Adaptive Mesh Refinement

Generated by Doxygen 1.8.13

Contents

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 Namespace Documentation

3.1 FEDD Namespace Reference

Data Structures

- class AdaptiveMeshRefinement
- class ErrorEstimation
- class ExporterParaViewAMR
- class RefinementFactory

3.1.1 Detailed Description

RefinementFactory.

ExporterParaView.

ErrorEstimation.

Declaration of Adaptive Mesh Refinement

Author

Lea Saßmannshausen

Declaration of ErrorEstimation

Author

Lea Saßmannshausen

Version

1.0

Copyright

CH

Declaration of ExporterParaViewAMR

Author

Lea Saßmannshausen

Declaration of RefinementFactory

Author

Lea Saßmannshausen

Version

1.0

Copyright

CH

4 Data Structure Documentation

4.1 AdaptiveMeshRefinement< SC, LO, GO, NO > Class Template Reference

Public Types

- typedef Mesh< SC, LO, GO, NO > **Mesh_Type**
- typedef MeshUnstructured< SC, LO, GO, NO > **MeshUnstr_Type**
- typedef Teuchos::RCP< MeshUnstructured< SC, LO, GO, NO > > **MeshUnstrPtr_Type**
- typedef std::vector< MeshUnstrPtr_Type > **MeshUnstrPtrArray_Type**
- typedef MeshUnstructuredRefinement< SC, LO, GO, NO > **MeshUnstrRef_Type**
- typedef Teuchos::RCP< MeshUnstrRef_Type > **MeshUnstrRefPtr_Type**
- typedef std::vector< MeshUnstrRefPtr_Type > **MeshUnstrRefPtrArray_Type**
- typedef Mesh_Type::CommPtr_Type **CommPtr_Type**
- typedef Mesh_Type::CommConstPtr_Type **CommConstPtr_Type**
- typedef Elements **Elements_Type**
- typedef Teuchos::RCP< Elements_Type > **ElementsPtr_Type**
- typedef SurfaceElements **SurfaceElements_Type**
- typedef Teuchos::RCP< SurfaceElements_Type > **SurfaceElementsPtr_Type**
- typedef EdgeElements **EdgeElements_Type**
- typedef Teuchos::RCP< EdgeElements_Type > **EdgeElementsPtr_Type**
- typedef MeshInterface< SC, LO, GO, NO > **MeshInterface_Type**
- typedef Teuchos::RCP< MeshInterface_Type > **MeshInterfacePtr_Type**
- typedef Map< LO, GO, NO > **Map_Type**
- typedef Map_Type::MapPtr_Type **MapPtr_Type**
- typedef Map_Type::MapConstPtr_Type **MapConstPtr_Type**
- typedef MultiVector< SC, LO, GO, NO > **MultiVector_Type**
- typedef Teuchos::RCP< MultiVector_Type > **MultiVectorPtr_Type**
- typedef MultiVector< LO, LO, GO, NO > **MultiVectorLO_Type**
- typedef Teuchos::RCP< MultiVectorLO_Type > **MultiVectorLOPtr_Type**
- typedef MultiVector< GO, LO, GO, NO > **MultiVectorGO_Type**
- typedef Teuchos::RCP< MultiVectorGO_Type > **MultiVectorGOPtr_Type**
- typedef Teuchos::RCP< const MultiVector_Type > **MultiVectorConstPtr_Type**
- typedef Teuchos::OrdinalTraits< LO > **OTLO**
- typedef Matrix< SC, LO, GO, NO > **Matrix_Type**
- typedef Teuchos::RCP< Matrix_Type > **MatrixPtr_Type**
- typedef ExporterParaViewAMR< SC, LO, GO, NO > **Exporter_Type**
- typedef Teuchos::RCP< Exporter_Type > **ExporterPtr_Type**
- typedef Teuchos::RCP< ExporterTxt > **ExporterTxtPtr_Type**
- typedef Problem< SC, LO, GO, NO > **Problem_Type**
- typedef Teuchos::RCP< Problem_Type > **ProblemPtr_Type**
- typedef Domain< SC, LO, GO, NO > **Domain_Type**
- typedef Teuchos::RCP< Domain_Type > **DomainPtr_Type**
- typedef std::vector< DomainPtr_Type > **DomainPtrArray_Type**
- typedef std::vector< MultiVectorPtr_Type > **MultiVectorPtrArray_Type**
- typedef BlockMultiVector< SC, LO, GO, NO > **BlockMultiVector_Type**
- typedef Teuchos::RCP< BlockMultiVector_Type > **BlockMultiVectorPtr_Type**
- typedef Teuchos::RCP< const BlockMultiVector_Type > **BlockMultiVectorConstPtr_Type**

