature.templates.h"
```

Include dependency graph for laplace_bem.h:



This graph shows which files directly or indirectly include this file:



Classes

- class LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >
- class LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >
- struct LaplaceBEM::CellWisePerTaskData
- struct LaplaceBEM::CellWiseScratchData
- struct LaplaceBEM::PairCellWiseScratchData
- struct LaplaceBEM::PairCellWisePerTaskData
- class LaplaceBEM::LaplaceKernel::KernelFunction< dim, RangeNumberType >
- class LaplaceBEM::LaplaceKernel::SingleLayerKernel< dim, RangeNumberType >
- class LaplaceBEM::LaplaceKernel::DoubleLayerKernel< dim, RangeNumberType >
- class LaplaceBEM::LaplaceKernel::AdjointDoubleLayerKernel< dim, RangeNumberType >
- class LaplaceBEM::LaplaceKernel::HyperSingularKernel< dim, RangeNumberType >
- class LaplaceBEM::KernelPulledbackToUnitCell< dim, spacedim, RangeNumberType >
- class LaplaceBEM::KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType >

Enumerations

- enum **CellNeighboringType** {
  SamePanel, **CommonEdge**, **CommonVertex**, **Regular**,
  None }
- enum **KernelType** {
  SingleLayer, **DoubleLayer**, **AdjointDoubleLayer**, **HyperSingular**,
  NoneType }

Functions

- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_values_same_panel (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 3, RangeNumberType > &kx_shape_value_table, Table< 3, RangeNumberType > &ky_shape_value_table)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_values_common_edge (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 3, RangeNumberType > &kx_shape_value_table, Table< 3, RangeNumberType > &ky_shape_value_table)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_values_common_vertex (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 3, RangeNumberType > &kx_shape_value_table, Table< 3, RangeNumberType > &ky_shape_value_table)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_values_regular (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 3, RangeNumberType > &kx_shape_value_table, Table< 3, RangeNumberType > &ky_shape_value_table)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_grad_matrices_same_panel (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 2, FullMatrix< RangeNumberType >> &kx_shape_grad_matrix_table, Table< 2, FullMatrix< RangeNumberType >> &ky_shape_grad_matrix_table)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_grad_matrices_common_edge (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 2, FullMatrix< RangeNumberType >> &kx_shape_grad_matrix_table, Table< 2, FullMatrix< RangeNumberType >> &ky_shape_grad_matrix_table)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_grad_matrices_common_vertex (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 2, FullMatrix< RangeNumberType >> &kx_shape_grad_matrix_table, Table< 2, FullMatrix< RangeNumberType >> &ky_shape_grad_matrix_table)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::bem_shape_grad_matrices_regular (const FiniteElement< dim, spacedim > &kx_fe, const FiniteElement< dim, spacedim > &ky_fe, const QGauss< 4 > &sauter_quad_rule, Table< 2, FullMatrix< RangeNumberType >> &kx_shape_grad_matrix_table, Table< 2, FullMatrix< RangeNumberType >> &ky_shape_grad_matrix_table)
- template<int dim, int spacedim>
  std::vector< types::global_dof_index > **LaplaceBEM::get_conflict_indices** (const typename DoFHandler< dim, spacedim >::active_cell_iterator &cell)
- template<int dim, int spacedim>
  std::array< types::global_vertex_index, GeometryInfo< dim >::vertices_per_cell > **LaplaceBEM::get_vertex_indices** (const typename Triangulation< dim, spacedim >::cell_iterator &cell)
- template<int dim, int spacedim>
  std::array< types::global_dof_index, GeometryInfo< dim >::vertices_per_cell > **LaplaceBEM::get_vertex_dof_indices** (const typename DoFHandler< dim, spacedim >::cell_iterator &cell)
- template<int dim>
  CellNeighboringType **LaplaceBEM::detect_cell_neighboring_type** (const std::array< types::global_vertex_index, GeometryInfo< dim >::vertices_per_cell > &first_cell_vertex_indices, const std::array< types::global_vertex_index, GeometryInfo< dim >::vertices_per_cell > &second_cell_vertex_indices, std::vector< types::global_vertex_index > &vertex_index_intersection)
- template<int dim>
  CellNeighboringType **LaplaceBEM::detect_cell_neighboring_type** (const std::array< types::global_dof_index, GeometryInfo< dim >::vertices_per_cell > &first_cell_vertex_dof_indices, const std::array< types::global_dof_index, GeometryInfo< dim >::vertices_per_cell > &second_cell_vertex_dof_indices, std::vector< types::global_dof_index > &vertex_dof_index_intersection)

- template<int dim, int spacedim, typename Number = double>
  Number **LaplaceBEM::cell_distance** (const typename Triangulation< dim, spacedim >::cell_iterator first_cell, const typename Triangulation< dim, spacedim >::cell_iterator second_cell)
- template<int dim, int spacedim>
  FullMatrix< double > **LaplaceBEM::shape_grad_matrix** (const FiniteElement< dim, spacedim > &fe, const std::vector< unsigned int > &dof_permutation, const Point< dim > &p)
- template<int dim, int spacedim>
  FullMatrix< double > **LaplaceBEM::shape_grad_matrix_in_hierarchical_order** (const FiniteElement< dim, spacedim > &fe, const Point< dim > &p)
- template<int dim, int spacedim>
  FullMatrix< double > **LaplaceBEM::shape_grad_matrix_in_tensor_product_order** (const FiniteElement< dim, spacedim > &fe, const Point< dim > &p)
- template<int dim, int spacedim>
  Vector< double > **LaplaceBEM::shape_values** (const FiniteElement< dim, spacedim > &fe, const std::vector< unsigned int > &dof_permutation, const Point< dim > &p)
- template<int dim, int spacedim>
  Vector< double > **LaplaceBEM::shape_values_in_hierarchical_order** (const FiniteElement< dim, spacedim > &fe, const Point< dim > &p)
- template<int dim, int spacedim>
  Vector< double > **LaplaceBEM::shape_values_in_tensor_product_order** (const FiniteElement< dim, spacedim > &fe, const Point< dim > &p)
- FullMatrix< double > **LaplaceBEM::collect_two_components_from_point3** (const std::vector< Point< 3 >> &points, const unsigned int first_component, const unsigned int second_component)
- template<int dim, int spacedim>
  std::vector< Point< spacedim > > **LaplaceBEM::support_points_in_real_cell** (const typename Triangulation< dim, spacedim >::cell_iterator &cell, const FiniteElement< dim, spacedim > &fe, const MappingQGeneric< dim, spacedim > &mapping, const std::vector< unsigned int > &dof_permutation)
- template<int dim, int spacedim>
  std::vector< Point< spacedim > > LaplaceBEM::hierarchical_support_points_in_real_cell (const typename Triangulation< dim, spacedim >::cell_iterator &cell, const FiniteElement< dim, spacedim > &fe, const MappingQGeneric< dim, spacedim > &mapping)
- template<int dim, int spacedim>
  void LaplaceBEM::hierarchical_support_points_in_real_cell (const typename Triangulation< dim, spacedim >::cell_iterator &cell, const FiniteElement< dim, spacedim > &fe, const MappingQGeneric< dim, spacedim > &mapping, std::vector< Point< spacedim >> &real_support_points)
- template<int dim, int spacedim>
  std::vector< Point< spacedim > > **LaplaceBEM::tensor_product_support_points_in_real_cell** (const typename Triangulation< dim, spacedim >::cell_iterator &cell, const FiniteElement< dim, spacedim > &fe, const MappingQGeneric< dim, spacedim > &mapping)
- template<int dim, int spacedim>
  double LaplaceBEM::surface_jacobian_det (const FiniteElement< dim, spacedim > &fe, const std::vector< Point< spacedim >> &support_points_in_real_cell, const Point< dim > &p)
- template<int spacedim>
  double LaplaceBEM::surface_jacobian_det (const unsigned int k3_index, const unsigned int quad_no, const Table< 2, FullMatrix< double >> &shape_grad_matrix_table, const std::vector< Point< spacedim >> &support_points_in_real_cell)
- template<int dim, int spacedim>
  double **LaplaceBEM::surface_jacobian_det_and_normal_vector** (const FiniteElement< dim, spacedim > &fe, const std::vector< Point< spacedim >> &support_points_in_real_cell, const Point< dim > &p, Tensor< 1, spacedim > &normal_vector)
- template<int spacedim>
  double **LaplaceBEM::surface_jacobian_det_and_normal_vector** (const unsigned int k3_index, const unsigned int quad_no, const Table< 2, FullMatrix< double >> &shape_grad_matrix_table, const std::vector< Point< spacedim >> &support_points_in_real_cell, Tensor< 1, spacedim > &normal_vector)
- template<int dim, int spacedim>
  Point< spacedim > LaplaceBEM::transform_unit_to_permuted_real_cell (const FiniteElement< dim, spacedim > &fe, const std::vector< Point< spacedim >> &support_points_in_real_cell, const Point< dim > &area_coords)

- template< int spacedim >
  Point< spacedim > LaplaceBEM::transform_unit_to_permuted_real_cell (const unsigned int k3_index, const unsigned int quad_no, const Table< 3, double > &shape_value_table, const std::vector< Point< spacedim >> &support_points_in_real_cell)
- template< int dim, int spacedim >
  std::vector< unsigned int > LaplaceBEM::generate_forward_dof_permutation (const FiniteElement< dim, spacedim > &fe, unsigned int starting_corner)
- template< int dim, int spacedim >
  void LaplaceBEM::generate_forward_dof_permutation (const FiniteElement< dim, spacedim > &fe, unsigned int starting_corner, std::vector< unsigned int > &dof_permutation)
- template< int dim, int spacedim >
  std::vector< unsigned int > LaplaceBEM::generate_backward_dof_permutation (const FiniteElement< dim, spacedim > &fe, unsigned int starting_corner)
- template< int dim, int spacedim >
  void LaplaceBEM::generate_backward_dof_permutation (const FiniteElement< dim, spacedim > &fe, unsigned int starting_corner, std::vector< unsigned int > &dof_permutation)
- template<typename T >
  std::vector< T > **LaplaceBEM::permute_vector** (const std::vector< T > &input_vector, const std::vector< unsigned int > &permutation_indices)
- template<typename T >
  void LaplaceBEM::permute_vector (const std::vector< T > &input_vector, const std::vector< unsigned int > &permutation_indices, std::vector< T > &permuted_vector)
- template<int dim, int spacedim, typename RangeNumberType = double>
  RangeNumberType **LaplaceBEM::ApplyQuadrature** (const Quadrature< dim *2 > &quad_rule, const KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType > &f, unsigned int component=0)
- template<int dim, int spacedim, typename RangeNumberType = double>
  RangeNumberType **LaplaceBEM::ApplyQuadratureUsingBEMValues** (const Quadrature< dim *2 > &quad_rule, const KernelPulledbackToSauterSpace< dim, spacedim, RangeNumberType > &f, unsigned int component=0)
- template<int dim, int spacedim>
  std::array< types::global_dof_index, GeometryInfo< dim >::vertices_per_cell > LaplaceBEM::get_vertex_dof_indices_swapped (const FiniteElement< dim, spacedim > &fe, const std::vector< types::global_dof_index > &dof_indices)
- template<int dim, int spacedim>
  void LaplaceBEM::get_vertex_dof_indices_swapped (const FiniteElement< dim, spacedim > &fe, const std::vector< types::global_dof_index > &dof_indices, std::array< types::global_dof_index, GeometryInfo< dim >::vertices_per_cell > &vertex_dof_indices)
- template<int vertices_per_cell>
  unsigned int LaplaceBEM::get_start_vertex_dof_index (const std::vector< types::global_dof_index > &vertex_dof_index_intersection, const std::array< types::global_dof_index, vertices_per_cell > &local_vertex_dof_indices_swapped)
- template<int dim, int spacedim, typename RangeNumberType = double>
  FullMatrix< RangeNumberType > LaplaceBEM::SauterQuadRule (const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > &kernel_function, const typename DoFHandler< dim, spacedim >::cell_iterator &kx_cell_iter, const typename DoFHandler< dim, spacedim >::cell_iterator &ky_cell_iter, const MappingQGeneric< dim, spacedim > &kx_mapping=MappingQGeneric< dim, spacedim >(1), const MappingQGeneric< dim, spacedim > &ky_mapping=MappingQGeneric< dim, spacedim >(1))
- template<int dim, int spacedim, typename RangeNumberType = double>
  FullMatrix< RangeNumberType > **LaplaceBEM::SauterQuadRule** (const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > &kernel_function, const BEMValues< dim, spacedim > &bem_values, const typename DoFHandler< dim, spacedim >::cell_iterator &kx_cell_iter, const typename DoFHandler< dim, spacedim >::cell_iterator &ky_cell_iter, const MappingQGeneric< dim, spacedim > &kx_mapping=MappingQGeneric< dim, spacedim >(1), const MappingQGeneric< dim, spacedim > &ky_mapping=MappingQGeneric< dim, spacedim >(1))
- template<int dim>
  void LaplaceBEM::sauter_same_panel_parametric_coords_to_unit_cells (const Point< dim *2 > &parametric_coords, const unsigned int k3_index, Point< dim > &kx_unit_cell_coords, Point< dim > &ky_unit_cell_coords)

- template<int dim>
  void LaplaceBEM::sauter_common_edge_parametric_coords_to_unit_cells (const Point< dim *2 > &parametric_coords, const unsigned int k3_index, Point< dim > &kx_unit_cell_coords, Point< dim > &ky_unit_cell_coords)
- template<int dim>
  void LaplaceBEM::sauter_common_vertex_parametric_coords_to_unit_cells (const Point< dim *2 > &parametric_coords, const unsigned int k3_index, Point< dim > &kx_unit_cell_coords, Point< dim > &ky_unit_cell_coords)
- template<int dim>
  void LaplaceBEM::sauter_regular_parametric_coords_to_unit_cells (const Point< dim *2 > &parametric_coords, Point< dim > &kx_unit_cell_coords, Point< dim > &ky_unit_cell_coords)
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::SauterQuadRule (FullMatrix< RangeNumberType > &system_matrix, const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > &kernel_function, const typename DoFHandler< dim, spacedim >::cell_iterator &kx_cell_iter, const typename DoFHandler< dim, spacedim >::cell_iterator &ky_cell_iter, const MappingQGeneric< dim, spacedim > &kx_mapping=MappingQGeneric< dim, spacedim >(1), const MappingQGeneric< dim, spacedim > &ky_mapping=MappingQGeneric< dim, spacedim >(1))
- template<int dim, int spacedim, typename RangeNumberType = double>
  void LaplaceBEM::SauterQuadRule (FullMatrix< RangeNumberType > &system_matrix, const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > &kernel_function, const BEMValues< dim, spacedim > &bem_values, const typename DoFHandler< dim, spacedim >::cell_iterator &kx_cell_iter, const typename DoFHandler< dim, spacedim >::cell_iterator &ky_cell_iter, const MappingQGeneric< dim, spacedim > &kx_mapping=MappingQGeneric< dim, spacedim >(1), const MappingQGeneric< dim, spacedim > &ky_mapping=MappingQGeneric< dim, spacedim >(1))

9.9.1 Detailed Description

Implementation of BEM involving kernel functions and singular numerical quadratures.

Date

2020-11-02

Author

Jihuan Tian

9.9.2 Function Documentation

9.9.2.1 bem_shape_grad_matrices_common_edge()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_grad_matrices_common_edge (
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const QGauss< 4 > & sauter_quad_rule,
            Table< 2, FullMatrix< RangeNumberType >> & kx_shape_grad_matrix_table,
            Table< 2, FullMatrix< RangeNumberType >> & ky_shape_grad_matrix_table )
```

Calculate the table storing shape function gradient matrices at Sauter quadrature points for the common edge case. N.B. The shape functions are in the tensor product order and each row of the gradient matrix corresponds to a shape function.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_grad_matrix_table	the 1st dimension is the index for k_3 terms; the 2nd dimension is the quadrature point number.
ky_shape_grad_matrix_table	the 1st dimension is the index for k_3 terms; the 2nd dimension is the quadrature point number.

References LaplaceBEM::bem_shape_grad_matrices_common_edge(), and LaplaceBEM::sauter_common_edge_parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_common_edge(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.2 bem_shape_grad_matrices_common_vertex()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_grad_matrices_common_vertex (
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const QGauss< 4 > & sauter_quad_rule,
            Table< 2, FullMatrix< RangeNumberType >> & kx_shape_grad_matrix_table,
            Table< 2, FullMatrix< RangeNumberType >> & ky_shape_grad_matrix_table )
```

Calculate the table storing shape function gradient matrices at Sauter quadrature points for the common vertex case. N.B. The shape functions are in the tensor product order and each row of the gradient matrix corresponds to a shape function.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_grad_matrix_table	the 1st dimension is the index for k_3 terms; the 2nd dimension is the quadrature point number.
ky_shape_grad_matrix_table	the 1st dimension is the index for k_3 terms; the 2nd dimension is the quadrature point number.

References LaplaceBEM::bem_shape_grad_matrices_common_vertex(), and LaplaceBEM::sauter_common_vertex_parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_common_vertex(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.3 bem_shape_grad_matrices_regular()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_grad_matrices_regular (
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const QGauss< 4 > & sauter_quad_rule,
            Table< 2, FullMatrix< RangeNumberType >> & kx_shape_grad_matrix_table,
            Table< 2, FullMatrix< RangeNumberType >> & ky_shape_grad_matrix_table )
```

Calculate the table storing shape function gradient matrices at Sauter quadrature points for the regular case. N.B. The shape functions are in the tensor product order and each row of the gradient matrix corresponds to a shape function.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_grad_matrix_table	the 1st dimension is the quadrature point number.
ky_shape_grad_matrix_table	the 1st dimension is the quadrature point number.

References LaplaceBEM::bem_shape_grad_matrices_regular(), and LaplaceBEM::sauter_regular_parametric_↩ coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_regular(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.4 bem_shape_grad_matrices_same_panel()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_grad_matrices_same_panel (
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const QGauss< 4 > & sauter_quad_rule,
            Table< 2, FullMatrix< RangeNumberType >> & kx_shape_grad_matrix_table,
            Table< 2, FullMatrix< RangeNumberType >> & ky_shape_grad_matrix_table )
```

Calculate the table storing shape function gradient matrices at Sauter quadrature points for the same panel case. N.B. The shape functions are in the tensor product order and each row of the gradient matrix corresponds to a shape function.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_grad_matrix_table	the 1st dimension is the index for k_3 terms; the 2nd dimension is the quadrature point number.
ky_shape_grad_matrix_table	the 1st dimension is the index for k_3 terms; the 2nd dimension is the quadrature point number.

References LaplaceBEM::bem_shape_grad_matrices_same_panel(), and LaplaceBEM::sauter_same_panel_←
parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_same_panel(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.5 bem_shape_values_common_edge()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_values_common_edge (
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const QGauss< 4 > & sauter_quad_rule,
            Table< 3, RangeNumberType > & kx_shape_value_table,
            Table< 3, RangeNumberType > & ky_shape_value_table )
```

Calculate the table storing shape function values at Sauter quadrature points for the common edge case.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.
ky_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.

References LaplaceBEM::bem_shape_values_common_edge(), and LaplaceBEM::sauter_common_edge_←
parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_values_common_edge(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.6 bem_shape_values_common_vertex()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_values_common_vertex (
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const QGauss< 4 > & sauter_quad_rule,
            Table< 3, RangeNumberType > & kx_shape_value_table,
            Table< 3, RangeNumberType > & ky_shape_value_table )
```

Calculate the table storing shape function values at Sauter quadrature points for the common vertex case.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.
ky_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.

References LaplaceBEM::bem_shape_values_common_vertex(), and LaplaceBEM::sauter_common_vertex_↩ parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_values_common_vertex(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.7 bem_shape_values_regular()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_values_regular (
            const FiniteElement< dim, spacedim > & kx_fe,
            const FiniteElement< dim, spacedim > & ky_fe,
            const QGauss< 4 > & sauter_quad_rule,
            Table< 3, RangeNumberType > & kx_shape_value_table,
            Table< 3, RangeNumberType > & ky_shape_value_table )
```

Calculate the table storing shape function values at Sauter quadrature points for the regular case.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.
ky_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.

References LaplaceBEM::bem_shape_values_regular(), and LaplaceBEM::sauter_regular_parametric_coords_↩ to_unit_cells().