Public Member Functions

- AdaptiveMeshRefinement (string problemType, ParameterListPtr_Type parameterListAll)
- AdaptiveMeshRefinement (string problemType, ParameterListPtr_Type parameterListAll, Func_Type exact↩ SolFunc)
- DomainPtr_Type globalAlgorithm (DomainPtr_Type domainP1, DomainPtr_Type domainP12, BlockMulti↩ VectorConstPtr_Type solution, ProblemPtr_Type problem, RhsFunc_Type rhsFunc)
- DomainPtr_Type **refineArea** (DomainPtr_Type domainP1, vec2D_dbl_Type area, int level)
- MultiVectorConstPtr_Type calcExactSolution ()
- void **identifyProblem** (BlockMultiVectorConstPtr_Type valuesSolution)
- void calcErrorNorms (MultiVectorConstPtr_Type exactSolution, MultiVectorConstPtr_Type solutionP12)
- void initExporter (ParameterListPtr_Type parameterListAll)
- void exportSolution (MeshUnstrPtr_Type mesh, MultiVectorConstPtr_Type exportSolutionMv, MultiVector↩ ConstPtr_Type errorValues, MultiVectorConstPtr_Type exactSolutionMv)
- void exportError (MeshUnstrPtr_Type mesh, MultiVectorConstPtr_Type errorElConst, MultiVectorConstPtr↩ _Type vecDecompositionConst)
- void writeRefinementInfo ()
- void **buildSurfaceTriangleElements** (ElementsPtr_Type elements, EdgeElementsPtr_Type edgeElements, SurfaceElementsPtr_Type surfaceTriangleElements)
- vec_bool_Type **checkInterfaceSurface** (EdgeElementsPtr_Type edgeElements, vec_int_Type originFlag, vec_int_Type edgeNumbers, int indexElement)

Private Attributes

- RhsFunc_Type **rhsFunc_**
- Func_Type **exactSolFunc_**
- MeshUnstrPtr_Type **inputMeshP1_**
- MeshUnstrPtr_Type **inputMeshP12_**
- MeshUnstrPtr_Type **outputMesh_**
- MultiVectorPtrArray_Type **errorEstimationMv_**
- MultiVectorPtr_Type **errorElementsMv_**
- MultiVectorConstPtr_Type **errorNodesMv_**
- BlockMultiVectorConstPtr_Type **solution_**
- CommConstPtr_Type **comm_**
- bool **exportWithParaview_** = true
- bool **initExporter_** =false
- ExporterPtr_Type **exporterSol_**
- ExporterPtr_Type **exporterError_**
- DomainPtrArray_Type **domainsP1_**
- DomainPtrArray_Type **domainsP12_**
- DomainPtr_Type **domainP1_**
- DomainPtr_Type **domainP12_**
- ProblemPtr_Type **problem_**
- string **refinementRestriction_** = "keepRegularity"
- string **markingStrategy_** = "Maximum"
- double **theta_** = 0.5
- double **tol_** = 0.001
- bool **meshQualityPrint_** = "false"
- bool **timeTablePrint_** = "false"
- int **refinement3DDiagonal_** = 0
- string **problemType_**
- int **dim_**
- int **currentIter_**
- int **maxIter_** = 5

- int **maxRank_**
- string **FEType1_**
- string **FEType2_**
- vec_dbl_Type **maxErrorEl**
- vec_dbl_Type **maxErrorKn**
- vec_int_Type **numElements**
- vec_int_Type **numElementsProc**
- vec_dbl_Type **relError**
- vec_dbl_Type **eRelError**
- vec_dbl_Type **errorH1**
- vec_dbl_Type **errorL2**
- vec_int_Type **numNodes**
- bool **writeRefinementTime_** = true
- bool **writeMeshQuality_** = true
- ParameterListPtr_Type **parameterListAll_**
- int **dofs_**
- int **dofsP_**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 AdaptiveMeshRefinement() [1/2]

```
AdaptiveMeshRefinement (
            string problemType,
            ParameterListPtr_Type parameterListAll )
```

Initializing problem with the kind of problem (e.g. Laplace, Stokes) for determining the correct error estimation and the dimension

Parameters

in	*problemType,dim*	

4.1.1.2 AdaptiveMeshRefinement() [2/2]

```
AdaptiveMeshRefinement (
            string problemType,
            ParameterListPtr_Type parameterListAll,
            Func_Type exactSolFunc )
```

Initializing problem with the kind of problem, dimension and refinement spectific parameters

4.1.2 Member Function Documentation

4.1.2.1 calcErrorNorms()

```
void calcErrorNorms (
            MultiVectorConstPtr_Type exactSolution,
            MultiVectorConstPtr_Type solutionP12 )
```

Calculating error norms. If the exact solution is unknown we use approxmated errorNorm and error indicators

Parameters

in	*exact*	solution if known
in	*FE*	solution
in	*error*	estimation

4.1.2.2 exportError()

```
void exportError (
            MeshUnstrPtr_Type mesh,
            MultiVectorConstPtr_Type errorElConst,
            MultiVectorConstPtr_Type vecDecompositionConst )
```

ParaView exporter export of error values and element distribution on current mesh

4.1.2.3 exportSolution()

```
void exportSolution (
            MeshUnstrPtr_Type mesh,
            MultiVectorConstPtr_Type exportSolutionMv,
            MultiVectorConstPtr_Type errorValues,
            MultiVectorConstPtr_Type exactSolutionMv )
```

ParaView exporter export of solution on current mesh

4.1.2.4 globalAlgorithm()

```
AdaptiveMeshRefinement< SC, LO, GO, NO >::DomainPtr_Type globalAlgorithm (
            DomainPtr_Type domainP1,
            DomainPtr_Type domainP12,
            BlockMultiVectorConstPtr_Type solution,
            ProblemPtr_Type problem,
            RhsFunc_Type rhsFunc )
```

Global Algorithm of Mesh Refinement.

Given domains and solutions depending on problem global mesh refinement algorithm and error estimation is performed

i.e. if to solve simple laplace problem, we have only one solution to put in, if to estimate error for Navier-Stokes equation we need pressure and velocity solution

Parameters

in	*domainP1*	domain with P1 discretization, always neccesary as refinement is performed on P1 Mesh
in	*domainP12*	domain with P1 or P2 discretization if available, otherwise input domainP1
in	*solution1*	solution of problem on P1 or P2 discretization
in	*solution2*	solution of problem on P1 or P2 discretization if available, otherwise input solutionP1

4.1.2.5 initExporter()

```
void initExporter (
            ParameterListPtr_Type parameterListAll )
```