Referenced by LaplaceBEM::bem_shape_values_regular(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.8 bem_shape_values_same_panel()

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::bem_shape_values_same_panel (
```

```
const FiniteElement< dim, spacedim > & kx_fe,
const FiniteElement< dim, spacedim > & ky_fe,
const QGauss< 4 > & sauter_quad_rule,
Table< 3, RangeNumberType > & kx_shape_value_table,
Table< 3, RangeNumberType > & ky_shape_value_table )
```

Calculate the table storing shape function values at Sauter quadrature points for the same panel case.

Parameters

kx_fe	finite element for K_x
ky_fe	finite element for K_y
sauter_quad_rule	
kx_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.
ky_shape_value_table	the 1st dimension is the shape function hierarchical numbering; the 2nd dimension is the index for k_3 terms; the 3rd dimension is the quadrature point number.

References LaplaceBEM::bem_shape_values_same_panel(), and LaplaceBEM::sauter_same_panel_parametric↵_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_values_same_panel(), and LaplaceBEM::BEMValues< dim, spacedim, RangeNumberType >::BEMValues().

9.9.2.9 generate_backward_dof_permutation() [1/2]

```
template<int dim, int spacedim>
std::vector<unsigned int> LaplaceBEM::generate_backward_dof_permutation (
            const FiniteElement< dim, spacedim > & fe,
            unsigned int starting_corner )
```

Generate the permutation of polynomial space inverse numbering by starting from the specified corner in the backward direction. The index for starting_corner starts from zero.

References LaplaceBEM::generate_backward_dof_permutation().

Referenced by LaplaceBEM::generate_backward_dof_permutation().

9.9.2.10 generate_backward_dof_permutation() [2/2]

```
template<int dim, int spacedim>
void LaplaceBEM::generate_backward_dof_permutation (
            const FiniteElement< dim, spacedim > & fe,
            unsigned int starting_corner,
            std::vector< unsigned int > & dof_permutation )
```

Generate the permutation of polynomial space inverse numbering by starting from the specified corner in the backward direction. The index for starting_corner starts from zero. This overloaded version has the returned vector as its argument.

References LaplaceBEM::generate_backward_dof_permutation().

9.9.2.11 generate_forward_dof_permutation() [1/2]

```
template<int dim, int spacedim>
std::vector<unsigned int> LaplaceBEM::generate_forward_dof_permutation (
            const FiniteElement< dim, spacedim > & fe,
            unsigned int starting_corner )
```

Generate the permutation of polynomial space inverse numbering by starting from the specified corner in the forward direction.

References LaplaceBEM::generate_forward_dof_permutation().

Referenced by LaplaceBEM::generate_forward_dof_permutation().

9.9.2.12 generate_forward_dof_permutation() [2/2]

```
template<int dim, int spacedim>
void LaplaceBEM::generate_forward_dof_permutation (
            const FiniteElement< dim, spacedim > & fe,
            unsigned int starting_corner,
            std::vector< unsigned int > & dof_permutation )
```

Generate the permutation of polynomial space inverse numbering by starting from the specified corner in the forward direction. This overloaded version has the returned vector as its argument.

References LaplaceBEM::generate_forward_dof_permutation().

9.9.2.13 get_start_vertex_dof_index()

```
template<int vertices_per_cell>
unsigned int LaplaceBEM::get_start_vertex_dof_index (
            const std::vector< types::global_dof_index > & vertex_dof_index_intersection,
            const std::array< types::global_dof_index, vertices_per_cell > & local_vertex_↩
dof_indices_swapped )
```

Get the starting vertex DoF index.

References LaplaceBEM::get_start_vertex_dof_index().

Referenced by LaplaceBEM::get_start_vertex_dof_index().

9.9.2.14 get_vertex_dof_indices_swapped() [1/2]

```
template<int dim, int spacedim>
std::array<types::global_dof_index, GeometryInfo<dim>::vertices_per_cell> LaplaceBEM::get_↩
vertex_dof_indices_swapped (
            const FiniteElement< dim, spacedim > & fe,
            const std::vector< types::global_dof_index > & dof_indices )
```

Get the DoF vertex indices from a list of DoF indices which have been arranged in forward or backward tensor product ordering. N.B. There are `GeometryInfo<dim>::vertices_per_cell` vertices in the returned array, among which the last two DoF vertex indices have been swapped so that the vertex DoFs are arranged in clockwise or counter clockwise order instead of the zigzag order.

References LaplaceBEM::get_vertex_dof_indices_swapped().

Referenced by LaplaceBEM::get_vertex_dof_indices_swapped().

9.9.2.15 get_vertex_dof_indices_swapped() [2/2]

```
template<int dim, int spacedim>
void LaplaceBEM::get_vertex_dof_indices_swapped (
            const FiniteElement< dim, spacedim > & fe,
            const std::vector< types::global_dof_index > & dof_indices,
            std::array< types::global_dof_index, GeometryInfo< dim >::vertices_per_cell > &
vertex_dof_indices )
```

Get the DoF vertex indices from a list of DoF indices which have been arranged in forward or backward tensor product ordering. N.B. There are `GeometryInfo<dim>::vertices_per_cell` vertices in the returned array, among which the last two DoF vertex indices have been swapped so that the vertex DoFs are arranged in clockwise or counter clockwise order instead of the zigzag order.

References LaplaceBEM::get_vertex_dof_indices_swapped().

9.9.2.16 hierarchical_support_points_in_real_cell() [1/2]

```
template<int dim, int spacedim>
std::vector<Point<spacedim> > LaplaceBEM::hierarchical_support_points_in_real_cell (
            const typename Triangulation< dim, spacedim >::cell_iterator & cell,
            const FiniteElement< dim, spacedim > & fe,
            const MappingQGeneric< dim, spacedim > & mapping )
```

Calculate support points in real cell in the default hierarchical ordering.

References LaplaceBEM::hierarchical_support_points_in_real_cell().

Referenced by LaplaceBEM::Erichsen1996Efficient::Example2::assemble_on_one_pair_of_cells(), LaplaceBEM↩
::hierarchical_support_points_in_real_cell(), and LaplaceBEM::SauterQuadRule().

9.9.2.17 hierarchical_support_points_in_real_cell() [2/2]

```
template<int dim, int spacedim>
void LaplaceBEM::hierarchical_support_points_in_real_cell (
            const typename Triangulation< dim, spacedim >::cell_iterator & cell,
            const FiniteElement< dim, spacedim > & fe,
            const MappingQGeneric< dim, spacedim > & mapping,
            std::vector< Point< spacedim >> & real_support_points )
```

Calculate support points in real cell in the default hierarchical ordering. The memory for storing these support points is pre-allocated.

References LaplaceBEM::hierarchical_support_points_in_real_cell().

9.9.2.18 permute_vector()

```
template<typename T >
void LaplaceBEM::permute_vector (
            const std::vector< T > & input_vector,
            const std::vector< unsigned int > & permutation_indices,
            std::vector< T > & permuted_vector )
```

Permute a vector according to the specified permutation indices. The memory for the output vector should be pre-allocated.

Parameters

input_vector	
permutation_indices	
permuted_vector	

9.9.2.19 sauter_common_edge_parametric_coords_to_unit_cells()

```
template<int dim>
void LaplaceBEM::sauter_common_edge_parametric_coords_to_unit_cells (
            const Point< dim *2 > & parametric_coords,
            const unsigned int k3_index,
            Point< dim > & kx_unit_cell_coords,
            Point< dim > & ky_unit_cell_coords )
```

Transform parametric coordinates in Sauter's quadrature rule for the common edge case to unit cell coordinates for K_x and K_y respectively.

Parameters

parametric_coords	
k3_index	
kx_unit_cell_coords	
ky_unit_cell_coords	

References LaplaceBEM::sauter_common_edge_parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_common_edge(), LaplaceBEM::bem_shape_values_↩
common_edge(), and LaplaceBEM::sauter_common_edge_parametric_coords_to_unit_cells().

9.9.2.20 sauter_common_vertex_parametric_coords_to_unit_cells()

```
template<int dim>
void LaplaceBEM::sauter_common_vertex_parametric_coords_to_unit_cells (
            const Point< dim *2 > & parametric_coords,
            const unsigned int k3_index,
            Point< dim > & kx_unit_cell_coords,
            Point< dim > & ky_unit_cell_coords )
```

Transform parametric coordinates in Sauter's quadrature rule for the common vertex case to unit cell coordinates for K_x and K_y respectively.

Parameters

parametric_coords	
k3_index	
kx_unit_cell_coords	
ky_unit_cell_coords	

References LaplaceBEM::sauter_common_vertex_parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_common_vertex(), LaplaceBEM::bem_shape_values_↩
common_vertex(), and LaplaceBEM::sauter_common_vertex_parametric_coords_to_unit_cells().

9.9.2.21 sauter_regular_parametric_coords_to_unit_cells()

```
template<int dim>
void LaplaceBEM::sauter_regular_parametric_coords_to_unit_cells (
            const Point< dim *2 > & parametric_coords,
            Point< dim > & kx_unit_cell_coords,
            Point< dim > & ky_unit_cell_coords )
```

Transform parametric coordinates in Sauter's quadrature rule for the regular case to unit cell coordinates for K_x and K_y respectively.

Parameters

parametric_coords	
k3_index	
kx_unit_cell_coords	
ky_unit_cell_coords	

References LaplaceBEM::sauter_regular_parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_regular(), LaplaceBEM::bem_shape_values_regular(), and LaplaceBEM::sauter_regular_parametric_coords_to_unit_cells().

9.9.2.22 sauter_same_panel_parametric_coords_to_unit_cells()

```
template<int dim>
void LaplaceBEM::sauter_same_panel_parametric_coords_to_unit_cells (
            const Point< dim *2 > & parametric_coords,
            const unsigned int k3_index,
            Point< dim > & kx_unit_cell_coords,
            Point< dim > & ky_unit_cell_coords )
```

Transform parametric coordinates in Sauter's quadrature rule for the same panel case to unit cell coordinates for K_x and K_y respectively.

Parameters

parametric_coords	
k3_index	
kx_unit_cell_coords	
ky_unit_cell_coords	

References LaplaceBEM::sauter_same_panel_parametric_coords_to_unit_cells().

Referenced by LaplaceBEM::bem_shape_grad_matrices_same_panel(), LaplaceBEM::bem_shape_values_↩ same_panel(), and LaplaceBEM::sauter_same_panel_parametric_coords_to_unit_cells().

9.9.2.23 SauterQuadRule() [1/3]

```
template<int dim, int spacedim, typename RangeNumberType = double>
FullMatrix<RangeNumberType> LaplaceBEM::SauterQuadRule (
            const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > & kernel_↩
function,
            const typename DoFHandler< dim, spacedim >::cell_iterator & kx_cell_iter,
            const typename DoFHandler< dim, spacedim >::cell_iterator & ky_cell_iter,
            const MappingQGeneric< dim, spacedim > & kx_mapping = MappingQGeneric<dim, spacedim>(1),
            const MappingQGeneric< dim, spacedim > & ky_mapping = MappingQGeneric<dim, spacedim>(1)
)
```

This function implements Sauter's quadrature rule on quadrangular mesh. It handles various cases including same panel, common edge, common vertex and regular cell neighboring types.

Parameters

kernel_function	Laplace kernel function.
kx_cell_iter	Iterator pointing to K_x.

Parameters

kx_cell_iter	Iterator pointing to K_y.
kx_mapping	Mapping used for K_x. Because a mesher usually generates 1st order grid, if there is no additional manifold specification, the mapping should be 1st order.
kx_mapping	Mapping used for K_y. Because a mesher usually generates 1st order grid, if there is no additional manifold specification, the mapping should be 1st order.

References LaplaceBEM::hierarchical_support_points_in_real_cell(), and LaplaceBEM::SauterQuadRule().

Referenced by LaplaceBEM::SauterQuadRule().

9.9.2.24 SauterQuadRule() [2/3]

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::SauterQuadRule (
            FullMatrix< RangeNumberType > & system_matrix,
            const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > & kernel_←
function,
            const typename DoFHandler< dim, spacedim >::cell_iterator & kx_cell_iter,
            const typename DoFHandler< dim, spacedim >::cell_iterator & ky_cell_iter,
            const MappingQGeneric< dim, spacedim > & kx_mapping = MappingQGeneric<dim, spacedim>(1),
            const MappingQGeneric< dim, spacedim > & ky_mapping = MappingQGeneric<dim, spacedim>(1)
)
```

This function implements Sauter's quadrature rule on quadrangular mesh. It handles various cases including same panel, common edge, common vertex and regular cell neighboring types.

Parameters

kernel_function	Laplace kernel function.
kx_cell_iter	Iterator pointing to K_x.
kx_cell_iter	Iterator pointing to K_y.
kx_mapping	Mapping used for K_x. Because a mesher usually generates 1st order grid, if there is no additional manifold specification, the mapping should be 1st order.
kx_mapping	Mapping used for K_y. Because a mesher usually generates 1st order grid, if there is no additional manifold specification, the mapping should be 1st order.

References LaplaceBEM::hierarchical_support_points_in_real_cell(), and LaplaceBEM::SauterQuadRule().

9.9.2.25 SauterQuadRule() [3/3]

```
template<int dim, int spacedim, typename RangeNumberType = double>
void LaplaceBEM::SauterQuadRule (
            FullMatrix< RangeNumberType > & system_matrix,
```

```
              const LaplaceKernel::KernelFunction< spacedim, RangeNumberType > & kernel_↩
function,
              const BEMValues< dim, spacedim > & bem_values,
              const typename DoFHandler< dim, spacedim >::cell_iterator & kx_cell_iter,
              const typename DoFHandler< dim, spacedim >::cell_iterator & ky_cell_iter,
              const MappingQGeneric< dim, spacedim > & kx_mapping = MappingQGeneric<dim, spacedim>(1),
              const MappingQGeneric< dim, spacedim > & ky_mapping = MappingQGeneric<dim, spacedim>(1)
)
```

This function implements Sauter's quadrature rule on quadrangular mesh. It handles various cases including same panel, common edge, common vertex and regular cell neighboring types.

Parameters

kernel_function	Laplace kernel function.
kx_cell_iter	Iterator pointing to K_x.
kx_cell_iter	Iterator pointing to K_y.
kx_mapping	Mapping used for K_x. Because a mesher usually generates 1st order grid, if there is no additional manifold specification, the mapping should be 1st order.
kx_mapping	Mapping used for K_y. Because a mesher usually generates 1st order grid, if there is no additional manifold specification, the mapping should be 1st order.

References LaplaceBEM::hierarchical_support_points_in_real_cell(), and LaplaceBEM::SauterQuadRule().

9.9.2.26 surface_jacobian_det() [1/2]

```
template<int dim, int spacedim>
double LaplaceBEM::surface_jacobian_det (
              const FiniteElement< dim, spacedim > & fe,
              const std::vector< Point< spacedim >> & support_points_in_real_cell,
              const Point< dim > & p )
```

Calculate the surface Jacobian determinant at specified area coordinates. Shape functions and support points in the real cell have been reordered to the tensor product ordering before the calculation.

References LaplaceBEM::surface_jacobian_det().

Referenced by LaplaceBEM::surface_jacobian_det().

9.9.2.27 surface_jacobian_det() [2/2]

```
template<int spacedim>
double LaplaceBEM::surface_jacobian_det (
              const unsigned int k3_index,
              const unsigned int quad_no,
              const Table< 2, FullMatrix< double >> & shape_grad_matrix_table,
              const std::vector< Point< spacedim >> & support_points_in_real_cell )
```

Calculate the surface Jacobian determinant at specified area coordinates. Shape functions and support points in the real cell have been reordered to the tensor product ordering before the calculation.

References LaplaceBEM::surface_jacobian_det().

9.9.2.28 transform_unit_to_permuted_real_cell() [1/2]

```
template<int dim, int spacedim>
Point<spacedim> LaplaceBEM::transform_unit_to_permuted_real_cell (
            const FiniteElement< dim, spacedim > & fe,
            const std::vector< Point< spacedim >> & support_points_in_real_cell,
            const Point< dim > & area_coords )
```

Calculate the coordinate transformation from a unit cell to real cell for Qp element.

Parameters

fe	
support_points_in_real_cell	coordinates of the support points in the real cell, which are in the tensor product order.
area_coords	

Returns

References LaplaceBEM::transform_unit_to_permuted_real_cell().

Referenced by LaplaceBEM::transform_unit_to_permuted_real_cell(), and LaplaceBEM::KernelPulledbackToUnit↩ Cell< dim, spacedim, RangeNumberType >::value().

9.9.2.29 transform_unit_to_permuted_real_cell() [2/2]

```
template<int spacedim>
Point<spacedim> LaplaceBEM::transform_unit_to_permuted_real_cell (
            const unsigned int k3_index,
            const unsigned int quad_no,
            const Table< 3, double > & shape_value_table,
            const std::vector< Point< spacedim >> & support_points_in_real_cell )
```

Calculate the coordinate transformation from a unit cell to real cell for Qp element.

Parameters

k3_index	
quad_no	
shape_value_table	
support_points_in_real_cell	coordinates of the support points in the real cell, which are in the tensor product order.

Returns

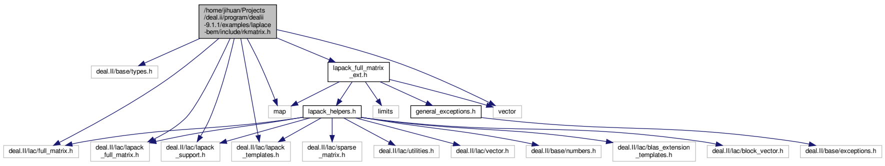
References LaplaceBEM::transform_unit_to_permuted_real_cell().

9.10 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/rkmatrix.h File Reference

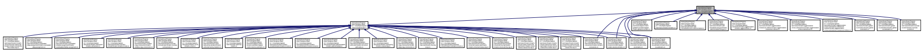
Definition of rank-k matrix.

```
#include <deal.II/base/types.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/lac/lapack_full_matrix.h>
#include <deal.II/lac/lapack_support.h>
#include <deal.II/lac/lapack_templates.h>
#include <map>
#include <vector>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for rkmatrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class HMatrix< spacedim, Number >
- class RkMatrix< Number >

Functions

- template<typename Number >
  void print_rkmatrix_to_mat (std::ostream &out, const std::string &name, const RkMatrix< Number > &values, const unsigned int precision=8, const bool scientific=true, const unsigned int width=0, const char ∗zero_↩ string="0", const double denominator=1., const double threshold=0.)

9.10.1 Detailed Description

Definition of rank-k matrix.

Date

2021-06-06

Author

Jihuan Tian

9.10.2 Function Documentation

9.10.2.1 print_rkmatrix_to_mat()

```
template<typename Number >
void print_rkmatrix_to_mat (
            std::ostream & out,
            const std::string & name,
            const RkMatrix< Number > & values,
            const unsigned int precision = 8,
            const bool scientific = true,
            const unsigned int width = 0,
            const char * zero_string = "0",
            const double denominator = 1.,
            const double threshold = 0. )
```

Print an RkMatrix in Octave text data format.

Parameters

out	
name	
values	
precision	
scientific	
width	
zero_string	
denominator	
threshold	

References RkMatrix< Number >::formal_rank, RkMatrix< Number >::m, RkMatrix< Number >::n, and Rk↩
Matrix< Number >::rank.

Referenced by main().

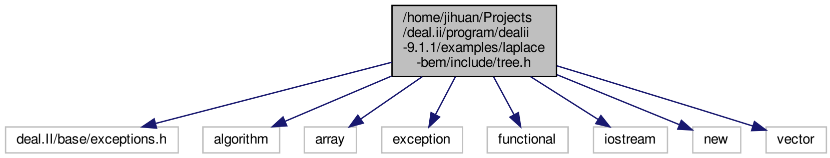
9.11 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/include/tree.h File Reference

Implementation of the classes for binary tree node, general tree node and functions for manipulating the trees constructed from these nodes.

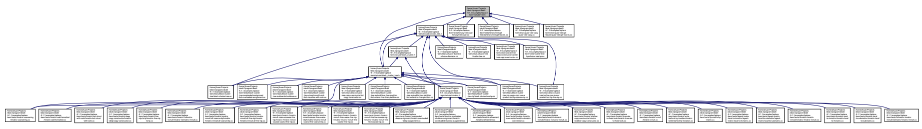
```
#include <deal.II/base/exceptions.h>
#include <algorithm>
#include <array>
#include <exception>
#include <functional>
#include <iostream>
#include <new>
```