ParaView exporter setup

4.1.2.6 writeRefinementInfo()

```
void writeRefinementInfo ( )
```

Writing refinement information

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

- AdaptiveMeshRefinement_decl.hpp
- AdaptiveMeshRefinement_def.hpp

4.2 ErrorEstimation< SC, LO, GO, NO > Class Template Reference

Public Types

- typedef Mesh< SC, LO, GO, NO > **Mesh_Type**
- typedef Teuchos::RCP< MeshUnstructured< SC, LO, GO, NO > > **MeshUnstrPtr_Type**
- typedef std::vector< MeshUnstrPtr_Type > **MeshUnstrPtrArray_Type**
- typedef Mesh_Type::CommPtr_Type **CommPtr_Type**
- typedef Mesh_Type::CommConstPtr_Type **CommConstPtr_Type**
- typedef Mesh_Type::Elements_Type **Elements_Type**
- typedef Mesh_Type::ElementsPtr_Type **ElementsPtr_Type**
- typedef EdgeElements **EdgeElements_Type**
- typedef Teuchos::RCP< EdgeElements_Type > **EdgeElementsPtr_Type**
- typedef SurfaceElements **SurfaceElements_Type**
- typedef Teuchos::RCP< SurfaceElements_Type > **SurfaceElementsPtr_Type**
- typedef MeshInterface< SC, LO, GO, NO > **MeshInterface_Type**
- typedef Teuchos::RCP< MeshInterface_Type > **MeshInterfacePtr_Type**
- typedef Map< LO, GO, NO > **Map_Type**
- typedef Map_Type::MapPtr_Type **MapPtr_Type**
- typedef Map_Type::MapConstPtr_Type **MapConstPtr_Type**
- typedef MultiVector< SC, LO, GO, NO > **MultiVector_Type**
- typedef Teuchos::RCP< MultiVector_Type > **MultiVectorPtr_Type**
- typedef MultiVector< LO, LO, GO, NO > **MultiVectorLO_Type**
- typedef Teuchos::RCP< MultiVectorLO_Type > **MultiVectorLOPtr_Type**
- typedef MultiVector< GO, LO, GO, NO > **MultiVectorGO_Type**
- typedef Teuchos::RCP< MultiVectorGO_Type > **MultiVectorGOPtr_Type**
- typedef Teuchos::RCP< const MultiVector_Type > **MultiVectorConstPtr_Type**
- typedef Teuchos::OrdinalTraits< LO > **OTLO**
- typedef Matrix< SC, LO, GO, NO > **Matrix_Type**
- typedef Teuchos::RCP< Matrix_Type > **MatrixPtr_Type**
- typedef BlockMultiVector< SC, LO, GO, NO > **BlockMultiVector_Type**
- typedef Teuchos::RCP< BlockMultiVector_Type > **BlockMultiVectorPtr_Type**
- typedef Teuchos::RCP< const BlockMultiVector_Type > **BlockMultiVectorConstPtr_Type**

Public Member Functions

- **ErrorEstimation** (int dim, string problemType)
- MultiVectorPtr_Type estimateError (MeshUnstrPtr_Type inputMeshP12, MeshUnstrPtr_Type inputMeshP1, BlockMultiVectorConstPtr_Type valuesSolution, RhsFunc_Type rhsFunc, string FEType)
- void identifyProblem (BlockMultiVectorConstPtr_Type valuesSolution)
- void makeRepeatedSolution (BlockMultiVectorConstPtr_Type valuesSolution)
- vec3D_dbl_Type calcNPhi (string phiDerivative, int dofsSol, string FEType)
- vec_dbl_Type calculateJump ()
- vec2D_dbl_Type **gradPhi** (int dim, int intFE, vec_dbl_Type &p)
- vec_dbl_Type **phi** (int dim, int intFE, vec_dbl_Type &p)
- vec_dbl_Type **divPhi** (int dim, int intFE, vec_dbl_Type &p)
- MultiVectorPtr_Type determineCoarseningError (MeshUnstrPtr_Type mesh_k, MeshUnstrPtr_Type mesh_↩ k_m, MultiVectorPtr_Type errorElementMv_k, string distribution, string markingStrategy, double theta)
- double determineResElement (FiniteElement element, RhsFunc_Type rhsFunc)
- double determineDivU (FiniteElement element)
- vec2D_dbl_Type getQuadValues (int dim, string FEType, string Type, vec_dbl_Type &QuadW, FiniteElement surface)
- void markElements (MultiVectorPtr_Type errorElementMv, double theta, string strategy, MeshUnstrPtr_Type meshUnstr)
- vec_dbl_Type determineH_T_min (ElementsPtr_Type elements, EdgeElementsPtr_Type edgeElements, vec2D_dbl_ptr_Type points, vec_dbl_Type &volTetraeder)
- vec_dbl_Type calcDiamElements (ElementsPtr_Type elements, vec2D_dbl_ptr_Type points)
- vec_dbl_Type determineAreaTriangles (ElementsPtr_Type elements, EdgeElementsPtr_Type edgeElements, SurfaceElementsPtr_Type surfaceElements, vec2D_dbl_ptr_Type points)
- void **buildTriangleMap** ()
- void updateElementsOfSurfaceLocalAndGlobal (EdgeElementsPtr_Type edgeElements, SurfaceElements↩ Ptr_Type surfaceTriangleElements)
- void **setErrorEstimate** (MultiVectorPtr_Type errorElements)
- MultiVectorPtr_Type **getErrorEstimate** ()
- void tagArea (MeshUnstrPtr_Type meshUnstr, vec2D_dbl_Type area)

Data Fields

- string **refinementRestriction_** = "none"
- string **markingStrategy_** = "Maximum"
- double **theta_** = 0.5
- bool **meshQualityPrint_** = "false"
- bool **timeTablePrint_** = "false"
- int **refinement3DDiagonal_** = 0
- int **dim_**
- string **problemType_**

Protected Attributes

- vec_GO_Type **globalInterfaceIDs_**
- MultiVectorPtr_Type **errorEstimation_**
- vec_dbl_Type **areaTriangles_**
- vec_dbl_Type **volTetraeders_**
- vec_dbl_Type **h_T_diam_E_**
- vec_dbl_Type **h_T_min_**
- MapConstPtr_Type **surfaceTriangleMap_**
- SurfaceElementsPtr_Type **surfaceElements_**
- int **dofs_**
- int **dofsP_**
- bool **calculatePressure_** = false
- BlockMultiVectorConstPtr_Type **valuesSolutionRepVel_**
- BlockMultiVectorConstPtr_Type **valuesSolutionRepPre_**

Private Attributes

- MeshUnstrPtr_Type **inputMesh_**
- MeshUnstrPtr_Type **inputMeshP1_**
- string **FEType1_**
- string **FEType2_**

4.2.1 Member Function Documentation

4.2.1.1 calcDiamElements()

```
vec_dbl_Type calcDiamElements (
            ElementsPtr_Type elements,
            vec2D_dbl_ptr_Type points )
```

Calculating the diameter of elements. This is necessary for 2D A-posteriori error estimation.

Parameters

in	*elements*	
in	*points*	

4.2.1.2 calcNPhi()

```
vec3D_dbl_Type calcNPhi (
            string phiDerivative,
            int dofsSol,
            string FEType )
```

Function that calculates the jump part for nabla u or p.

Parameters

in	*phiDerivative*	is either 'Gradient' or 'None' and what kind of jump is calculated depends on the problemType we have at hand. If phiDerivative is 'Gradient' the nabla u jump part is caluculated and if its 'None' then the p-
in	*dofsSol*	is the degree of freedom of the caluclated jump part. p's dof is typically 1 whereas u's dof can vary depending on the problem
in	*FEType*	of the calculated jump part.

4.2.1.3 calculateJump()

```
vec_dbl_Type calculateJump ( )
```

Part of the error estimator that calculates the jump part of the estimation.

Parameters

in	*none,as*	all necessary parameters for the calculation are already part of the Error estimation class.

What kind of jump is calculated depends on the problemType we have at hand.

4.2.1.4 determineAreaTriangles()

```
vec_dbl_Type determineAreaTriangles (
             ElementsPtr_Type elements,
             EdgeElementsPtr_Type edgeElements,
             SurfaceElementsPtr_Type surfaceElements,
             vec2D_dbl_ptr_Type points )
```

Calculating the area of the triangle elements of tetrahedra.

Parameters

in	*elements*	
in	*edgeElements*	
in	*surfaceElements*	
in	*points*	

4.2.1.5 determineCoarseningError()

```
ErrorEstimation< SC, LO, GO, NO >::MultiVectorPtr_Type determineCoarseningError (
             MeshUnstrPtr_Type mesh_k,
             MeshUnstrPtr_Type mesh_k_m,
             MultiVectorPtr_Type errorElementMv_k,
             string distribution,
             string markingStrategy,
             double theta )
```

determineCoarseningError is the essential part of the mesh coarsening process.

instead of calulating a error of mesh level k, we redestribute it to lower mesh levels and defining those.// We execute this function with an estimated error from the above 'estimateCoarseningError' function. With this error, we mark the elements according to that error and refine afterwards If we decide to coarsen a certain mesh level, we take that level, look at the k-m level and refine that to the point where we are at the same level we wanted to perform the coarsening on

Parameters

in	*mesh_k*	the current mesh of level k
in	*mesh_k_m*	the mesh of refinement level k-m
in	*errorElementMv↩_k*	as the error estimation of mesh level k
in	*distribution*	is either 'forwards' or 'backwards'. We determine the error estimate in level k-m with redistributing backwards. if we are in level k-m we calculate the k-m+1 mesh level error estimation via redistributing the k-m error forward.
in	*markingStrategy*	the strategy with which element are marked
in	*theta*	as the a marking threshold

4.2.1.6 determineDivU()

```
double determineDivU (
            FiniteElement element )
```

Function that that determines || div(u) ||_T for a Element T.