```
#include <vector>
```

Include dependency graph for tree.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BinaryTreeNode< T >

    Class for binary tree node.
- class TreeNode< T, N >

    Class for general tree node.

- enum TreeNodeSplitMode { **HorizontalSplitMode**, **VerticalSplitMode**, **CrossSplitMode**, **UnsplitMode** }
- template<typename T >
  BinaryTreeNode< T > ∗ CreateTreeNode (const T &data, unsigned int level=0, BinaryTreeNode< T > ∗left=nullptr, BinaryTreeNode< T > ∗right=nullptr, BinaryTreeNode< T > ∗parent=nullptr)
- template<typename T , std::size_t N>
  TreeNode< T, N > ∗ CreateTreeNode (const T &data, unsigned int level, const std::array< TreeNode< T, N > ∗, N > &children, TreeNode< T, N > ∗parent=nullptr, TreeNodeSplitMode split_mode=UnsplitMode)
- template<typename T >
  void DeleteTreeNode (const BinaryTreeNode< T > ∗p)
- template<typename T , std::size_t N>
  void DeleteTreeNode (const TreeNode< T, N > ∗p)
- template<typename T >
  void PrintTreeNode (std::ostream &out, const BinaryTreeNode< T > ∗p)
- template<typename T , std::size_t N>
  void PrintTreeNode (std::ostream &out, const TreeNode< T, N > ∗p)
- template<typename T >
  unsigned int calc_depth (const BinaryTreeNode< T > ∗p)
- template<typename T , std::size_t N>
  unsigned int calc_depth (TreeNode< T, N > ∗p)
- template<typename T >
  void Preorder (BinaryTreeNode< T > ∗p, std::function< void(BinaryTreeNode< T > ∗)> operate)
- template<typename T >
  void Preorder (const BinaryTreeNode< T > ∗p, std::function< void(const BinaryTreeNode< T > ∗)> operate)
- template<typename T , std::size_t N>
  void Preorder (TreeNode< T, N > ∗p, std::function< void(TreeNode< T, N > ∗)> operate)

- template<typename T , std::size_t N>
  void Preorder (const TreeNode< T, N > *p, std::function< void(const TreeNode< T, N > *)> operate)
- template<typename T >
  void Inorder (BinaryTreeNode< T > *p, std::function< void(BinaryTreeNode< T > *)> operate)
- template<typename T >
  void Inorder (const BinaryTreeNode< T > *p, std::function< void(const BinaryTreeNode< T > *)> operate)
- template<typename T >
  void Postorder (BinaryTreeNode< T > *p, std::function< void(BinaryTreeNode< T > *)> operate)
- template<typename T >
  void Postorder (const BinaryTreeNode< T > *p, std::function< void(const BinaryTreeNode< T > *)> operate)
- template<typename T , std::size_t N>
  void Postorder (TreeNode< T, N > *p, std::function< void(TreeNode< T, N > *)> operate)
- template<typename T , std::size_t N>
  void Postorder (const TreeNode< T, N > *p, std::function< void(const TreeNode< T, N > *)> operate)
- template<typename T >
  BinaryTreeNode< T > * CopyTree (const BinaryTreeNode< T > *p)
- template<typename T , std::size_t N>
  TreeNode< T, N > * CopyTree (const TreeNode< T, N > *p)
- template<typename T >
  void GetTreeLeaves (const BinaryTreeNode< T > *p, std::vector< BinaryTreeNode< T > *> &leaf_set)
- template<typename T , std::size_t N>
  void GetTreeLeaves (const TreeNode< T, N > *p, std::vector< TreeNode< T, N > *> &leaf_set)
- template<typename T >
  void PrintTree (std::ostream &out, const BinaryTreeNode< T > *p)
- template<typename T , std::size_t N>
  void PrintTree (std::ostream &out, const TreeNode< T, N > *p)
- template<typename T >
  unsigned int CountTreeNodes (const BinaryTreeNode< T > *p)
- template<typename T , std::size_t N>
  unsigned int CountTreeNodes (const TreeNode< T, N > *p)
- template<typename T >
  void DeleteTree (const BinaryTreeNode< T > *p)
- template<typename T , std::size_t N>
  void DeleteTree (const TreeNode< T, N > *p)

9.11.1 Detailed Description

Implementation of the classes for binary tree node, general tree node and functions for manipulating the trees constructed from these nodes.

Date

2021-04-18

Author

Jihuan Tian

9.11.2 Enumeration Type Documentation

9.11.2.1 TreeNodeSplitMode

```
enum TreeNodeSplitMode
```

The splitting mode for a tree node, which can be cross split (`CrossSplitMode`), horizontal split (`Horizontal↩ SplitMode`), vertical split (`VerticalSplitMode`) and unsplit (`UnsplitMode`).

Note

1. When a tree node has four children, which are constructed via a tensor product of the children of the τ cluster and the children of the σ cluster, it must be cross split.
2. When a tree node has two children, which are constructed via a tensor product of the children of the τ cluster and the σ cluster, i.e. only the rows of the associated $\mathcal{H}$-matrix node will be split, then it should be horizontal split.
3. When a tree node has two children, which are constructed via a tensor product of the τ cluster and the children of the σ cluster, i.e. only the columns of the associated $\mathcal{H}$-matrix node will be split, then it should be vertical split.

9.11.3 Function Documentation

9.11.3.1 calc_depth() [1/2]

```
template<typename T >
unsigned int calc_depth (
            const BinaryTreeNode< T > * p )
```

Calculate the depth of a BinaryTree using recursion.

References BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >::Right().

Referenced by calc_depth(), BlockClusterTree< spacedim, Number >::calc_depth_and_max_level(), and Cluster↩ Tree< spacedim, Number >::partition().

9.11.3.2 calc_depth() [2/2]

```
template<typename T , std::size_t N>
unsigned int calc_depth (
            TreeNode< T, N > * p )
```

Calculate the depth of a TreeNode using recursion.

References calc_depth(), and TreeNode< T, N >::get_child_pointer().

9.11.3.3 CopyTree() [1/2]

```
template<typename T >
BinaryTreeNode<T>* CopyTree (
             const BinaryTreeNode< T > * p )
```

Copy the current node to a new node without setting children, parent and child_num at the moment.

Associate the newly created left child node with the current node.

Associate the newly created right child node with the current node.

References CreateTreeNode(), BinaryTreeNode< T >::get_data_reference(), BinaryTreeNode< T >::get_level(), BinaryTreeNode< T >::increase_child_num(), BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >::Right().

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree(), ClusterTree< spacedim, Number >↩ ::ClusterTree(), CopyTree(), and BlockClusterTree< spacedim, Number >::operator=().

9.11.3.4 CopyTree() [2/2]

```
template<typename T , std::size_t N>
TreeNode<T, N>* CopyTree (
             const TreeNode< T, N > * p )
```

Perform deep copy of a tree.

Parameters

p	

Returns

Create and copy the current node.

References CopyTree(), CreateTreeNode(), TreeNode< T, N >::get_child_num(), TreeNode< T, N >::get_child_↩ pointer(), TreeNode< T, N >::get_data_reference(), TreeNode< T, N >::get_level(), TreeNode< T, N >::get_split↩ _mode(), TreeNode< T, N >::increase_child_num(), TreeNode< T, N >::Parent(), and TreeNode< T, N >::set_↩ child_pointer().

9.11.3.5 CountTreeNodes() [1/2]

```
template<typename T >
unsigned int CountTreeNodes (
             const BinaryTreeNode< T > * p )
```

Count the total number of the binary tree nodes by recursion.

References BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >::Right().

Referenced by CountTreeNodes().

9.11.3.6 CountTreeNodes() [2/2]

```
template<typename T , std::size_t N>
unsigned int CountTreeNodes (
              const TreeNode< T, N > * p )
```

Count the total number of the tree nodes by recursion.

References CountTreeNodes(), and TreeNode< T, N >::get_child_pointer().

9.11.3.7 CreateTreeNode() [1/2]

```
template<typename T >
BinaryTreeNode<T>* CreateTreeNode (
              const T & data,
              unsigned int level = 0,
              BinaryTreeNode< T > * left = nullptr,
              BinaryTreeNode< T > * right = nullptr,
              BinaryTreeNode< T > * parent = nullptr )
```

Create a new binary tree node from the provided data.

Referenced by BlockClusterTree< spacedim, Number >::BlockClusterTree(), and CopyTree().

9.11.3.8 CreateTreeNode() [2/2]

```
template<typename T , std::size_t N>
TreeNode<T, N>* CreateTreeNode (
              const T & data,
              unsigned int level,
              const std::array< TreeNode< T, N > *, N > & children,
              TreeNode< T, N > * parent = nullptr,
              TreeNodeSplitMode split_mode = UnsplitMode )
```

Create a new tree node from the provided data.

9.11.3.9 DeleteTree() [1/2]

```
template<typename T >
void DeleteTree (
              const BinaryTreeNode< T > * p )
```

Use the Delete a whole binary tree using post-order traversal.

Referenced by BlockClusterTree< spacedim, Number >::release(), and ClusterTree< spacedim, Number >::∼↩
ClusterTree().

9.11.3.10 DeleteTree() [2/2]

```
template<typename T , std::size_t N>
void DeleteTree (
            const TreeNode< T, N > * p )
```

Use the Delete a whole tree using post-order traversal.

9.11.3.11 DeleteTreeNode() [1/2]

```
template<typename T >
void DeleteTreeNode (
            const BinaryTreeNode< T > * p )
```

Destroy a binary tree node.

9.11.3.12 DeleteTreeNode() [2/2]

```
template<typename T , std::size_t N>
void DeleteTreeNode (
            const TreeNode< T, N > * p )
```

Destroy a tree node.

9.11.3.13 GetTreeLeaves() [1/2]

```
template<typename T >
void GetTreeLeaves (
            const BinaryTreeNode< T > * p,
            std::vector< BinaryTreeNode< T > *> & leaf_set )
```

Construct the leaf set of a binary tree.

Note The leaf set should be cleared before calling this function.

Parameters

p	
leaf_set	

If the current node has no children, append it to the leaf set.

If the current node has children, recursively collect leaves from its left and right children.

References BinaryTreeNode< T >::get_child_num(), BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >↩ ::Right().

Referenced by ClusterTree< spacedim, Number >::build_leaf_set(), BlockClusterTree< spacedim, Number >↩ ::build_leaf_set(), and GetTreeLeaves().

9.11.3.14 GetTreeLeaves() [2/2]

```
template<typename T , std::size_t N>
void GetTreeLeaves (
            const TreeNode< T, N > * p,
            std::vector< TreeNode< T, N > *> & leaf_set )
```

Construct the leaf set of a general tree.

Note The leaf set should be cleared before calling this function.

Parameters

p	
leaf_set	

If the current node has no children, append it to the leaf set.

If the current node has children, recursively collect leaves from each of its children.

References TreeNode< T, N >::get_child_num(), TreeNode< T, N >::get_child_pointer(), and GetTreeLeaves().

9.11.3.15 Inorder() [1/2]

```
template<typename T >
void Inorder (
            BinaryTreeNode< T > * p,
            std::function< void(BinaryTreeNode< T > *)> operate )
```

In-order traverse of a binary tree.

References BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >::Right().

Referenced by Inorder().

9.11.3.16 Inorder() [2/2]

```
template<typename T >
void Inorder (
            const BinaryTreeNode< T > * p,
            std::function< void(const BinaryTreeNode< T > *)> operate )
```

In-order traverse of a binary tree (const version).

References Inorder(), BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >::Right().

9.11.3.17 Postorder() [1/4]

```
template<typename T >
void Postorder (
            BinaryTreeNode< T > * p,
            std::function< void(BinaryTreeNode< T > *)> operate )
```

Post-order traverse of a binary tree.

References BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >::Right().

Referenced by Postorder().

9.11.3.18 Postorder() [2/4]

```
template<typename T >
void Postorder (
            const BinaryTreeNode< T > * p,
            std::function< void(const BinaryTreeNode< T > *)> operate )
```

Post-order traverse of a binary tree (const version).

References BinaryTreeNode< T >::Left(), Postorder(), and BinaryTreeNode< T >::Right().

9.11.3.19 Postorder() [3/4]

```
template<typename T , std::size_t N>
void Postorder (
            TreeNode< T, N > * p,
            std::function< void(TreeNode< T, N > *)> operate )
```

Post-order traverse of a tree. Recursively call the operate function on each child.

References TreeNode< T, N >::get_child_pointer(), and Postorder().

9.11.3.20 Postorder() [4/4]

```
template<typename T , std::size_t N>
void Postorder (
            const TreeNode< T, N > * p,
            std::function< void(const TreeNode< T, N > *)> operate )
```

Post-order traverse of a tree (const version). Recursively call the operate function on each child.

References TreeNode< T, N >::get_child_pointer(), and Postorder().

9.11.3.21 Preorder() [1/4]

```
template<typename T >
void Preorder (
            BinaryTreeNode< T > * p,
            std::function< void(BinaryTreeNode< T > *)> operate )
```

Pre-order traverse of a binary tree.

References BinaryTreeNode< T >::Left(), and BinaryTreeNode< T >::Right().

Referenced by Preorder().

9.11.3.22 Preorder() [2/4]

```
template<typename T >
void Preorder (
            const BinaryTreeNode< T > * p,
            std::function< void(const BinaryTreeNode< T > *)> operate )
```

Pre-order traverse of a binary tree (const version).

References BinaryTreeNode< T >::Left(), Preorder(), and BinaryTreeNode< T >::Right().

9.11.3.23 Preorder() [3/4]

```
template<typename T , std::size_t N>
void Preorder (
            TreeNode< T, N > * p,
            std::function< void(TreeNode< T, N > *)> operate )
```

Pre-order traverse of a tree. Recursively call the function itself on each child.

References TreeNode< T, N >::get_child_pointer(), and Preorder().

9.11.3.24 Preorder() [4/4]

```
template<typename T , std::size_t N>
void Preorder (
            const TreeNode< T, N > * p,
            std::function< void(const TreeNode< T, N > *)> operate )
```

Pre-order traverse of a tree (const version). Recursively call the operate function on each child.

References TreeNode< T, N >::get_child_pointer(), and Preorder().

9.11.3.25 PrintTree() [1/2]

```
template<typename T >
void PrintTree (
            std::ostream & out,
            const BinaryTreeNode< T > * p )
```

Print a binary tree recursively by starting from a node.

9.11.3.26 PrintTree() [2/2]

```
template<typename T , std::size_t N>
void PrintTree (
            std::ostream & out,
            const TreeNode< T, N > * p )
```

Print a tree recursively by starting from a node.

9.11.3.27 PrintTreeNode() [1/2]

```
template<typename T >
void PrintTreeNode (
            std::ostream & out,
            const BinaryTreeNode< T > * p )
```

Print the data contained in a binary tree node.

References BinaryTreeNode< T >::get_child_num(), BinaryTreeNode< T >::get_data_reference(), BinaryTree↩ Node< T >::get_level(), and BinaryTreeNode< T >::Parent().

9.11.3.28 PrintTreeNode() [2/2]

```
template<typename T , std::size_t N>
void PrintTreeNode (
            std::ostream & out,
            const TreeNode< T, N > * p )
```

Print the data contained in a tree node.

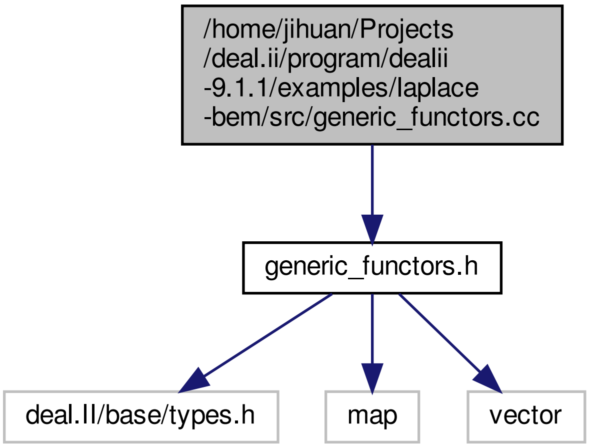
References TreeNode< T, N >::get_child_num(), TreeNode< T, N >::get_data_reference(), TreeNode< T, N >↩ ::get_level(), TreeNode< T, N >::get_split_mode(), and TreeNode< T, N >::Parent().

9.12 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/src/generic_functors.cc File Reference

Introduction of generic_functors.cc.

```
#include "generic_functors.h"
```

Include dependency graph for generic_functors.cc:



Functions

- void build_index_set_global_to_local_map (const std::vector< dealii::types::global_dof_index > &index_set_as_local_to_global_map, std::map< dealii::types::global_dof_index, size_t > &global_to_local_map)

9.12.1 Detailed Description

Introduction of generic_functors.cc.

Date

2021-07-31

Author

Jihuan Tian

9.12.2 Function Documentation

9.12.2.1 build_index_set_global_to_local_map()

```
void build_index_set_global_to_local_map (
            const std::vector< dealii::types::global_dof_index > & index_set_as_local_to_↩
global_map,
            std::map< dealii::types::global_dof_index, size_t > & global_to_local_map )
```

Build a map from global indices to local indices based on the given index set, which is actually a map from local indices to global indices.

Parameters

index_set_as_local_to_global_map	
global_to_local_map	

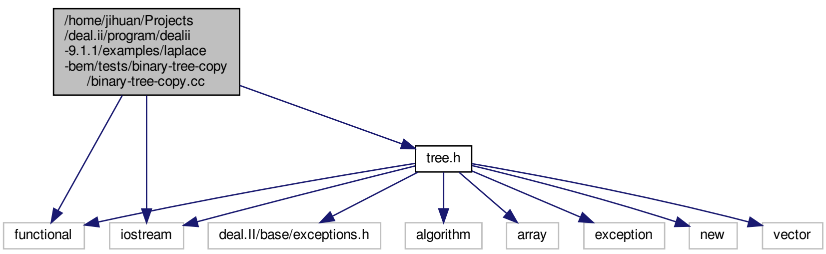
Referenced by convertHMatBlockToRkMatrix(), f_h_mmult(), find_pointer_data(), h_f_mmult(), h_rk_mmult(), Init↩ AndCreateHMatrixChildren(), main(), and rk_h_mmult().

9.13 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/binary-tree-copy/binary-tree-copy.cc File Reference

Verify the copy constructor of a binary tree.

```
#include <functional>
#include <iostream>
#include "tree.h"
```

Include dependency graph for binary-tree-copy.cc:



Functions

- BinaryTreeNode< int > ∗ MakeIntExampleTree ()
- int **main** ()

9.13.1 Detailed Description

Verify the copy constructor of a binary tree.

Author

Jihuan Tian

Date

2021-07-20

9.13.2 Function Documentation

9.13.2.1 MakeIntExampleTree()

```
BinaryTreeNode<int>* MakeIntExampleTree ( )
```

Create an example tree containing integers as below. The tree will be constructed in a bottom-up approach.

```
     10
    /  \
  3      5
 / \    / \
2   4  7   9
   / \
  8   6
```

The function returns the pointer to the root node of the tree.

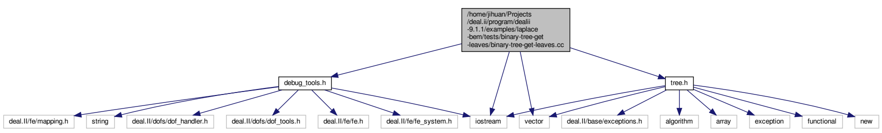
References BinaryTreeNode< T >::Parent().

9.14 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/binary-tree-get-leaves/binary-tree-get-leaves.cc File Reference

Verify extraction of the leaves of a binary tree.

```
#include <iostream>
#include <vector>
#include "debug_tools.h"
#include "tree.h"
```

Include dependency graph for binary-tree-get-leaves.cc:



Functions

- BinaryTreeNode< int > * MakeIntExampleTree ()
- int **main** ()

9.14.1 Detailed Description

Verify extraction of the leaves of a binary tree.

Author

Jihuan Tian

Date

2021-07-21

9.14.2 Function Documentation

9.14.2.1 MakeIntExampleTree()

```
BinaryTreeNode<int>* MakeIntExampleTree ( )
```

Create an example tree containing integers as below. The tree will be constructed in a bottom-up approach.

```
     10
    /  \
  3      5
 / \    / \
2   4  7   9
   / \
  8   6
```

The function returns the pointer to the root node of the tree.

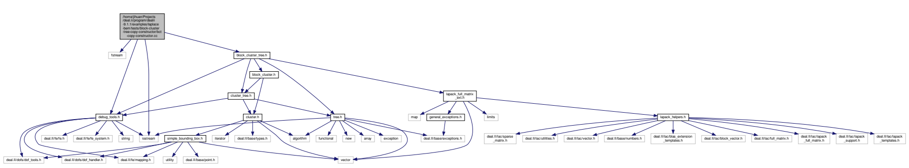
References BinaryTreeNode< T >::Parent().

9.15 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-copy-constructor/bct-copy-constructor.cc File Reference

Verify the copy constructor of block cluster tree.

```
#include <fstream>
#include <iostream>
#include "block_cluster_tree.h"
#include "debug_tools.h"
```

Include dependency graph for bct-copy-constructor.cc:



Functions

- int main ()

9.15.1 Detailed Description

Verify the copy constructor of block cluster tree.

Author

Jihuan Tian

Date

2021-07-21

9.15.2 Function Documentation

9.15.2.1 main()

```
int main ( )
```

Set the dimension of the H^p-matrix to be built.

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

Generate the block cluster tree.

Deep copy

Shallow copy

After shallow copy, try to print BCT1.

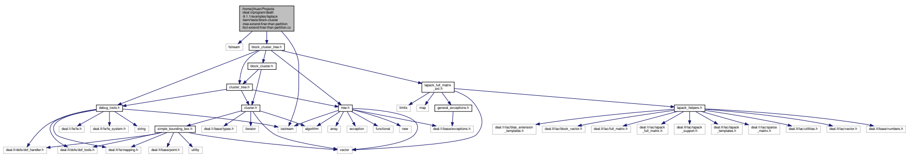
References ClusterTree< spacedim, Number >::partition(), and BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product().

9.16 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-extend-finer-than-partition/bct-extend-finer-than-partition.cc File Reference

Verify extend a block cluster tree to be finer than a given partition.