Parameters

in	*FiniteElement*	element where ||div(u)||_T is calculated on

4.2.1.7 determineH_T_min()

```
vec_dbl_Type determineH_T_min (
            ElementsPtr_Type elements,
            EdgeElementsPtr_Type edgeElements,
            vec2D_dbl_ptr_Type points,
            vec_dbl_Type & volTetraeder )
```

function, that determines h_T as the shortest vector inside a tetraeder as propose in...

Parameters

in	*elements*	
in	*edgeelements*	
in	*points*	
in	*volTetraeder*	is calulated along the way and also usefull at another part of 3D jump calculation

4.2.1.8 determineResElement()

```
double determineResElement (
            FiniteElement element,
            RhsFunc_Type rhsFunc )
```

Function that that determines || u_h + f ||_(L2(T)) or || u_h + f - p ||_T for a Element T.

Parameters

in	*FiniteElement*	element where ||div(u)||_T is calculated on
in	*RhsFunc_Type*	rhsFunc which is the function used for the rhs of the pde

4.2.1.9 estimateError()

```
ErrorEstimation< SC, LO, GO, NO >::MultiVectorPtr_Type estimateError (
             MeshUnstrPtr_Type inputMeshP12,
             MeshUnstrPtr_Type inputMeshP1,
             BlockMultiVectorConstPtr_Type valuesSolution,
             RhsFunc_Type rhsFunc,
             string FETypeV )
```

Main Function for A-posteriori Error Estimation.

depending on the problem the the error estimation is calculated accordingly

Parameters

in	*inputMeshP1*	the P1 Mesh that is used for later refinement
in	*inputMeshP12*	the possible P2 Mesh, if one of the solutions is of P2 Discretisation, otherwise both meshes are P1
in	*solution*	of the PDE in BlockMultiVector Format (Block 0: Velocity, Block 1: Pressure)
in	*rhs*	Function
in	*FETypeV*	as the maximum FEType for the Velocity, pressure is assumed to be P1

4.2.1.10 getQuadValues()

```
vec2D_dbl_Type getQuadValues (
             int dim,
             string FEType,
             string Type,
             vec_dbl_Type & QuadW,
             FiniteElement surface )
```

Returns neccesary quadrature Values. Is distinguishes between needing Element or Surface information.

Parameters

in	*dim*	for which the quadrature points are needed
in	*FEType*	for which the quadrature points are needed
in	*Type*	of quadrature points are need. Either 'Element' if you integrate over an element or 'Surface' if you need to integrate over a surface (i.e. for calculating the jump)
in	*QuadW*	Vector to be filled with the quadrature weights accordingly
in	*FiniteElement*	surface for which you need the quadrature points in case if 'Surface' type, as it is needed for figuring out the quadrature points

4.2.1.11 markElements()

```
void markElements (
             MultiVectorPtr_Type errorElementMv,
             double theta,
             string strategy,
             MeshUnstrPtr_Type meshP1 )
```

Function that marks the elements for refinement.

Parameters

in	*errorElementMv*	is the MultiVector that contains the estimated error for each element
in	*theta*	is a parameter determining a certain error bound for marking
in	*markingStrategy*	is the strategy with which the elements are marked. Implemented Strategies 'Doerfler' or 'Maximum'
in	*meshP1*	is the P1 mesh which is used for later refinement and has to be the one beeing marked

!! it is essential that the meshP1 mesh inserted here is the mesh that will be used for mesh refinement, as it contains the elementwise-information determining refinement. !!

4.2.1.12 tagArea()

```
void tagArea (
            MeshUnstrPtr_Type inputMeshP1,
            vec2D_dbl_Type area )
```

Tags only a certain Area for refinement and is independent of any error estimation.

Parameters

in	*inputMeshP1*	the P1 Mesh that is used for later refinement
in	*the*	area, that is suppose to be refined. If is a vector defining the area as follows: row1:[x_0,x_1] x-limits, row2: [y_0,y_1] y-limits, row3: [z_0,z_1] z-limits

4.2.1.13 updateElementsOfSurfaceLocalAndGlobal()

```
void updateElementsOfSurfaceLocalAndGlobal (
            EdgeElementsPtr_Type edgeElements,
            SurfaceElementsPtr_Type surfaceTriangleElements )
```

updateElementsOfSurfaceLocalAndGlobal is performed here instead of in meshRefinement, as the information is only needed in case of error estimation

Parameters

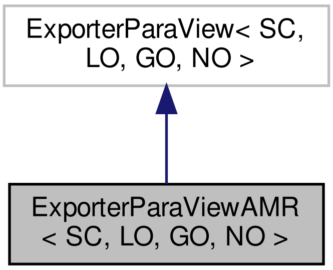
in	*edgeElements*	
in	*surfaceTriangleElements*	

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

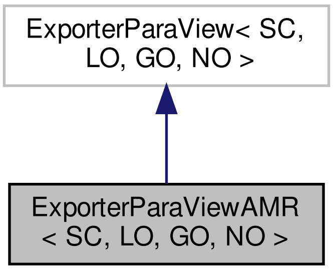
- ErrorEstimation_decl.hpp
- ErrorEstimation_def.hpp

4.3 ExporterParaViewAMR< SC, LO, GO, NO > Class Template Reference

Inheritance diagram for ExporterParaViewAMR< SC, LO, GO, NO >:


ExporterParaView< SC, LO, GO, NO >
ExporterParaViewAMR < SC, LO, GO, NO >

Collaboration diagram for ExporterParaViewAMR< SC, LO, GO, NO >:


ExporterParaView< SC, LO, GO, NO >
ExporterParaViewAMR < SC, LO, GO, NO >

Public Types

- typedef std::vector< double > **vec_dbl**
- typedef std::vector< std::vector< double > > **vec2D_dbl**
- typedef std::vector< std::vector< int > > **vec2D_int**
- typedef std::vector< std::vector< long long > > **vec2D_longlong**
- typedef Teuchos::RCP< std::vector< int > > **vec_int_ptr**
- typedef Teuchos::RCP< std::vector< long long > > **vec_longlong_ptr**
- typedef Teuchos::RCP< vec_dbl > **vec_dbl_ptr**
- typedef Teuchos::RCP< std::vector< std::vector< double > > > **vec2D_dbl_ptr**
- typedef Teuchos::RCP< std::vector< std::vector< int > > > **vec2D_int_ptr**
- typedef Teuchos::RCP< vec2D_longlong > **vec2D_longlong_ptr**
- typedef Teuchos::RCP< Epetra_Vector > **EpetraVec_ptr**
- typedef Teuchos::RCP< Epetra_MpiComm > **EpetraComm_ptr**
- typedef Teuchos::RCP< Epetra_IntVector > **EpetraVecInt_ptr**

- typedef Teuchos::RCP< Epetra_LongLongVector > **EpetraVecLongLong_ptr**
- typedef Teuchos::RCP< Epetra_MultiVector > **EpetraMVPtr_Type**
- typedef Teuchos::RCP< Epetra_Map > **EpetraMapPtr_Type**
- typedef EpetraExt::HDF5 **HDF5_Type**
- typedef Teuchos::RCP< HDF5_Type > **HDF5Ptr_Type**
- typedef Teuchos::Comm< int > **Comm_Type**
- typedef Teuchos::RCP< const Comm_Type > **CommConstPtr_Type**
- typedef const Teuchos::RCP< const Comm_Type > **CommConstPtrConst_Type**
- typedef Map< LO, GO, NO > **Map_Type**
- typedef Teuchos::RCP< const Map_Type > **MapConstPtr_Type**
- typedef const MapConstPtr_Type **MapConstPtrConst_Type**
- typedef MultiVector< SC, LO, GO, NO > **MultiVector_Type**
- typedef Teuchos::RCP< const MultiVector_Type > **MultiVectorConstPtr_Type**
- typedef const MultiVectorConstPtr_Type **MultiVectorConstPtrConst_Type**
- typedef Mesh< SC, LO, GO, NO > **Mesh_Type**
- typedef Teuchos::RCP< Mesh_Type > **MeshPtr_Type**
- typedef Mesh_Type::ElementsPtr_Type **ElementsPtr_Type**