```
#include <fstream>
#include <iostream>
#include "block_cluster_tree.h"
```

Include dependency graph for bct-extend-finer-than-partition.cc:



Functions

- int main ()

9.16.1 Detailed Description

Verify extend a block cluster tree to be finer than a given partition.

Author

Jihuan Tian

Date

2021-07-22

9.16.2 Function Documentation

9.16.2.1 main()

```
int main ( )
```

Set the dimension of the H^p-matrix to be built.

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

Create two block cluster trees. One has the fine non-tensor product, and the other has the coarse non-tensor product partition.

Make a copy of the first tree then extend it.

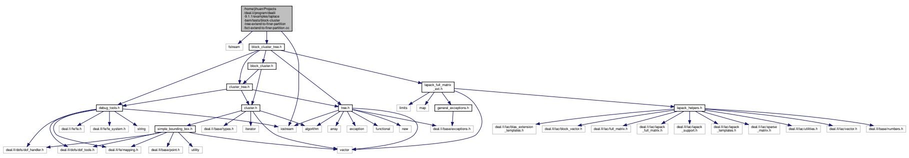
References BlockClusterTree< spacedim, Number >::extend_finer_than_partition(), BlockClusterTree< spacedim, Number >::get_leaf_set(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and BlockClusterTree< spacedim, Number >::write_leaf_set().

9.17 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-extend-to-finer-partition/bct-extend-to-finer-partition.cc File Reference

Verify extend a block cluster tree to a given finer partition.

```
#include <fstream>
#include <iostream>
#include "block_cluster_tree.h"
```

Include dependency graph for bct-extend-to-finer-partition.cc:



Functions

- int main ()

9.17.1 Detailed Description

Verify extend a block cluster tree to a given finer partition.

Author

Jihuan Tian

Date

2021-07-23

9.17.2 Function Documentation

9.17.2.1 main()

```
int main ( )
```

Set the dimension of the H^p-matrix to be built.

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

Create two block cluster trees. One has the fine non-tensor product partition and the other has the coarse non-tensor product partition.

Note The minimum cluster size `n_min` associated the cluster trees used for building the block cluster tree might be different from that of the block cluster tree. Usually, `n_min` for the cluster tree is set to 1, which allows block cluster trees of different depths can be built from it.

Make a copy of the first tree and then extend it be finer than the second tree.

Make copy of the second tree and then extend it to the finer partition which is obtained from above by extending the first tree.

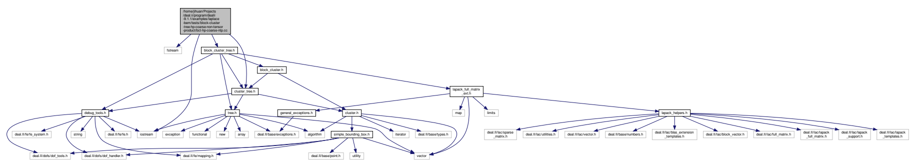
References BlockClusterTree< spacedim, Number >::extend_finer_than_partition(), BlockClusterTree< spacedim, Number >::extend_to_finer_partition(), BlockClusterTree< spacedim, Number >::get_leaf_set(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_↩ product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and BlockClusterTree< spacedim, Number >::write_leaf_set().

9.18 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-hp-coarse-non-tensor-product/bct-hp-coarse-ntp.cc File Reference

Test the construction of a $\mathcal{H}^p$ block cluster tree using the coarse non-tensor product partition.

```
#include <fstream>
#include <iostream>
#include "block_cluster_tree.h"
#include "cluster_tree.h"
```

Include dependency graph for bct-hp-coarse-ntp.cc:



Functions

- int main ()

9.18.1 Detailed Description

Test the construction of a $\mathcal{H}^p$ block cluster tree using the coarse non-tensor product partition.

Author

Jihuan Tian

Date

2021-06-22

9.18.2 Function Documentation

9.18.2.1 main()

```
int main ( )
```

Print the whole block cluster tree leaf set.

Generate a large structure.

Print the whole block cluster tree leaf set.

Print the whole block cluster tree leaf set.

Print the whole block cluster tree leaf set.

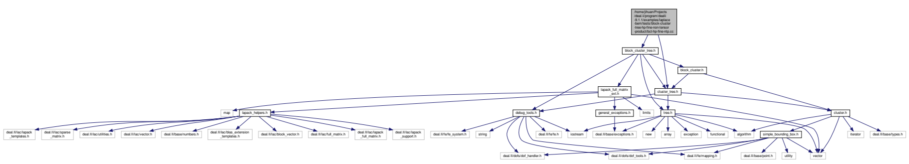
Print the whole block cluster tree leaf set.

References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_↩ coarse_non_tensor_product(), and BlockClusterTree< spacedim, Number >::write_leaf_set().

9.19 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-hp-fine-non-tensor-product/bct-hp-fine-ntp.cc File Reference

```
#include "block_cluster_tree.h"
#include "cluster_tree.h"
```

Include dependency graph for bct-hp-fine-ntp.cc:



Functions

- int main ()

9.19.1 Detailed Description

Author

Jihuan Tian

Date

2021-06-23

9.19.2 Function Documentation

9.19.2.1 main()

```
int main ( )
```

Print the whole block cluster tree leaf set.

Generate a large structure.

Print the whole block cluster tree leaf set.

Print the whole block cluster tree leaf set.

Print the whole block cluster tree leaf set.

Print the whole block cluster tree leaf set.

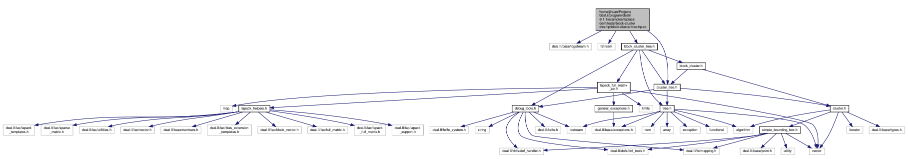
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_↩
fine_non_tensor_product(), and BlockClusterTree< spacedim, Number >::write_leaf_set().

9.20 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-hp/block-cluster-tree-hp.cc File Reference

Test the construction of a $\mathcal{H}^p$ block cluster tree using the tensor product partition.

```
#include <deal.II/base/logstream.h>
#include <fstream>
#include "block_cluster_tree.h"
#include "cluster_tree.h"
```

Include dependency graph for block-cluster-tree-hp.cc:



Functions

- int main ()

9.20.1 Detailed Description

Test the construction of a $\mathcal{H}^p$ block cluster tree using the tensor product partition.

Author

Jihuan Tian

Date

2021-06-22

9.20.2 Function Documentation

9.20.2.1 main()

```
int main ( )
```

Initialize deal.ii log stream.

Print the whole block cluster tree.

Print the tree structure.

Print the whole block cluster tree.

Print the tree structure.

Print the whole block cluster tree.

Print the tree structure.

Print the whole block cluster tree.

Print the tree structure.

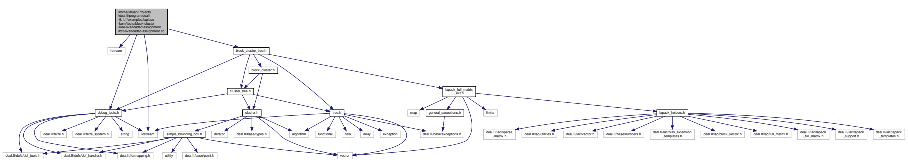
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_↩ tensor_product(), and BlockClusterTree< spacedim, Number >::write_leaf_set().

9.21 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-overloaded-assignment/bct-overloaded-assignment.cc File Reference

Verify the deep and shallow overloaded assignment operators for `BlockClusterTree`.

```
#include <fstream>
#include <iostream>
#include "block_cluster_tree.h"
#include "debug_tools.h"
```

Include dependency graph for bct-overloaded-assignment.cc:



Functions

- int main ()

9.21.1 Detailed Description

Verify the deep and shallow overloaded assignment operators for `BlockClusterTree`.

Author

Jihuan Tian

Date

2021-08-23

9.21.2 Function Documentation

9.21.2.1 main()

```
int main ( )
```

Set the dimension of the H^p-matrix to be built.

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

Generate the block cluster tree.

Deep assignment

Shallow assignment

After shallow assignment, try to print BCT1.

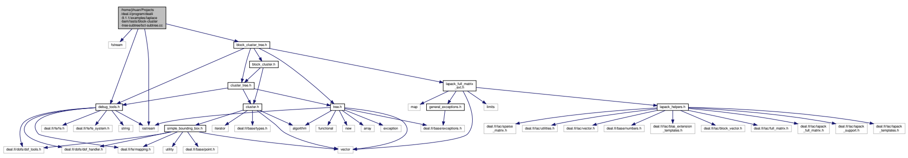
References ClusterTree< spacedim, Number >::partition(), and BlockClusterTree< spacedim, Number >↩
::partition_fine_non_tensor_product().

9.22 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-subtree/bct-subtree.cc File Reference

Verify the construction of a block cluster subtree and check the equality of its nodes with the original block cluster tree.

```
#include <fstream>
#include <iostream>
#include "block_cluster_tree.h"
#include "debug_tools.h"
```

Include dependency graph for bct-subtree.cc:



Functions

- int main ()

9.22.1 Detailed Description

Verify the construction of a block cluster subtree and check the equality of its nodes with the original block cluster tree.

Author

Jihuan Tian

Date

2021-07-19

9.22.2 Function Documentation

9.22.2.1 main()

```
int main ( )
```

Set the dimension of the H^p-matrix to be built.

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

Generate the block cluster tree.

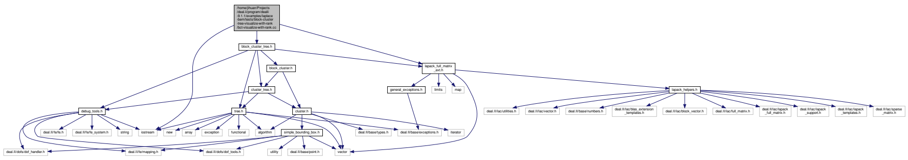
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and BlockClusterTree< spacedim, Number >::write_leaf_set().

9.23 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree-visualize-with-rank/bct-visualize-with-rank.cc File Reference

Visualize the block cluster tree structure with matrix block's rank calculated.

```
#include <iostream>
#include "block_cluster_tree.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for bct-visualize-with-rank.cc:



Functions

- int main ()

9.23.1 Detailed Description

Visualize the block cluster tree structure with matrix block's rank calculated.

Author

Jihuan Tian

Date

2021-07-04

9.23.2 Function Documentation

9.23.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

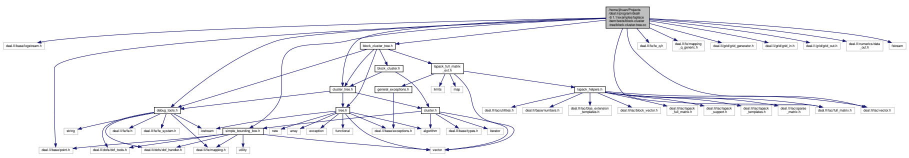
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and BlockClusterTree< spacedim, Number >::write_leaf_set().

9.24 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/block-cluster-tree/block-cluster-tree.cc File Reference

This files verifies the admissible block cluster partition.

```
#include <deal.II/base/logstream.h>
#include <deal.II/base/point.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/fe/mapping_q_generic.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/grid_in.h>
#include <deal.II/grid/grid_out.h>
#include <deal.II/lac/full_matrix.h>
#include <deal.II/lac/vector.h>
#include <deal.II/numerics/data_out.h>
#include <simple_bounding_box.h>
#include <fstream>
#include "block_cluster_tree.h"
#include "cluster_tree.h"
#include "debug_tools.h"
```

Include dependency graph for block-cluster-tree.cc:



Functions

- int main ()

9.24.1 Detailed Description

This files verifies the admissible block cluster partition.

Date

2021-04-28

Author

Jihuan Tian

9.24.2 Function Documentation

9.24.2.1 main()

```
int main ( )
```

Initialize deal.ii log stream.

Generate the 3x3 grid in a 2D square.

Save the mesh to a file for visualization.

Read the mesh from a file.

Create the Lagrangian finite element.

Create a DoFHandler, which is associated with the triangulation and distributed with the finite element.

Create the mapping object, which is required in generating the map from DoF indices to support points.

Generate a list of all DoF indices.

Get the spatial coordinates of the support points associated with DoF indices.

Calculate the average mesh cell size at each support point.

Initialize the cluster tree $T(I)$ and $T(J)$ for all the DoF indices.

Partition the cluster tree.

Print the cluster tree.

Create the block cluster tree.

Perform admissible partition on the block cluster tree.

Print the block cluster tree, even though there is only a root node.

Create a deal.ii vector storing the block cluster levels for all DoFs.

```
Iterate through the \form#0 component of each block cluster and get
```

the level of each DoF in τ. Then assign the level value to all DoFs contained in the cluster τ.

Cluster indices in a cluster tree: on level 2.

Save the block cluster data matrices.

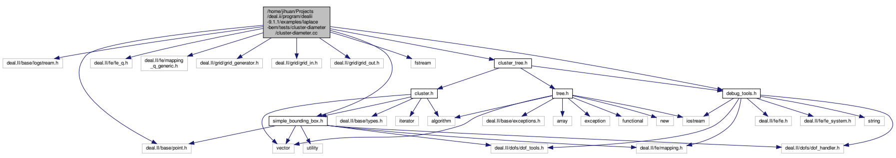
References ClusterTree< spacedim, Number >::get_leaf_set(), map_dofs_to_average_cell_size(), ClusterTree< spacedim, Number >::partition(), and BlockClusterTree< spacedim, Number >::partition().

9.25 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/cluster-diameter/cluster-diameter.cc File Reference

This files verifies the cluster diameter and pair-wise distance calculation using a 3x3 grid in a square.

```
#include <deal.II/base/logstream.h>
#include <deal.II/base/point.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/fe/mapping_q_generic.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/grid_in.h>
#include <deal.II/grid/grid_out.h>
#include <simple_bounding_box.h>
#include <fstream>
#include "cluster_tree.h"
#include "debug_tools.h"
```

Include dependency graph for cluster-diameter.cc:



Functions

- int main ()

9.25.1 Detailed Description

This files verifies the cluster diameter and pair-wise distance calculation using a 3x3 grid in a square.

Date

2021-04-25

Author

Jihuan Tian

9.25.2 Function Documentation

9.25.2.1 main()

```
int main ( )
```

Initialize deal.ii log stream.

Generate the 3x3 grid in a 2D square.

Save the mesh to a file for visualization.

Create the Lagrangian finite element.

Create a DoFHandler, which is associated with the triangulation and distributed with the finite element.

Create the mapping object, which is required in generating the map from DoF indices to support points.

Generate a list of all DoF indices.

Print the coordinates of all support points.

Write DoF indices at each support point.

Calculate the average mesh cell size at each support point.

Initialize the cluster tree $T(I)$ and $T(J)$ for all the DoF indices.

Partition the cluster tree.

Print the whole cluster trees.

Get the cluster containing DoF indices [0 1 2 3].

Get the cluster containing DoF indices [10, 11, 14, 15].

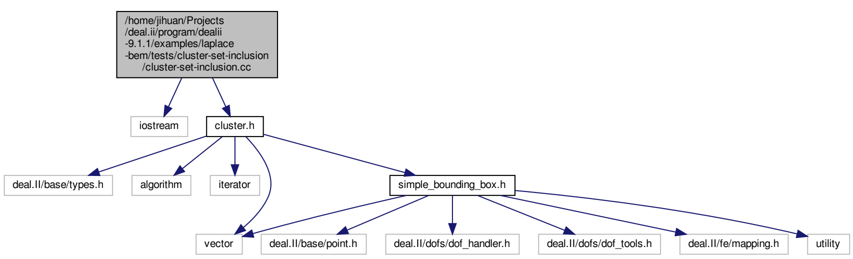
References map_dofs_to_average_cell_size(), map_dofs_to_max_cell_size(), map_dofs_to_min_cell_size(), and print_support_point_info().

9.26 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/cluster-set-inclusion/cluster-set-inclusion.cc File Reference

Verify set inclusion operation for cluster index sets.

```
#include <iostream>
#include "cluster.h"
```

Include dependency graph for cluster-set-inclusion.cc:



Functions

- int **main** ()

9.26.1 Detailed Description

Verify set inclusion operation for cluster index sets.

Author

Jihuan Tian

Date

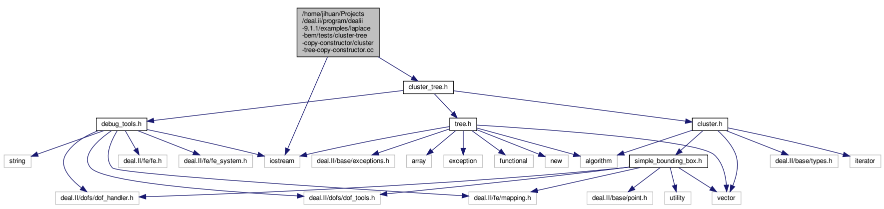
2021-07-21

9.27 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/cluster-tree-copy-constructor/cluster-tree-copy-constructor.cc File Reference

Verify the copy constructor of cluster tree.

```
#include <iostream>
#include "cluster_tree.h"
```

Include dependency graph for cluster-tree-copy-constructor.cc:



Functions

- int main ()

9.27.1 Detailed Description

Verify the copy constructor of cluster tree.

Author

Jihuan Tian

Date

2021-07-21

9.27.2 Function Documentation

9.27.2.1 main()

```
int main ( )
```

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

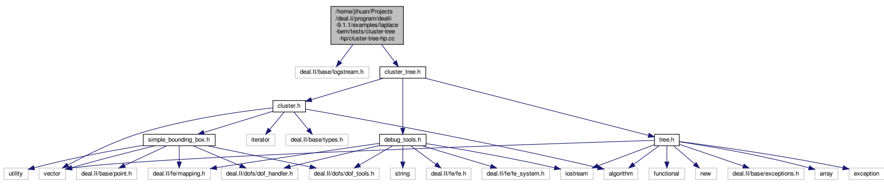
References ClusterTree$<$ spacedim, Number $>$::partition().

9.28 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/cluster-tree-hp/cluster-tree-hp.cc File Reference

Test the construction of a $\mathcal{H}^p$ cluster tree.

```
#include <deal.II/base/logstream.h>
#include "cluster_tree.h"
```

Include dependency graph for cluster-tree-hp.cc:



Functions

- int main ()

9.28.1 Detailed Description

Test the construction of a $\mathcal{H}^p$ cluster tree.

Author

Jihuan Tian

Date

2021-06-21

9.28.2 Function Documentation

9.28.2.1 main()

```
int main ( )
```

Initialize deal.ii log stream.

Print the whole cluster tree.

Print the whole cluster tree.

Print the whole cluster tree.

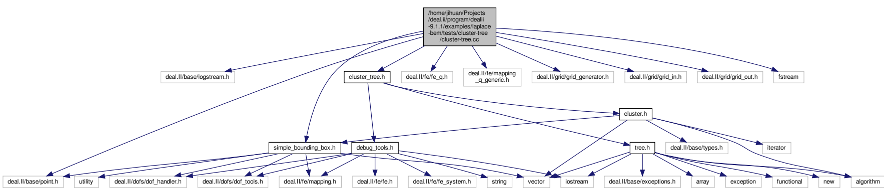
Print the whole cluster tree.

References ClusterTree< spacedim, Number >::partition().

9.29 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/cluster-tree/cluster-tree.cc File Reference

```
#include <deal.II/base/logstream.h>
#include <deal.II/base/point.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/fe/mapping_q_generic.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/grid_in.h>
#include <deal.II/grid/grid_out.h>
#include <simple_bounding_box.h>
#include <fstream>
#include "cluster_tree.h"
```

Include dependency graph for cluster-tree.cc:



Functions

- int main ()

9.29.1 Detailed Description

This file verifies the ClusterTree class.

9.29.2 Function Documentation

9.29.2.1 main()

```
int main ( )
```

Initialize deal.ii log stream.

Generate the grid for a 3D sphere.

N.B. Use type cast for triangulation to suppress Eclipse editor error prompt.

Save the mesh to a file for visualization.

Create a Lagrangian finite element.

Create a DoFHandler, which is associated with the triangulation and distributed with the finite element.

Create a mapping object, which is required in generating the map from DoF indices to support points.

Generate a list of all DoF indices.

Get the spatial coordinates of the support points associated with DoF indices.

Initialize a cluster tree for all the DoF indices.

Partition the cluster tree.

Print the coordinates of all support points.

Print the whole cluster tree.

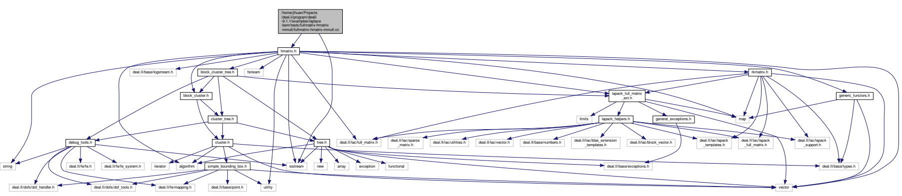
References ClusterTree< spacedim, Number >::partition().

9.30 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/fullmatrix-hmatrix-mmult/fullmatrix-hmatrix-mmult.cc File Reference

Verify the full matrix/H-matrix multiplication.

```
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for fullmatrix-hmatrix-mmult.cc:



Functions

- int main ()

9.30.1 Detailed Description

Verify the full matrix/H-matrix multiplication.

Author

Jihuan Tian

Date

2021-08-14

9.30.2 Function Documentation

9.30.2.1 main()

```
int main ( )
```

Create the first full matrix.

Create the second full matrix for initializing the second H-matrix.

Generate the DoF index set.

Construct the cluster tree.

Construct the block cluster tree using fine non-tensor product partition.

Create an $\mathcal{H}$-matrix based on the block cluster tree.

Perform matrix-matrix multiplication and the result is a full matrix.

Perform matrix-matrix multiplication and the result is a rank-k matrix.

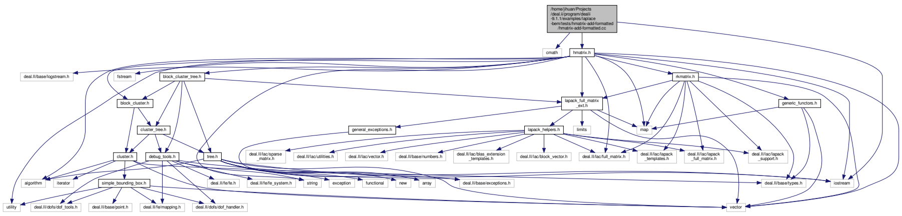
References f_h_mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), RkMatrix< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::print_formatted_to_mat().

9.31 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-add-formatted/hmatrix-add-formatted.cc File Reference

Verify formatted addition of two h-matrices.

```
#include <cmath>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-add-formatted.cc:



Functions

- int main ()

9.31.1 Detailed Description

Verify formatted addition of two h-matrices.

Author

Jihuan Tian

Date

2021-07-03

9.31.2 Function Documentation

9.31.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create two full matrices as the source data.

Create the two H-matrices `H1` and `H2` from `M1` and `M2`.

Get the full matrix representation of `H1` and `H2`.

Add the full matrix versions of `H1` and `H2`.

Add the two H-matrices `H1` and `H2`.

Convert the result matrix into a full matrix.

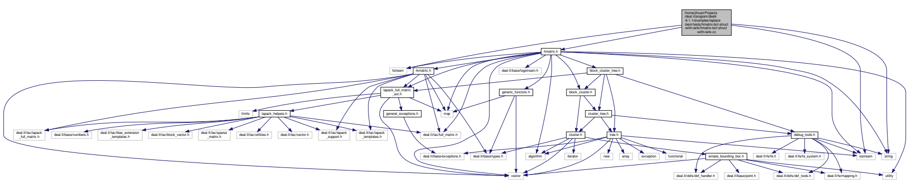
References HMatrix< spacedim, Number >::add(), LAPACKFullMatrixExt< Number >::add(), HMatrix< spacedim, Number >::convertToFullMatrix(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and LAPACKFullMatrixExt< Number >::print_formatted_to_↩ mat().

9.32 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-bct-struct-with-rank/hmatrix-bct-struct-with-rank.cc File Reference

Visualize the block cluster tree structure of an H-matrix with displayed rank.

```
#include <fstream>
#include <iostream>
#include <string>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-bct-struct-with-rank.cc:



Functions

- int main ()

9.32.1 Detailed Description

Visualize the block cluster tree structure of an H-matrix with displayed rank.

Author

Jihuan Tian

Date

2021-07-04

9.32.2 Function Documentation

9.32.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create the full matrix.

Convert the full matrix to H-matrix with specified rank. Then we convert it back to a full matrix for comparison with the original matrix.

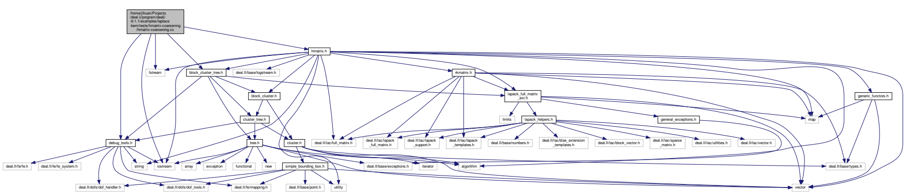
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_↩
fine_non_tensor_product(), and HMatrix< spacedim, Number >::write_leaf_set().

9.33 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-coarsening/hmatrix-coarsening.cc File Reference

Coarsen a H-matrix to its subtree.

```
#include <fstream>
#include <iostream>
#include "block_cluster_tree.h"
#include "debug_tools.h"
#include "hmatrix.h"
```

Include dependency graph for hmatrix-coarsening.cc:



Functions

- int main ()

9.33.1 Detailed Description

Coarsen a H-matrix to its subtree.

Author

Jihuan Tian

Date

2021-07-19

9.33.2 Function Documentation

9.33.2.1 main()

```
int main ( )
```

Set the dimension of the H^p-matrix to be built.

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

Generate two block cluster trees. One is coarser than the other.

Verify the block cluster partition structures for the two trees by printing out their leaf sets.

Create a full matrix as the data source.

Create an $\mathcal{H}$-matrix from the full matrix based on the fine block cluster tree.

Convert the H-matrix back to full matrix for verification in postprocessing.

Coarsen the H-matrix to the subtree.

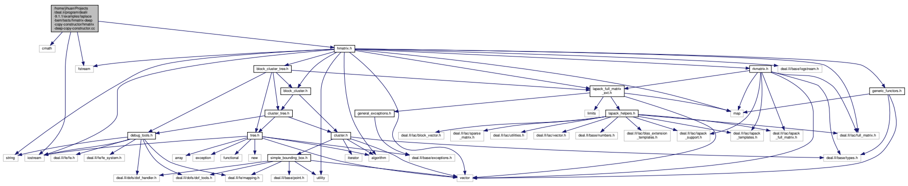
References HMatrix< spacedim, Number >::coarsen_to_subtree(), HMatrix< spacedim, Number >::convertToFullMatrix(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set().

9.34 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-deep-copy-constructor/hmatrix-deep-copy-constructor.cc File Reference

Verify the deep copy constructor of $\mathcal{H}$-matrix.

```
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-deep-copy-constructor.cc:



Functions

- int main ()

9.34.1 Detailed Description

Verify the deep copy constructor of $\mathcal{H}$-matrix.

Author

Jihuan Tian

Date

2021-08-24

9.34.2 Function Documentation

9.34.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create a full matrix as the source data.

Create an H-matrix from the block cluster tree.

Create an H-matrix using deep copy.

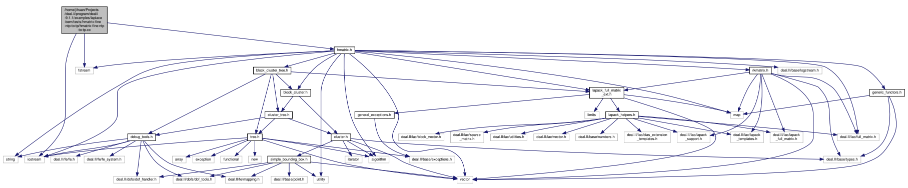
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition↩
_fine_non_tensor_product(), HMatrix< spacedim, Number >::print_formatted(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.35 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-fine-ntp-to-tp/hmatrix-fine-ntp-to-tp.cc File Reference

Verify the conversion of an $\mathcal{H}$-matrix from fine non-tensor product structure to tensor product structure.

```
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-fine-ntp-to-tp.cc:



Functions

- int main ()

9.35.1 Detailed Description

Verify the conversion of an $\mathcal{H}$-matrix from fine non-tensor product structure to tensor product structure.

Author

Jihuan Tian

Date

2021-07-30

9.35.2 Function Documentation

9.35.2.1 main()

```
int main ( )
```

Generate the DoF index set.

Construct the cluster tree.

Construct the block cluster tree using fine non-tensor product partition.

Construct the block cluster tree using tensor product partition.

Create a full matrix for initializing $\mathcal{H}$-matrices.

Create an $\mathcal{H}$-matrix based on the first block cluster tree.

Convert $\mathcal{H}$-matrix back to full matrix for verification.

Convert the $\mathcal{H}$-matrix to the second block cluster tree.

Convert $\mathcal{H}$-matrix back to full matrix for verification.

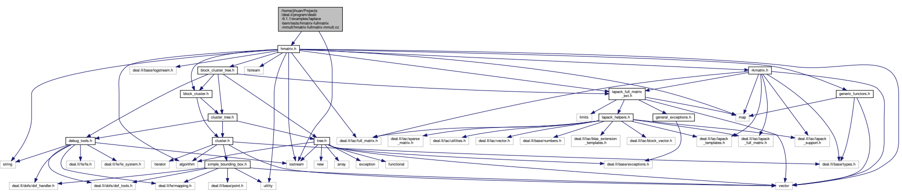
References HMatrix< spacedim, Number >::convert_between_different_block_cluster_trees(), HMatrix< spacedim, Number >::convertToFullMatrix(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition↩_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.36 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-fullmatrix-mmult/hmatrix-fullmatrix-mmult.cc File Reference

Verify the H-matrix/full matrix multiplication.

```
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-fullmatrix-mmult.cc:



Functions

- int main ()

9.36.1 Detailed Description

Verify the H-matrix/full matrix multiplication.

Author

Jihuan Tian

Date

2021-08-14

9.36.2 Function Documentation

9.36.2.1 main()

```
int main ( )
```

Generate the DoF index set.

Construct the cluster tree.

Construct the block cluster tree using fine non-tensor product partition.

Create a full matrix for initializing the $\mathcal{H}$-matrix.

Create an $\mathcal{H}$-matrix based on the block cluster tree.

Create the second full matrix.

Perform matrix-matrix multiplication and the result is a full matrix.

Perform matrix-matrix multiplication and the result is a rank-k matrix.

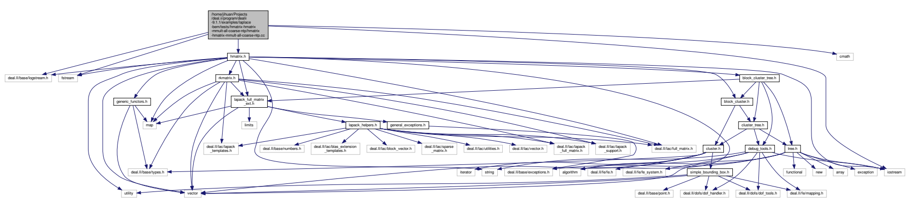
References h_f_mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), RkMatrix< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::print_formatted_to_mat().

9.37 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-hmatrix-mmult-all-coarse-ntp/hmatrix-hmatrix-mmult-all-coarse-ntp.cc File Reference

Verify the multiplication of two $\mathcal{H}$-matrices. Both operands and the result matrices have the coarse non-tensor product partitions.

```
#include <deal.II/base/logstream.h>
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-hmatrix-mmult-all-coarse-ntp.cc:



Functions

- int main ()

9.37.1 Detailed Description

Verify the multiplication of two $\mathcal{H}$-matrices. Both operands and the result matrices have the coarse non-tensor product partitions.

Author

Jihuan Tian

Date

2021-08-26

9.37.2 Function Documentation

9.37.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create two full matrices as the source data.

Create the two H-matrices `H1` and `H2` from `M1` and `M2`.

Get the full matrix representations of `H1` and `H2` as well as their product.

Multiply the two H-matrices `H1` and `H2`.

Convert the result matrix into a full matrix for verification.

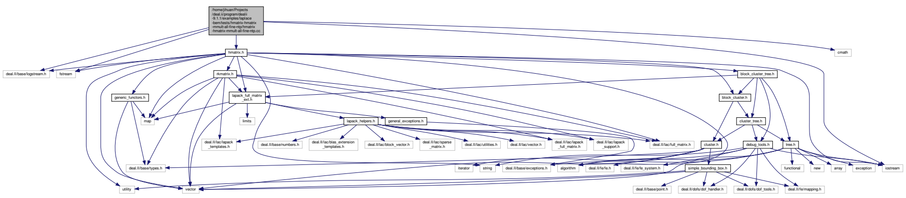
References HMatrix< spacedim, Number >::convertToFullMatrix(), HMatrix< spacedim, Number >::mmult(), LAPACKFullMatrixExt< Number >::mmult(), ClusterTree< spacedim, Number >::partition(), BlockCluster↩Tree< spacedim, Number >::partition_coarse_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_↩formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.38 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-hmatrix-mmult-all-fine-ntp/hmatrix-hmatrix-mmult-all-fine-ntp.cc File Reference

Verify the multiplication of two $\mathcal{H}$-matrices. Both operands and the result matrices have the fine non-tensor product partitions.

```
#include <deal.II/base/logstream.h>
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-hmatrix-mmult-all-fine-ntp.cc:



Functions

- int main ()

9.38.1 Detailed Description

Verify the multiplication of two $\mathcal{H}$-matrices. Both operands and the result matrices have the fine non-tensor product partitions.

Author

Jihuan Tian

Date

2021-08-19

9.38.2 Function Documentation

9.38.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create two full matrices as the source data.

Create the two H-matrices `H1` and `H2` from `M1` and `M2`.

Get the full matrix representations of `H1` and `H2` as well as their product.

Multiply the two H-matrices `H1` and `H2`.

Convert the result matrix into a full matrix for verification.

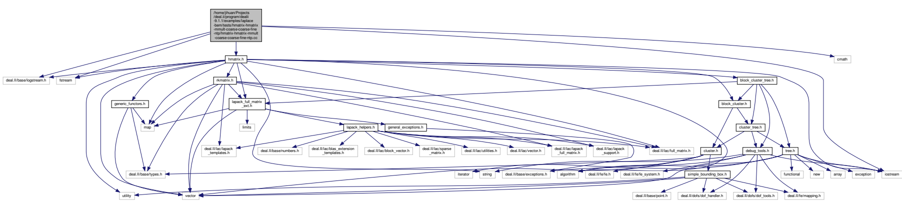
References HMatrix< spacedim, Number >::convertToFullMatrix(), HMatrix< spacedim, Number >::mmult(), LAPACKFullMatrixExt< Number >::mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.39 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-hmatrix-mmult-coarse-coarse-fine-ntp/hmatrix-hmatrix-mmult-coarse-coarse-fine-ntp.cc File Reference

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have coarse-coarse-fine non-tensor product partitions.

```
#include <deal.II/base/logstream.h>
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-hmatrix-mmult-coarse-coarse-fine-ntp.cc:



Functions

- int main ()

9.39.1 Detailed Description

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have coarse-coarse-fine non-tensor product partitions.

Author

Jihuan Tian

Date

2021-08-26

9.39.2 Function Documentation

9.39.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create two full matrices as the source data.

Create the two H-matrices `H1` and `H2` from `M1` and `M2`.

Get the full matrix representations of `H1` and `H2` as well as their product.

Multiply the two H-matrices `H1` and `H2`.

Convert the result matrix into a full matrix for verification.

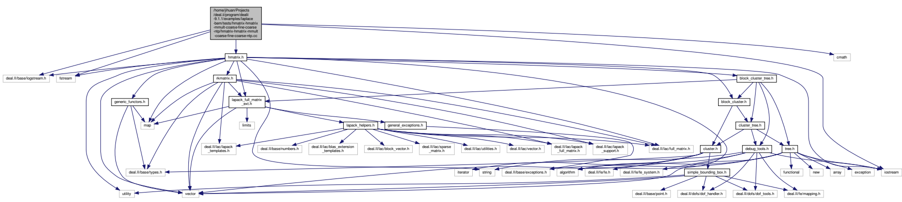
References HMatrix< spacedim, Number >::convertToFullMatrix(), HMatrix< spacedim, Number >::mmult(), LAPACKFullMatrixExt< Number >::mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.40 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-hmatrix-mmult-coarse-fine-coarse-ntp/hmatrix-hmatrix-mmult-coarse-fine-coarse-ntp.cc File Reference

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have coarse-fine-coarse non-tensor product partitions.

```
#include <deal.II/base/logstream.h>
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-hmatrix-mmult-coarse-fine-coarse-ntp.cc:



Functions

- int main ()

9.40.1 Detailed Description

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have coarse-fine-coarse non-tensor product partitions.

Author

Jihuan Tian

Date

2021-08-26

9.40.2 Function Documentation

9.40.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create two full matrices as the source data.

Create the two H-matrices `H1` and `H2` from `M1` and `M2`.

Get the full matrix representations of `H1` and `H2` as well as their product.

Multiply the two H-matrices `H1` and `H2`.

Convert the result matrix into a full matrix for verification.

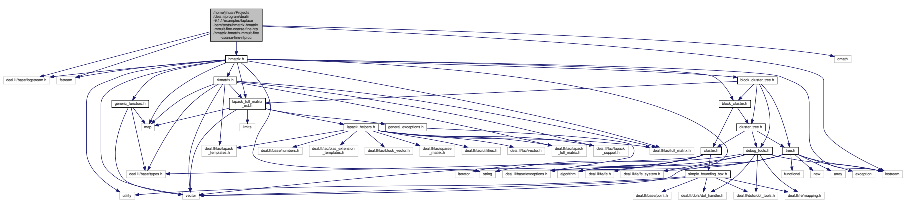
References HMatrix< spacedim, Number >::convertToFullMatrix(), HMatrix< spacedim, Number >::mmult(), LAPACKFullMatrixExt< Number >::mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.41 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-hmatrix-mmult-fine-coarse-fine-ntp/hmatrix-hmatrix-mmult-fine-coarse-fine-ntp.cc File Reference

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have fine-coarse-fine non-tensor product partitions.

```
#include <deal.II/base/logstream.h>
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-hmatrix-mmult-fine-coarse-fine-ntp.cc:



Functions

- int main ()

9.41.1 Detailed Description

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have fine-coarse-fine non-tensor product partitions.

Author

Jihuan Tian

Date

2021-08-26

9.41.2 Function Documentation

9.41.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create two full matrices as the source data.

Create the two H-matrices `H1` and `H2` from `M1` and `M2`.

Get the full matrix representations of `H1` and `H2` as well as their product.

Multiply the two H-matrices `H1` and `H2`.

Convert the result matrix into a full matrix for verification.

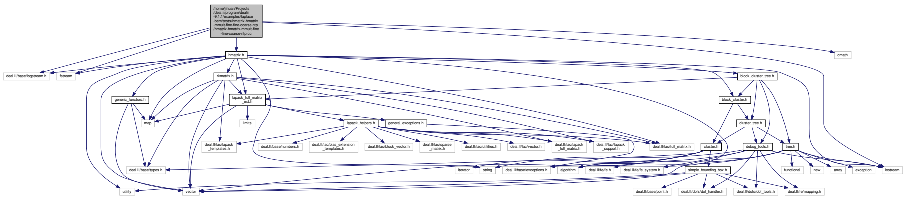
References HMatrix< spacedim, Number >::convertToFullMatrix(), HMatrix< spacedim, Number >::mmult(), LAPACKFullMatrixExt< Number >::mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.42 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-hmatrix-mmult-fine-fine-coarse-ntp/hmatrix-hmatrix-mmult-fine-fine-coarse-ntp.cc File Reference

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have fine-fine-coarse non-tensor product partitions.

```
#include <deal.II/base/logstream.h>
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-hmatrix-mmult-fine-fine-coarse-ntp.cc:



Functions

- int main ()

9.42.1 Detailed Description

Verify the multiplication of two $\mathcal{H}$-matrices, where M1, M2 and M have fine-fine-coarse non-tensor product partitions.

Author

Jihuan Tian

Date

2021-08-26

9.42.2 Function Documentation

9.42.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create two full matrices as the source data.

Create the two H-matrices `H1` and `H2` from `M1` and `M2`.

Get the full matrix representations of `H1` and `H2` as well as their product.

Multiply the two H-matrices `H1` and `H2`.

Convert the result matrix into a full matrix for verification.

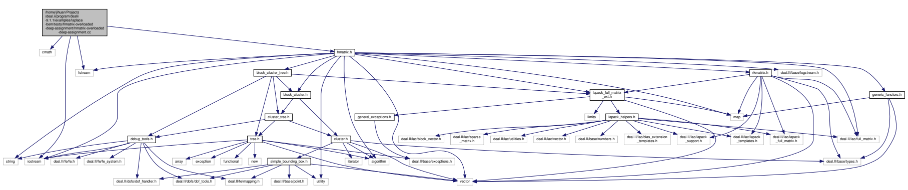
References HMatrix< spacedim, Number >::convertToFullMatrix(), HMatrix< spacedim, Number >::mmult(), LAPACKFullMatrixExt< Number >::mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_coarse_non_tensor_product(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.43 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-overloaded-deep-assignment/hmatrix-overloaded-deep-assignment.cc File Reference

Verify the overloaded deep assignment operator of $\mathcal{H}$-matrix.

```
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-overloaded-deep-assignment.cc:



Functions

- int main ()

9.43.1 Detailed Description

Verify the overloaded deep assignment operator of $\mathcal{H}$-matrix.

Author

Jihuan Tian

Date

2021-08-24

9.43.2 Function Documentation

9.43.2.1 main()

`int main ( )`

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create a full matrix as the source data.

Create an H-matrix from the block cluster tree.

Create an H-matrix using deep assignment.

Try to print H1. Because the data of H1 have been migrated to H2, an exception will be thrown from this function call.

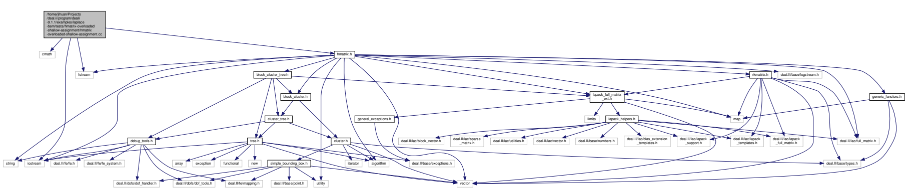
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition↩_fine_non_tensor_product(), HMatrix< spacedim, Number >::print_formatted(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.44 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-overloaded-shallow-assignment/hmatrix-overloaded-shallow-assignment.cc File Reference

Verify the overloaded shallow assignment operator of $\mathcal{H}$-matrix.

```
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-overloaded-shallow-assignment.cc:



Functions

- int main ()

9.44.1 Detailed Description

Verify the overloaded shallow assignment operator of $\mathcal{H}$-matrix.

Author

Jihuan Tian

Date

2021-08-24

9.44.2 Function Documentation

9.44.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create a full matrix as the source data.

Create an H-matrix from the block cluster tree.

Create an H-matrix using shallow assignment.

Try to print H1. Because the data of H1 have been migrated to H2, an exception will be thrown from this function call.

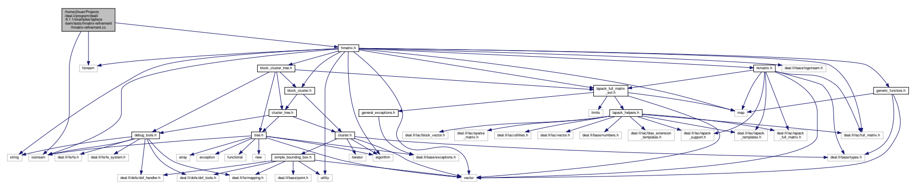
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition↩_fine_non_tensor_product(), HMatrix< spacedim, Number >::print_formatted(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.45 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-refinement/hmatrix-refinement.cc File Reference

Verify the refinement of an $\mathcal{H}$-matrix hierarchy with respect to its extended block cluster tree.

```
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-refinement.cc:



Functions

- int main ()

9.45.1 Detailed Description

Verify the refinement of an $\mathcal{H}$-matrix hierarchy with respect to its extended block cluster tree.

Author

Jihuan Tian

Date

2021-07-29

9.45.2 Function Documentation

9.45.2.1 main()

```
int main ( )
```

Set the dimension of the H^p-matrix to be built.

Set the minimum cluster size.

Generate the cluster tree using cardinality based partition.

Generate two block cluster trees with different depth.

Print out the partition structures of the two block cluster trees.

Create a full matrix as the data source.

Create an $\mathcal{H}$-matrix based on the coarse block cluster tree.

Transform the $\mathcal{H}$-matrix to full matrix for verification.

Extend the block cluster tree associated with the $\mathcal{H}$-matrix to a finer partition.

Refine the $\mathcal{H}$-matrix.

Transform the $\mathcal{H}$-matrix to full matrix for verification.

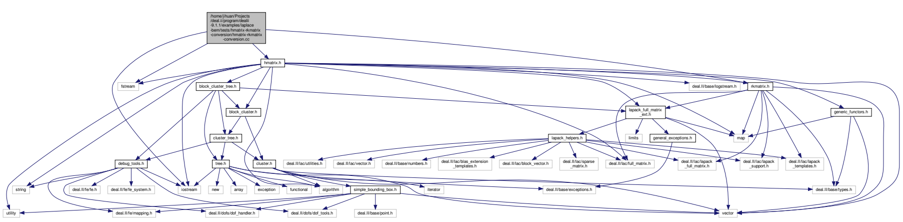
References HMatrix< spacedim, Number >::convertToFullMatrix(), BlockClusterTree< spacedim, Number >↩ ::extend_to_finer_partition(), BlockClusterTree< spacedim, Number >::get_leaf_set(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPA↩ CKFullMatrixExt< Number >::print_formatted_to_mat(), HMatrix< spacedim, Number >::refine_to_supertree(), BlockClusterTree< spacedim, Number >::write_leaf_set(), and HMatrix< spacedim, Number >::write_leaf_set_↩ by_iteration().

9.46 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-rkmatrix-conversion/hmatrix-rkmatrix-conversion.cc File Reference

Verify the conversion from an H-matrix to a rank-k matrix.

```
#include <fstream>
#include <iostream>
#include "hmatrix.h"
#include "rkmatrix.h"
```

Include dependency graph for hmatrix-rkmatrix-conversion.cc:



Functions

- int main ()

9.46.1 Detailed Description

Verify the conversion from an H-matrix to a rank-k matrix.

Author

Jihuan Tian

Date

2021-07-09

9.46.2 Function Documentation

9.46.2.1 main()

```
int main ( )
```

Create a full matrix as the data source.

Create an HMatrix of fixed rank 2 using the fine non-tensor product partition. In practice, the rank may not be a constant but a block dependent distribution or map.

Print the initial partition structure.

Convert the H-matrix to a rank-2 matrix.

Extract the resulted matrix, which is either a rank-k matrix or a full matrix.

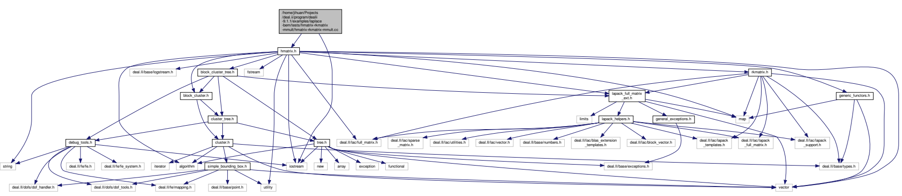
References convertHMatBlockToRkMatrix(), FullMatrixType, HMatrix< spacedim, Number >::get_fullmatrix(), HMatrix< spacedim, Number >::get_rkmatrix(), HMatrix< spacedim, Number >::get_type(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), RkMatrixType, and HMatrix< spacedim, Number >::write_leaf_set().

9.47 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-rkmatrix-mmult/hmatrix-rkmatrix-mmult.cc File Reference

Verify the H-matrix/rank-k matrix multiplication.

```
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-rkmatrix-mmult.cc:



Functions

- int main ()

9.47.1 Detailed Description

Verify the H-matrix/rank-k matrix multiplication.

Author

Jihuan Tian

Date

2021-08-14

9.47.2 Function Documentation

9.47.2.1 main()

```
int main ( )
```

Generate the DoF index set.

Construct the cluster tree.

Construct the block cluster tree using fine non-tensor product partition.

Create a full matrix for initializing the $\mathcal{H}$-matrix.

Create an $\mathcal{H}$-matrix based on the block cluster tree.

Create the second full matrix for initializing the rank-k matrix.

Create a rank-k matrix from `M2`.

Perform matrix-matrix multiplication.

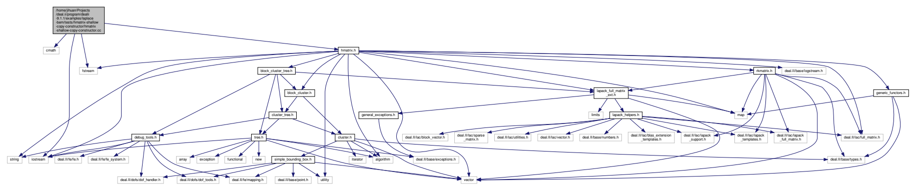
References h_rk_mmult(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), RkMatrix< Number >::print_formatted_to_mat(), and LAPACKFull↩MatrixExt< Number >::print_formatted_to_mat().

9.48 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-shallow-copy-constructor/hmatrix-shallow-copy-constructor.cc File Reference

Verify the deep copy constructor of $\mathcal{H}$-matrix.

```
#include <cmath>
#include <fstream>
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-shallow-copy-constructor.cc:



Functions

- int main ()

9.48.1 Detailed Description

Verify the deep copy constructor of $\mathcal{H}$-matrix.

Author

Jihuan Tian

Date

2021-08-24

9.48.2 Function Documentation

9.48.2.1 main()

```
int main ( )
```

Generate index set.

Generate cluster tree.

Generate block cluster tree via fine structured non-tensor product partition.

Create a full matrix as the source data.

Create an H-matrix from the block cluster tree.

Create an H-matrix using shallow copy.

Try to print H1. Because the data of H1 have been migrated to H2, an exception will be thrown from this function call.

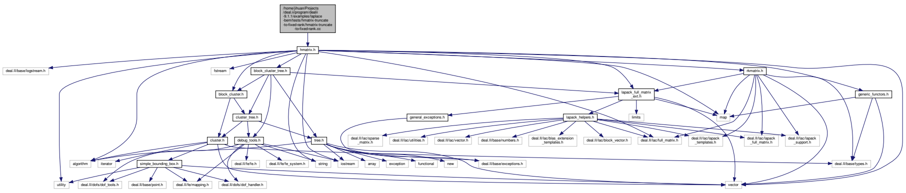
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition↩_fine_non_tensor_product(), HMatrix< spacedim, Number >::print_formatted(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.49 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-truncate-to-fixed-rank/hmatrix-truncate-to-fixed-rank.cc File Reference

Verify the truncation of an HMatrix to an RkMatrix.

```
#include "hmatrix.h"
```

Include dependency graph for hmatrix-truncate-to-fixed-rank.cc:



Functions

- int main ()

9.49.1 Detailed Description

Verify the truncation of an HMatrix to an RkMatrix.

Author

Jihuan Tian

Date

2021-06-24

9.49.2 Function Documentation

9.49.2.1 main()

```
int main ( )
```

Create a full matrix as the data source.

Create an HMatrix of fixed rank 2 using the fine non-tensor product partition. In practice, the rank may not be a constant but a block dependent distribution or map.

Truncate the HMatrix to fixed rank 1.

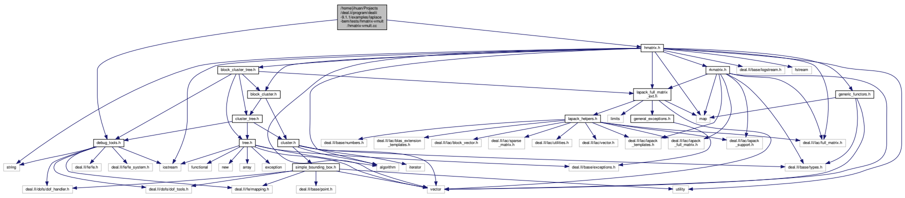
Convert the HMatrix back to a full matrix.

References HMatrix< spacedim, Number >::convertToFullMatrix(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), HMatrix< spacedim, Number >::print_formatted(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and HMatrix< spacedim, Number >::truncate_to_rank().

9.50 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-vmult/hmatrix-vmult.cc File Reference

```
#include "debug_tools.h"
#include "hmatrix.h"
```

Include dependency graph for hmatrix-vmult.cc:



Functions

- int main ()

9.50.1 Detailed Description

Author

Jihuan Tian

Date

2021-06-23

9.50.2 Function Documentation

9.50.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a rank-1 HMatrix.

Create the vector x.

Perform matrix-vector multiplication.

Perform transposed matrix-vector multiplication.

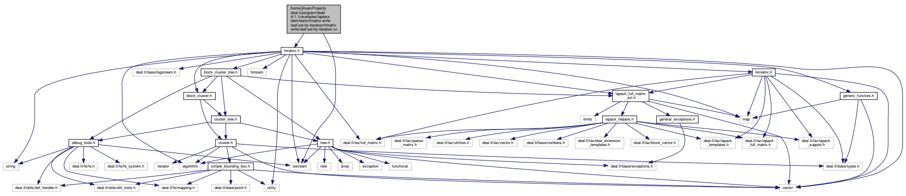
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and LAPACKFullMatrixExt< Number >::print_formatted_to_mat().

9.51 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hmatrix-write-leaf-set-by-iteration/hmatrix-write-leaf-set-by-iteration.cc File Reference

Verify the method for write out leaf set by iteration over the constructed leaf set instead of recursion.

```
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for hmatrix-write-leaf-set-by-iteration.cc:



Functions

- int main ()

9.51.1 Detailed Description

Verify the method for write out leaf set by iteration over the constructed leaf set instead of recursion.

Author

Jihuan Tian

Date

2021-07-28

9.51.2 Function Documentation

9.51.2.1 main()

```
int main ( )
```

Create the global index set.

Construct the cluster tree.

Construct the block cluster tree.

Create a full matrix with data.

Create a rank-1 HMatrix.

Write out the leaf nodes by iteration.

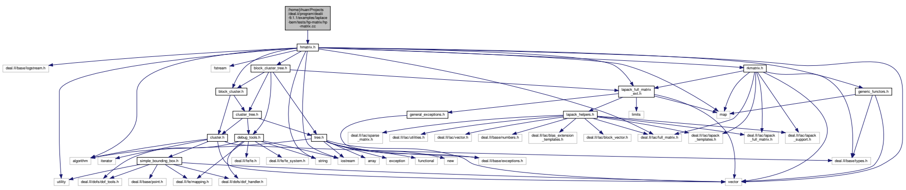
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and HMatrix< spacedim, Number >::write_leaf_set_by_iteration().

9.52 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/hp-matrix/hp-matrix.cc File Reference

Test the H^p matrix.

```
#include "hmatrix.h"
```

Include dependency graph for hp-matrix.cc:



Functions

- int main ()

9.52.1 Detailed Description

Test the H^p matrix.

Author

Jihuan Tian

Date

2021-06-23

9.52.2 Function Documentation

9.52.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a rank-1 HMatrix.

Convert the rank-1 HMatrix back to a full matrix.

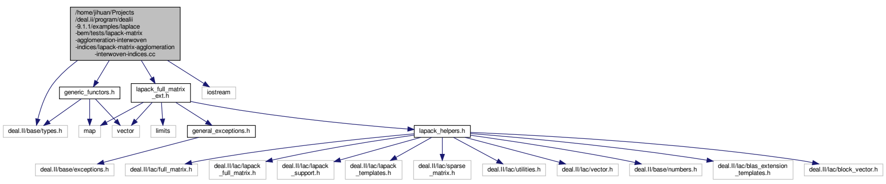
References HMatrix< spacedim, Number >::convertToFullMatrix(), ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_fine_non_tensor_product(), and HMatrix< spacedim, Number >::print_formatted().

9.53 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-agglomeration-interwoven-indices/lapack-matrix-agglomeration-interwoven-indices.cc File Reference

Verify the agglomeration of four full submatrices into a larger full matrix when the index sets of several child clusters are interwoven together into the index set of the parent cluster.

```
#include <deal.II/base/types.h>
#include <iostream>
#include "generic_functors.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-agglomeration-interwoven-indices.cc:



Functions

- int main ()

9.53.1 Detailed Description

Verify the agglomeration of four full submatrices into a larger full matrix when the index sets of several child clusters are interwoven together into the index set of the parent cluster.

Author

Jihuan Tian

Date

2021-07-31

9.53.2 Function Documentation

9.53.2.1 main()

```
int main ( )
```

Generate submatrices `M11`, `M12`, `M21`, `M22` with their row and column index sets for agglomeration.

Output the four submatrices.

Define the row and column index sets of the large matrix `M`, which has a dimension 5×5.

Build the global to local indices map for the matrix `M`.

Output the agglomerated matrix `M`.

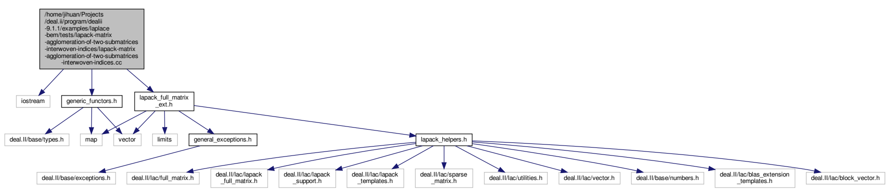
References build_index_set_global_to_local_map(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.54 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-agglomeration-of-two-submatrices-interwoven-indices/lapack-matrix-agglomeration-of-two-submatrices-interwoven-indices.cc File Reference

Verify the agglomeration of two full submatrices which have been obtained from horizontal or vertical splitting. The index sets of several child clusters are interwoven together into the index set of the parent cluster.

```
#include <iostream>
#include "generic_functors.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-agglomeration-of-two-submatrices-interwoven-indices.cc:



Functions

- int main ()

9.54.1 Detailed Description

Verify the agglomeration of two full submatrices which have been obtained from horizontal or vertical splitting. The index sets of several child clusters are interwoven together into the index set of the parent cluster.

Author

Jihuan Tian

Date

2021-08-04

9.54.2 Function Documentation

9.54.2.1 main()

```
int main ( )
```

Define the row and column index sets of the large matrix `M_agglomerated1` which is obtained from vertically split submatrices.

Define the row and column index sets of the large matrix `M_agglomerated2` which is obtained from horizontally split submatrices.

Build the global to local indices maps.

After agglomeration, we should have `2 8 4 10 6 1 7 3 9 5`

After agglomeration, we should have `3 5 1 12 13 11 4 6 2`

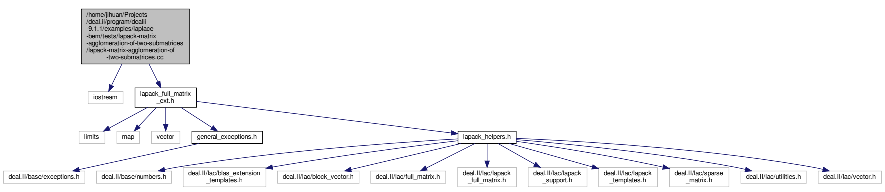
References build_index_set_global_to_local_map(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.55 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-agglomeration-of-two-submatrices/lapack-matrix-agglomeration-of-two-submatrices.cc File Reference

Verify the agglomeration of two full submatrices which have been obtained from horizontal or vertical splitting.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-agglomeration-of-two-submatrices.cc:



Functions

- int **main** ()

9.55.1 Detailed Description

Verify the agglomeration of two full submatrices which have been obtained from horizontal or vertical splitting.

Author

Jihuan Tian

Date

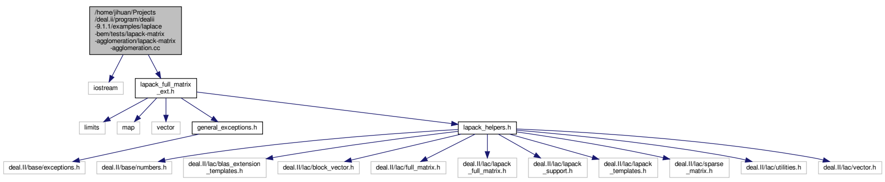
2021-08-04

9.56 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-agglomeration/lapack-matrix-agglomeration.cc File Reference

Verify the agglomeration of four full submatrices into a larger full matrix.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-agglomeration.cc:



Functions

- int main ()

9.56.1 Detailed Description

Verify the agglomeration of four full submatrices into a larger full matrix.

Author

Jihuan Tian

Date

2021-07-08

9.56.2 Function Documentation

9.56.2.1 main()

```
int main ( )
```

Output the matrices.

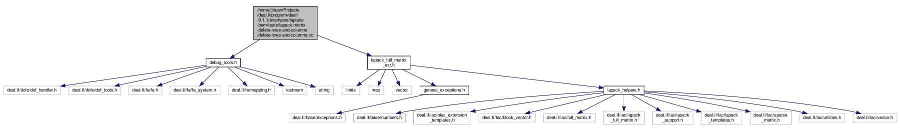
References LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.57 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-delete-rows-and-columns/delete-rows-and-columns.cc File Reference

Test deleting rows and columns as well as keeping the first `n` rows or columns from a LAPACKFullMatrixExt. .

```
#include "debug_tools.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for delete-rows-and-columns.cc:



Functions

- int **main** ()

9.57.1 Detailed Description

Test deleting rows and columns as well as keeping the first `n` rows or columns from a LAPACKFullMatrixExt. .

Date

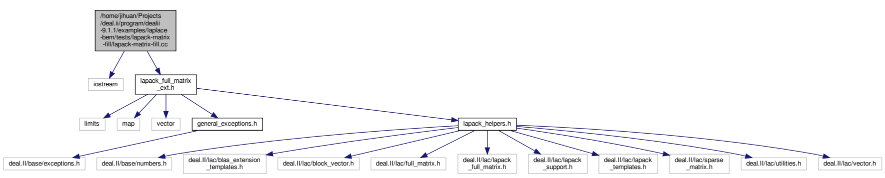
2021-06-19

9.58 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-fill/lapack-matrix-fill.cc File Reference

Verify filling a LAPACKFullMatrixExt from a source matrix.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-fill.cc:



Functions

- int **main** ()

9.58.1 Detailed Description

Verify filling a LAPACKFullMatrixExt from a source matrix.

Author

Jihuan Tian

Date

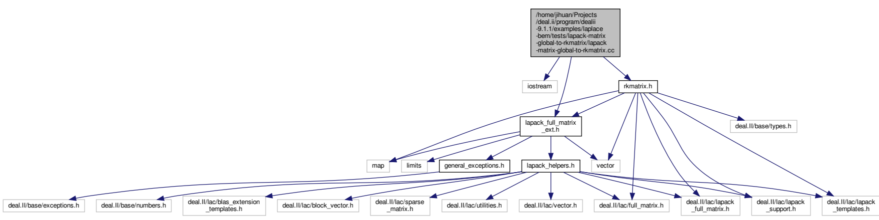
2021-06-30

9.59 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-global-to-rkmatrix/lapack-matrix-global-to-rkmatrix.cc File Reference

Verify the restriction of a global full matrix to a rank-k submatrix.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for lapack-matrix-global-to-rkmatrix.cc:



Functions

- int main ()

9.59.1 Detailed Description

Verify the restriction of a global full matrix to a rank-k submatrix.

Author

Jihuan Tian

Date

2021-07-28

9.59.2 Function Documentation

9.59.2.1 main()

```
int main ( )
```

Create a full matrix with data.

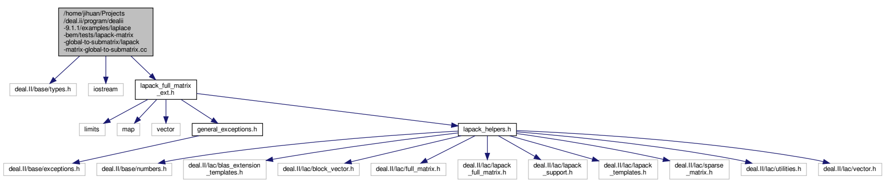
References RkMatrix< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::print_↩
formatted_to_mat().

9.60 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-global-to-submatrix/lapack-matrix-global-to-submatrix.cc File Reference

Verify the restriction of a global full matrix to sub full matrix.

```
#include <deal.II/base/types.h>
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-global-to-submatrix.cc:



Functions

- int main ()

9.60.1 Detailed Description

Verify the restriction of a global full matrix to sub full matrix.

Author

Jihuan Tian

Date

2021-07-28

9.60.2 Function Documentation

9.60.2.1 main()

```
int main ( )
```

Create a full matrix with data.

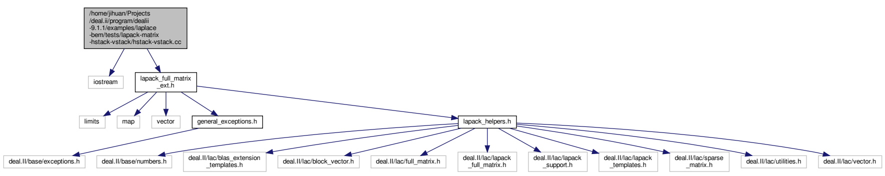
References LAPACKFullMatrixExt< Number >::print_formatted_to_mat().

9.61 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-hstack-vstack/hstack-vstack.cc File Reference

Verify horizontal and vertical stacking of two LAPACKFullMatrixExt objects.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for hstack-vstack.cc:



Functions

- int **main** ()

9.61.1 Detailed Description

Verify horizontal and vertical stacking of two LAPACKFullMatrixExt objects.

Author

Jihuan Tian

Date

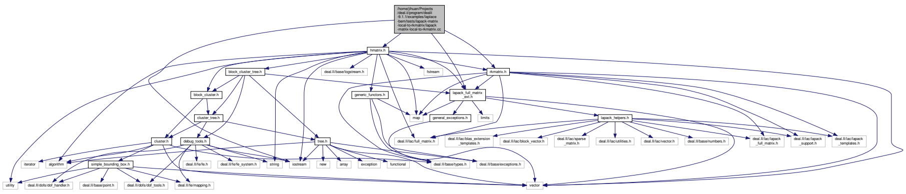
2021-06-30

9.62 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-local-to-rkmatrix/lapack-matrix-local-to-rkmatrix.cc File Reference

Verify the restriction of a local full matrix to a rank-k submatrix.

```
#include <iostream>
#include "hmatrix.h"
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for lapack-matrix-local-to-rkmatrix.cc:



Functions

- int main ()

9.62.1 Detailed Description

Verify the restriction of a local full matrix to a rank-k submatrix.

Author

Jihuan Tian

Date

2021-07-28

9.62.2 Function Documentation

9.62.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a local matrix as a sub matrix of the original matrix.

Build the maps from global row and column indices respectively to local indices wrt. M_b.

Create a sub rank-k matrix of `M_b` by specifying its block cluster as a subset of the block cluster $\tau \times \sigma$ for `M_b`.

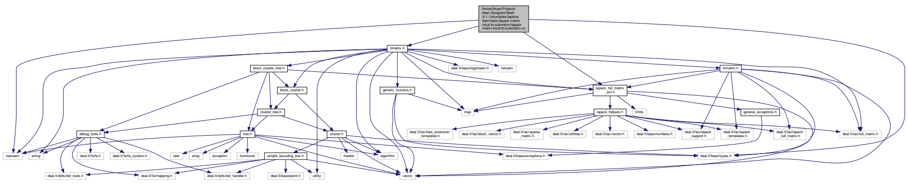
References build_index_set_global_to_local_map(), RkMatrix< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::print_formatted_to_mat().

9.63 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-local-to-submatrix/lapack-matrix-local-to-submatrix.cc File Reference

Verify the restriction of a local full matrix to sub full matrix.

```
#include <deal.II/base/types.h>
#include <iostream>
#include "hmatrix.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-local-to-submatrix.cc:



Functions

- int main ()

9.63.1 Detailed Description

Verify the restriction of a local full matrix to sub full matrix.

Author

Jihuan Tian

Date

2021-07-28

9.63.2 Function Documentation

9.63.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a local matrix as a sub matrix of the original matrix.

Build the maps from global row and column indices respectively to local indices wrt. M_b.

Create a sub full matrix of `M_b` by specifying its block cluster as a subset of the block cluster $\tau \times \sigma$ for `M_b`.

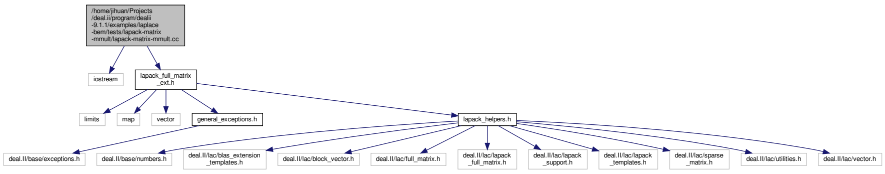
References build_index_set_global_to_local_map(), and LAPACKFullMatrixExt< Number >::print_formatted_to↩_mat().

9.64 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-mmult/lapack-matrix-mmult.cc File Reference

Verify the multiplication of two `LAPACKFullMatrixExt`.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-mmult.cc:



Functions

- int **main** ()

9.64.1 Detailed Description

Verify the multiplication of two `LAPACKFullMatrixExt`.

Author

Jihuan Tian

Date

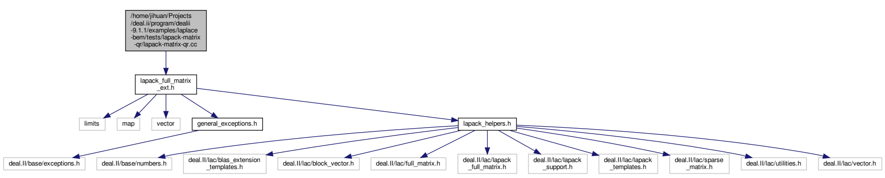
2021-08-19

9.65 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-qr/lapack-matrix-qr.cc File Reference

Verify QR decomposition.

```
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-qr.cc:



Functions

- int main ()

9.65.1 Detailed Description

Verify QR decomposition.

Author

Jihuan Tian

Date

2021-07-02

9.65.2 Function Documentation

9.65.2.1 main()

```
int main ( )
```

QR decomposition of a matrix with more columns than rows.

QR decomposition of a matrix with more rows than columns.

Reduced QR decomposition of a matrix with more rows than columns.

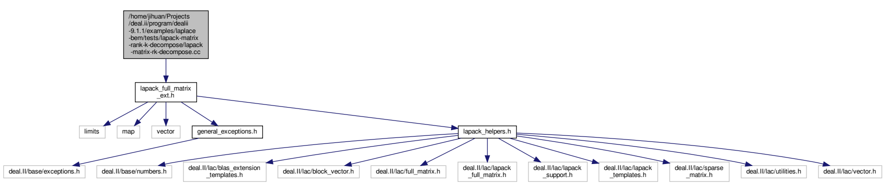
References LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >::qr(), LAPACKFullMatrixExt< Number >::reduced_qr(), and LAPACKFullMatrixExt< Number >::Reshape().

9.66 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-rank-k-decompose/lapack-matrix-rk-decompose.cc File Reference

Verify decomposition of a full matrix into the two components of a rank-k matrix.

```
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-rk-decompose.cc:



Functions

- int main ()

9.66.1 Detailed Description

Verify decomposition of a full matrix into the two components of a rank-k matrix.

Author

Jihuan Tian

Date

2021-07-05

9.66.2 Function Documentation

9.66.2.1 main()

```
int main ( )
```

Construct a full matrix.

Decompose the full matrix into the product of `A` and `B^T` with rank truncated to 1.

Decompose the full matrix into the product of `A` and `B^T` with rank truncated to 2.

Decompose the full matrix into the product of `A` and `B^T` with rank truncated to 3.

Decompose the full matrix into the product of `A` and `B^T` with rank truncated to 4.

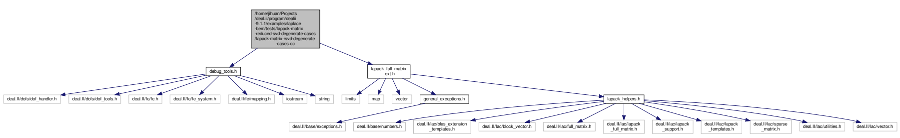
References LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >↩
::rank_k_decompose(), and LAPACKFullMatrixExt< Number >::Reshape().

9.67 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-reduced-svd-degenerate-cases/lapack-matrix-rsvd-degenerate-cases.cc File Reference

Verify degenerate cases for reduced SVD.

```
#include "debug_tools.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-rsvd-degenerate-cases.cc:



Functions

- int main ()

9.67.1 Detailed Description

Verify degenerate cases for reduced SVD.

Author

Jihuan Tian

Date

2021-07-05

9.67.2 Function Documentation

9.67.2.1 main()

```
int main ( )
```

RSVD of a scalar matrix.

RSVD of a row matrix.

RSVD of a column matrix.

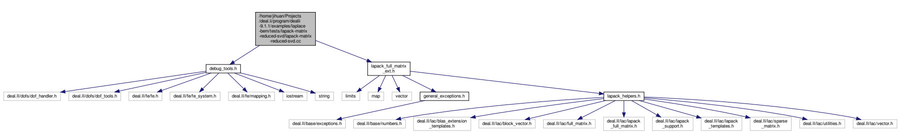
References LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::reduced_svd().

9.68 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-reduced-svd/lapack-matrix-reduced-svd.cc File Reference

Verify reduced SVD.

```
#include "debug_tools.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-reduced-svd.cc:



Functions

- int **main** ()

9.68.1 Detailed Description

Verify reduced SVD.

Author

Jihuan Tian

Date

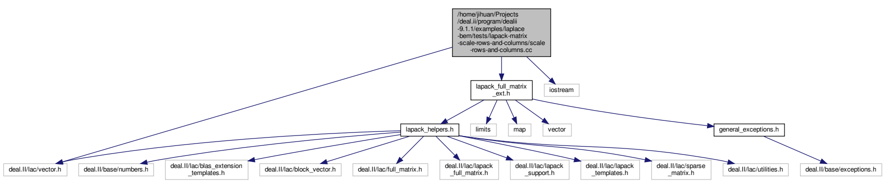
2021-07-05

9.69 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-scale-rows-and-columns/scale-rows-and-columns.cc File Reference

Test scaling rows and columns of a LAPACKFullMatrixExt, which is actually left and right multiplication with a diagonal matrix.

```
#include <deal.II/lac/vector.h>
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for scale-rows-and-columns.cc:



Functions

- int main ()

9.69.1 Detailed Description

Test scaling rows and columns of a LAPACKFullMatrixExt, which is actually left and right multiplication with a diagonal matrix.

Author

Jihuan Tian

Date

2021-06-20

9.69.2 Function Documentation

9.69.2.1 main()

```
int main ( )
```

Diagonal matrix.

Matrix dimension.

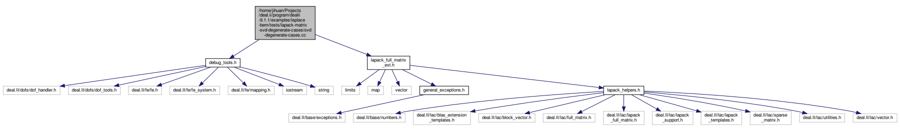
References LAPACKFullMatrixExt< Number >::ConstantMatrix(), LAPACKFullMatrixExt< Number >::scale_columns(), and LAPACKFullMatrixExt< Number >::scale_rows().

9.70 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-svd-degenerate-cases/svd-degenerate-cases.cc File Reference

Test SVD and RSVD for degenerate cases, such as the matrix is a scalar, row vector or column vector.

```
#include "debug_tools.h"
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for svd-degenerate-cases.cc:



Functions

- int main ()

9.70.1 Detailed Description

Test SVD and RSVD for degenerate cases, such as the matrix is a scalar, row vector or column vector.

Author

Jihuan Tian

Date

2021-06-24

9.70.2 Function Documentation

9.70.2.1 main()

```
int main ( )
```

SVD of a scalar matrix.

SVD of a row matrix.

SVD of a column matrix.

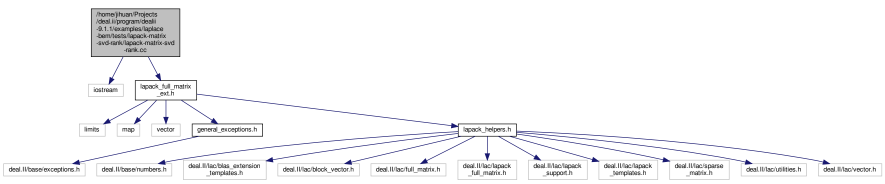
References LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::svd().

9.71 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-svd-rank/lapack-matrix-svd-rank.cc File Reference

Verify matrix rank calculation using SVD.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
```

Include dependency graph for lapack-matrix-svd-rank.cc:



Functions

- int **main** ()

9.71.1 Detailed Description

Verify matrix rank calculation using SVD.

Author

Jihuan Tian

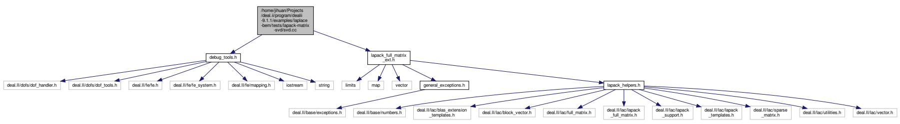
Date

2021-07-04

9.72 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-svd/svd.cc File Reference

Test singular value decomposition (SVD) and reduced SVD.

```
#include "debug_tools.h"
#include "lapack_full_matrix_ext.h"
```
Include dependency graph for svd.cc:



Functions

- int **main** ()

9.72.1 Detailed Description

Test singular value decomposition (SVD) and reduced SVD.

Author

Jihuan Tian

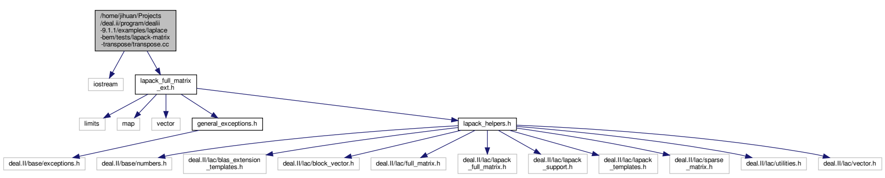
Date

2021-06-19

9.73 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/lapack-matrix-transpose/transpose.cc File Reference

Test in-place transpose of a LAPACKFullMatrixExt.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
```
Include dependency graph for transpose.cc:



Functions

- int **main** ()

9.73.1 Detailed Description

Test in-place transpose of a LAPACKFullMatrixExt.

Author

Jihuan Tian

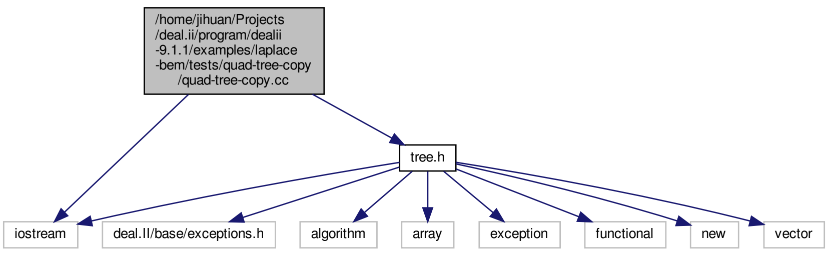
Date

2021-06-20

9.74 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/quad-tree-copy/quad-tree-copy.cc File Reference

Verify.

```
#include <iostream>
#include "tree.h"
```
Include dependency graph for quad-tree-copy.cc:



Functions

- TreeNode$<$ int, 4 $>$ $*$ MakeIntExampleTree ()
- int **main** ()

9.74.1 Detailed Description

Verify.

Author

Jihuan Tian

Date

2021-07-20

9.74.2 Function Documentation

9.74.2.1 MakeIntExampleTree()

```
TreeNode<int, 4>* MakeIntExampleTree ( )
```

Create an example tree containing integers as below. The tree will be constructed in a bottom-up approach.

```
      10
  /   |   |   \
 3    4   5    6
```

/ | | \ 2 8 7 9

The function returns the pointer to the root node of the tree.

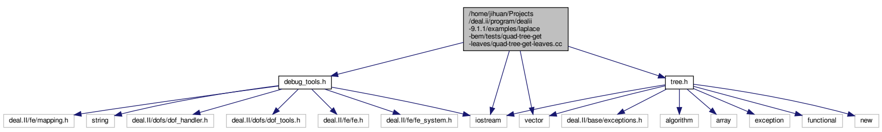
References TreeNode< T, N >::set_split_mode().

9.75 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/quad-tree-get-leaves/quad-tree-get-leaves.cc File Reference

Verify extraction of the leaves of a tree.

```
#include <iostream>
#include <vector>
#include "debug_tools.h"
#include "tree.h"
```

Include dependency graph for quad-tree-get-leaves.cc:



Functions

- TreeNode< int, 4 > ∗ MakeIntExampleTree ()
- int **main** ()

9.75.1 Detailed Description

Verify extraction of the leaves of a tree.

Author

Jihuan Tian

Date

2021-07-21

9.75.2 Function Documentation

9.75.2.1 MakeIntExampleTree()

```
TreeNode<int, 4>* MakeIntExampleTree ( )
```

Create an example tree containing integers as below. The tree will be constructed in a bottom-up approach.

```
       10
  /   |   |   \
 3    4   5    6
```

/ | | \ 2 8 7 9

The function returns the pointer to the root node of the tree.

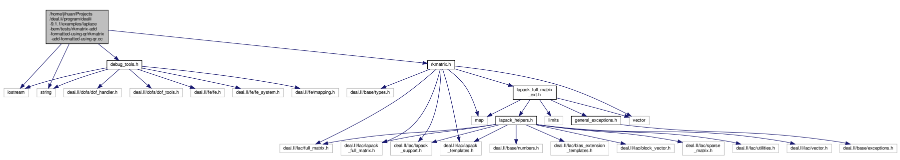
References TreeNode< T, N >::set_split_mode().

9.76 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-add-formatted-using-qr/rkmatrix-add-formatted-using-qr.cc File Reference

Verify the formatted addition of two rank-k matrices by using the QR decomposition. This requires that the component matrices of rank-k matrices should wide matrix, i.e. having more rows than columns.

```
#include <iostream>
#include <string>
#include "debug_tools.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-add-formatted-using-qr.cc:



Functions

- int main ()

9.76.1 Detailed Description

Verify the formatted addition of two rank-k matrices by using the QR decomposition. This requires that the component matrices of rank-k matrices should wide matrix, i.e. having more rows than columns.

N.B. Because the data of the two operand matrices are created by hand, it cannot ensure a exponential decrease of singular values like that generated from a integral operator. Hence, the approximation accuracy is not high.

Author

Jihuan Tian

Date

2021-07-02

9.76.2 Function Documentation

9.76.2.1 main()

```
int main ( )
```

Create two full matrices as the data source.

Create two rank-k matrices converted from the two matrices.

Perform formatted addition.

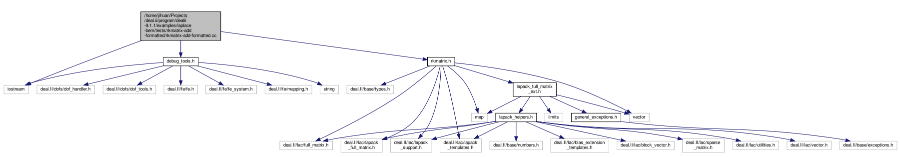
References RkMatrix< Number >::add(), RkMatrix< Number >::print_formatted_to_mat(), LAPACKFullMatrix↩ Ext< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.77 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-add-formatted/rkmatrix-add-formatted.cc File Reference

Verify the formatted addition of two rank-k matrices.

```
#include <iostream>
#include "debug_tools.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-add-formatted.cc:



Functions

- int main ()

9.77.1 Detailed Description

Verify the formatted addition of two rank-k matrices.

Author

Jihuan Tian

Date

2021-06-30

9.77.2 Function Documentation

9.77.2.1 main()

```
int main ( )
```

Create two full matrices as the data source.

Create two rank-k matrices converted from the two matrices. N.B. The matrix `M1` has a dimension 3×5 but has a rank 2. Even though the rank-k matrix `A` created from `M1` is declared to have rank 3, the final actual rank is automatically truncated to 2.

Perform raw addition of rank-k matrices via juxtaposition.

Perform formatted addition with different truncation ranks.

Calculate $A = A + B$.

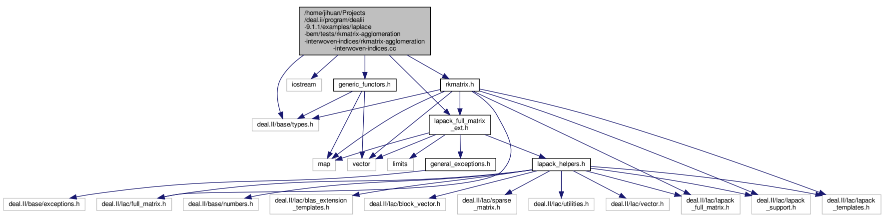
References RkMatrix< Number >::add(), RkMatrix< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.78 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-agglomeration-interwoven-indices/rkmatrix-agglomeration-interwoven-indices.cc File Reference

Verify the agglomeration of four rank-k submatrices into a larger rank-k matrix when the index sets of several child clusters are interwoven together into the index set of the parent cluster.

```
#include <deal.II/base/types.h>
#include <iostream>
#include "generic_functors.h"
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-agglomeration-interwoven-indices.cc:



Functions

- int main ()

9.78.1 Detailed Description

Verify the agglomeration of four rank-k submatrices into a larger rank-k matrix when the index sets of several child clusters are interwoven together into the index set of the parent cluster.

Author

Jihuan Tian

Date

2021-08-04

9.78.2 Function Documentation

9.78.2.1 main()

```
int main ( )
```

Define the row and column index sets of the large matrix M, which has a dimension 5×5.

Build the global to local indices map for the matrix M.

Build the agglomerated full matrix which is to be compared with rank-k matrix.

Output the matrices.

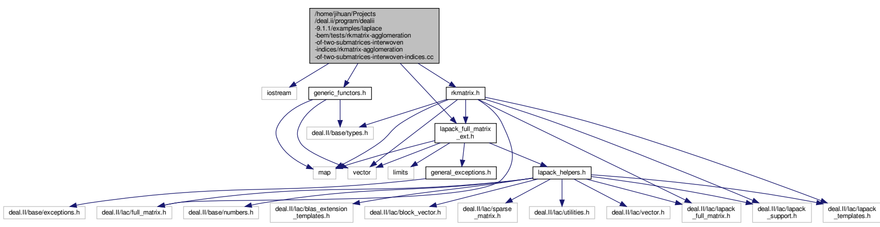
References build_index_set_global_to_local_map(), RkMatrix< Number >::print_formatted_to_mat(), LAPACK←
FullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.79 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-agglomeration-of-two-submatrices-interwoven-indices/rkmatrix-agglomeration-of-two-submatrices-interwoven-indices.cc File Reference

Verify the agglomeration of two rank-k submatrices which have been obtained from horizontal or vertical splitting. The index sets of several child clusters are interwoven together into the index set of the parent cluster.

```
#include <iostream>
#include "generic_functors.h"
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-agglomeration-of-two-submatrices-interwoven-indices.cc:



Functions

- int main ()

9.79.1 Detailed Description

Verify the agglomeration of two rank-k submatrices which have been obtained from horizontal or vertical splitting. The index sets of several child clusters are interwoven together into the index set of the parent cluster.

Author

Jihuan Tian

Date

2021-08-05

9.79.2 Function Documentation

9.79.2.1 main()

```
int main ( )
```

Define the row and column index sets of the large matrix `M_agglomerated1` which is obtained from vertically split submatrices.

Define the row and column index sets of the large matrix `M_agglomerated2` which is obtained from horizontally split submatrices.

Build the global to local indices maps.

Agglomeration of full submatrices, we should have `2 8 4 10 6 1 7 3 9 5`

Agglomeration of full submatrices, we should have `3 5 1 12 13 11 4 6 2`

Create corresponding rank-k submatrices.

Agglomeration of rank-k submatrices.

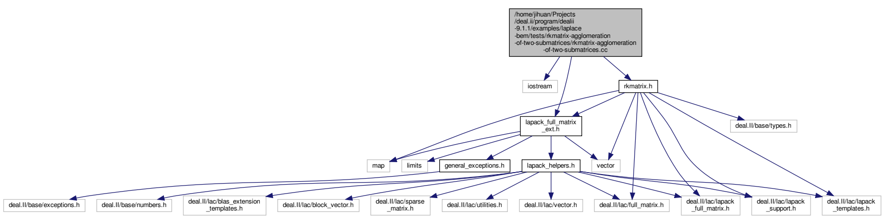
References build_index_set_global_to_local_map(), RkMatrix< Number >::print_formatted_to_mat(), LAPACK↩
FullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.80 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-agglomeration-of-two-submatrices/rkmatrix-agglomeration-of-two-submatrices.cc File Reference

Verify the agglomeration of two rank-k submatrices which have been obtained from horizontal or vertical splitting.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-agglomeration-of-two-submatrices.cc:



Functions

- int main ()

9.80.1 Detailed Description

Verify the agglomeration of two rank-k submatrices which have been obtained from horizontal or vertical splitting.

Author

Jihuan Tian

Date

2021-08-04

9.80.2 Function Documentation

9.80.2.1 main()

```
int main ( )
```

Create rank-k matrices from the full matrices.

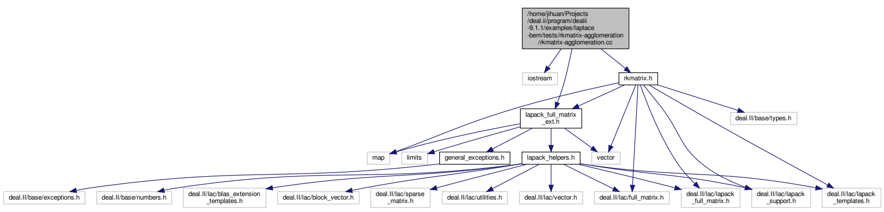
References RkMatrix< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >::print_↩
formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.81 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-agglomeration/rkmatrix-agglomeration.cc File Reference

Verify the agglomeration of four rank-k submatrices into a larger rank-k matrix.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-agglomeration.cc:



Functions

- int main ()

9.81.1 Detailed Description

Verify the agglomeration of four rank-k submatrices into a larger rank-k matrix.

Author

Jihuan Tian

Date

2021-07-08

9.81.2 Function Documentation

9.81.2.1 main()

```
int main ( )
```

Output the matrices.

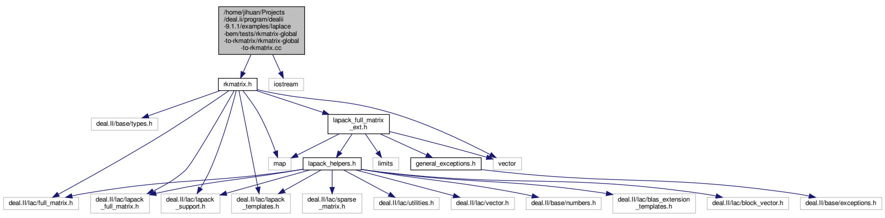
References RkMatrix< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.82 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-global-to-rkmatrix/rkmatrix-global-to-rkmatrix.cc File Reference

Verify the restriction of a global rank-k matrix to a rank-k submatrix.

```
#include "rkmatrix.h"
#include <iostream>
```

Include dependency graph for rkmatrix-global-to-rkmatrix.cc:



Functions

- int main ()

9.82.1 Detailed Description

Verify the restriction of a global rank-k matrix to a rank-k submatrix.

Author

Jihuan Tian

Date

2021-07-28

9.82.2 Function Documentation

9.82.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a rank-k matrix from the full matrix.

Create a rank-k matrix by restriction from the large rank-k matrix on the block cluster $\tau \times \sigma$.

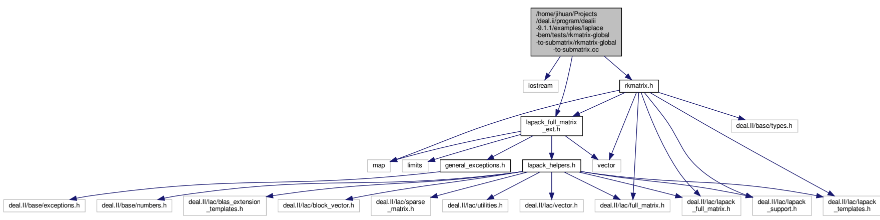
References RkMatrix< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::print_↩ formatted_to_mat().

9.83 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-global-to-submatrix/rkmatrix-global-to-submatrix.cc File Reference

Verify the restriction of a global rank-k matrix to a full submatrix.

```
#include <iostream>
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-global-to-submatrix.cc:



Functions

- int main ()

9.83.1 Detailed Description

Verify the restriction of a global rank-k matrix to a full submatrix.

Author

Jihuan Tian

Date

2021-07-28

9.83.2 Function Documentation

9.83.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a rank-k matrix from the full matrix.

Create a full submatrix matrix by restriction from the large rank-k matrix on the block cluster $\tau \times \sigma$.

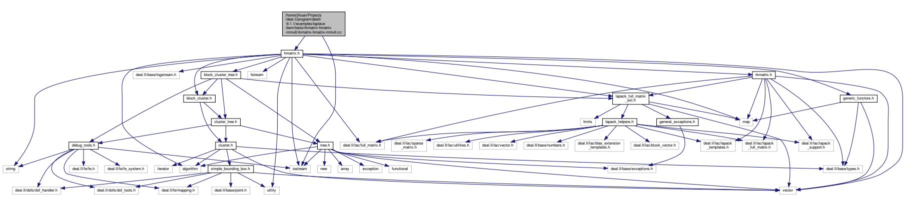
References RkMatrix< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), and RkMatrix< Number >::restrictToFullMatrix().

9.84 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-hmatrix-mmult/rkmatrix-hmatrix-mmult.cc File Reference

Verify the rank-k/H-matrix matrix multiplication.

```
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for rkmatrix-hmatrix-mmult.cc:



Functions

- int main ()

9.84.1 Detailed Description

Verify the rank-k/H-matrix matrix multiplication.

Author

Jihuan Tian

Date

2021-08-14

9.84.2 Function Documentation

9.84.2.1 main()

```
int main ( )
```

Create a full matrix for initializing the first rank-k matrix.

Create a rank-k matrix from `M1`.

Create the second full matrix for initializing the second H-matrix.

Generate the DoF index set.

Construct the cluster tree.

Construct the block cluster tree using fine non-tensor product partition.

Create an $\mathcal{H}$-matrix based on the block cluster tree.

Perform matrix-matrix multiplication.

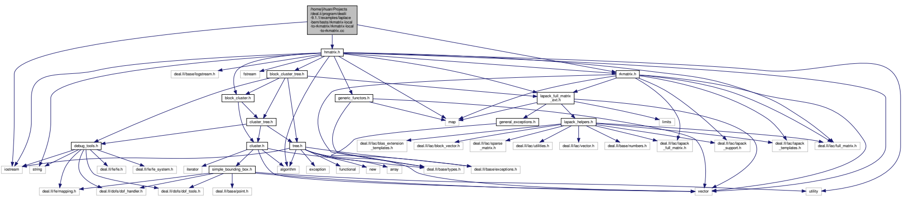
References ClusterTree< spacedim, Number >::partition(), BlockClusterTree< spacedim, Number >::partition_↩ fine_non_tensor_product(), RkMatrix< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >↩ ::print_formatted_to_mat(), and rk_h_mmult().

9.85 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-local-to-rkmatrix/rkmatrix-local-to-rkmatrix.cc File Reference

Verify the restriction of a local rank-k matrix to a rank-k submatrix.

```
#include "rkmatrix.h"
#include <iostream>
#include "hmatrix.h"
```

Include dependency graph for rkmatrix-local-to-rkmatrix.cc:



Functions

- int main ()

9.85.1 Detailed Description

Verify the restriction of a local rank-k matrix to a rank-k submatrix.

Author

Jihuan Tian

Date

2021-07-28

9.85.2 Function Documentation

9.85.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a rank-k matrix by restriction from the global full matrix on the block cluster $\tau \times \sigma$.

Build the maps from global row and column indices respectively to local indices wrt. M_b.

Create a rank-k submatrix of `M_b` by specifying its block cluster as a subset of the block cluster $\tau \times \sigma$ for `M_b`.

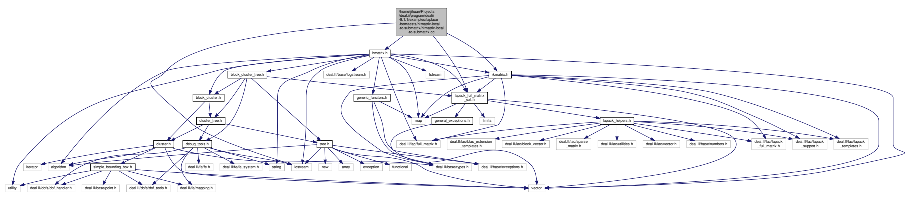
References build_index_set_global_to_local_map(), RkMatrix< Number >::print_formatted_to_mat(), and LAPACKFullMatrixExt< Number >::print_formatted_to_mat().

9.86 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-local-to-submatrix/rkmatrix-local-to-submatrix.cc File Reference

Verify the restriction of a local rank-k matrix to a full submatrix.

```
#include <iostream>
#include "hmatrix.h"
#include "lapack_full_matrix_ext.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-local-to-submatrix.cc:



Functions

- int main ()

9.86.1 Detailed Description

Verify the restriction of a local rank-k matrix to a full submatrix.

Author

Jihuan Tian

Date

2021-07-28

9.86.2 Function Documentation

9.86.2.1 main()

```
int main ( )
```

Create a full matrix with data.

Create a rank-k matrix by restriction from the global full matrix on the block cluster $\tau \times \sigma$.

Build the maps from global row and column indices respectively to local indices wrt. M_b.

Create a full submatrix of `M_b` by specifying its block cluster as a subset of the block cluster $\tau \times \sigma$ for `M_b`.

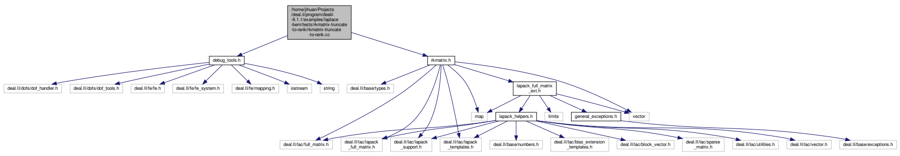
References build_index_set_global_to_local_map(), RkMatrix< Number >::print_formatted_to_mat(), LAPACK↩ FullMatrixExt< Number >::print_formatted_to_mat(), and RkMatrix< Number >::restrictToFullMatrix().

9.87 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix-truncate-to-rank/rkmatrix-truncate-to-rank.cc File Reference

Verify the truncation of an RkMatrix to a given rank.

```
#include "debug_tools.h"
#include "rkmatrix.h"
```

Include dependency graph for rkmatrix-truncate-to-rank.cc:



Functions

- int main ()

9.87.1 Detailed Description

Verify the truncation of an RkMatrix to a given rank.

Author

Jihuan Tian

Date

2021-06-24

9.87.2 Function Documentation

9.87.2.1 main()

```
int main ( )
```

Create a full matrix with rank=2, which is not of full rank.

Convert the full matrix into a rank-3 matrix. Even though rank 3 is required, because it is larger than the effective rank of the full matrix, the rank-k matrix will actually have rank 2, both the formal rank and its actual rank.

Truncate the RkMatrix `A` to rank-3. Because the original rank of `M` is 2 and the created rank-k matrix `A` has a rank 2, no actual rank truncation will be performed here.

Truncate the RkMatrix `A` to rank-2. Because the original rank of `M` is 2 and the created rank-k matrix `A` has a rank 2, no actual rank truncation will be performed here.

Truncate the RkMatrix `A` to rank-1. Because the original rank of `M` is 2 and the created rank-k matrix `A` has a rank 2, rank truncation will be performed.

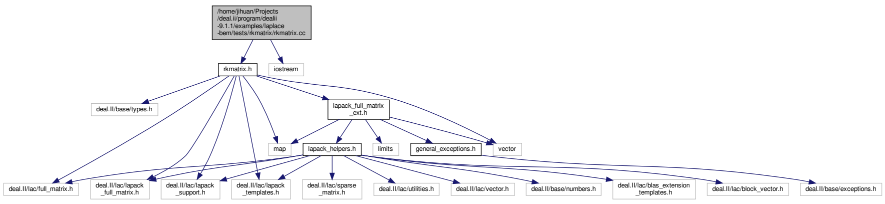
References RkMatrix< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >::print_formatted_to_mat(), LAPACKFullMatrixExt< Number >::Reshape(), and RkMatrix< Number >::truncate_to_rank().

9.88 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/rkmatrix/rkmatrix.cc File Reference

Test RkMatrix class.

```
#include "rkmatrix.h"
#include <iostream>
```

Include dependency graph for rkmatrix.cc:



Functions

- int main ()

9.88.1 Detailed Description

Test RkMatrix class.

Author

Jihuan Tian

Date

2021-06-20

9.88.2 Function Documentation

9.88.2.1 main()

```
int main ( )
```

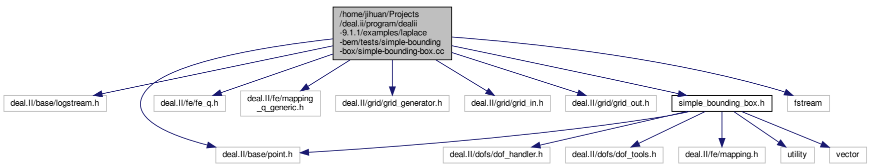
Finally, let's write the RkMatrix in the Octave text data format.

References RkMatrix< Number >::convertToFullMatrix(), RkMatrix< Number >::print_formatted(), print_rkmatrix↩
_to_mat(), and LAPACKFullMatrixExt< Number >::Reshape().

9.89 /home/jihuan/Projects/deal.ii/program/dealii-9.1.1/examples/laplace-bem/tests/simple-bounding-box/simple-bounding-box.cc File Reference

```
#include <deal.II/base/logstream.h>
#include <deal.II/base/point.h>
#include <deal.II/fe/fe_q.h>
#include <deal.II/fe/mapping_q_generic.h>
#include <deal.II/grid/grid_generator.h>
#include <deal.II/grid/grid_in.h>
#include <deal.II/grid/grid_out.h>
#include <simple_bounding_box.h>
#include <fstream>
```

Include dependency graph for simple-bounding-box.cc:



Functions

- template<int dim>
  void **print_bbox_info** (const SimpleBoundingBox< dim > &bbox)
- int main ()

9.89.1 Detailed Description

This file verifies the SimpleBoundingBox class.

9.89.2 Function Documentation

9.89.2.1 main()

```
int main ( )
```

Initialize deal.ii log stream.

Generate the grid for a 3D sphere.

Save the mesh to a file for visualization.

Generate a bounding box for the triangulation using the vertex coordinates of the triangulation.

Print geometric information of the bounding box.

Divide the bounding box into halves.

Print geometric information of the two children boxes.

Create a high order Lagrangian finite element.

Create a DoFHandler, which is associated with the triangulation and distributed with the finite element.

Create a 2nd order mapping, which is required in generating the map from DoF indices to support points.

Create bounding box based on the high order mapping.

Print geometric information of the bounding box.

Create a DoF index set.

Get the vector of support points.

Create a bounding box for the DoF index set.

Print geometric information of the bounding box.