Public Member Functions

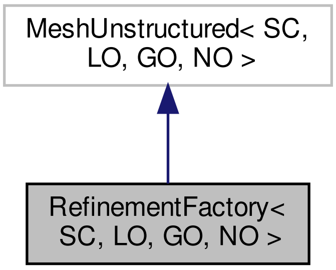
- void **updateVariables** (MultiVectorConstPtr_Type &u, std::string varName)
- void **reSetup** (MeshPtr_Type mesh)

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

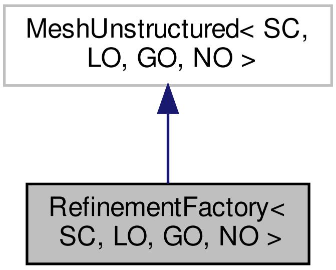
- ExporterParaViewAMR_decl.hpp
- ExporterParaViewAMR_def.hpp

4.4 RefinementFactory< SC, LO, GO, NO > Class Template Reference

Inheritance diagram for RefinementFactory< SC, LO, GO, NO >:


MeshUnstructured< SC, LO, GO, NO >
RefinementFactory< SC, LO, GO, NO >

Collaboration diagram for RefinementFactory< SC, LO, GO, NO >:


MeshUnstructured< SC, LO, GO, NO >
RefinementFactory< SC, LO, GO, NO >

Public Types

- typedef Mesh< SC, LO, GO, NO > **Mesh_Type**
- typedef MeshUnstructured< SC, LO, GO, NO > **MeshUnstr_Type**
- typedef Teuchos::RCP< MeshUnstructured< SC, LO, GO, NO > > **MeshUnstrPtr_Type**
- typedef std::vector< MeshUnstrPtr_Type > **MeshUnstrPtrArray_Type**
- typedef Mesh_Type::CommPtr_Type **CommPtr_Type**
- typedef Mesh_Type::CommConstPtr_Type **CommConstPtr_Type**
- typedef Elements **Elements_Type**
- typedef Teuchos::RCP< Elements_Type > **ElementsPtr_Type**
- typedef SurfaceElements **SurfaceElements_Type**
- typedef Teuchos::RCP< SurfaceElements_Type > **SurfaceElementsPtr_Type**
- typedef EdgeElements **EdgeElements_Type**
- typedef Teuchos::RCP< EdgeElements_Type > **EdgeElementsPtr_Type**
- typedef MeshInterface< SC, LO, GO, NO > **MeshInterface_Type**
- typedef Teuchos::RCP< MeshInterface_Type > **MeshInterfacePtr_Type**
- typedef Map< LO, GO, NO > **Map_Type**
- typedef Map_Type::MapPtr_Type **MapPtr_Type**
- typedef Map_Type::MapConstPtr_Type **MapConstPtr_Type**
- typedef MultiVector< SC, LO, GO, NO > **MultiVector_Type**
- typedef Teuchos::RCP< MultiVector_Type > **MultiVectorPtr_Type**
- typedef MultiVector< LO, LO, GO, NO > **MultiVectorLO_Type**
- typedef Teuchos::RCP< MultiVectorLO_Type > **MultiVectorLOPtr_Type**
- typedef MultiVector< GO, LO, GO, NO > **MultiVectorGO_Type**
- typedef Teuchos::RCP< MultiVectorGO_Type > **MultiVectorGOPtr_Type**
- typedef Teuchos::RCP< const MultiVector_Type > **MultiVectorPtrConst_Type**
- typedef Teuchos::OrdinalTraits< LO > **OTLO**
- typedef Matrix< SC, LO, GO, NO > **Matrix_Type**
- typedef Teuchos::RCP< Matrix_Type > **MatrixPtr_Type**

Public Member Functions

- RefinementFactory (CommConstPtr_Type comm, int volumeID=10)
- RefinementFactory (CommConstPtr_Type comm, int volumeID, string refinementRestriction, int refinement3↵ DDiagonal=0)
- void refineMesh (MeshUnstrPtr_Type meshP1, int iteration, MeshUnstrPtr_Type outputMesh)
- void assignEdgeFlags (MeshUnstrPtr_Type meshP1, EdgeElementsPtr_Type edgeElements)
- void refineRegular (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int i)
- void refineGreen (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int i)
- void refineBlue (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int i)
- void refineRed (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int i)
- void refineType1 (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int indexElement)
- void refineType2 (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int indexElement)
- void refineType3 (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int indexElement)
- void refineType4 (EdgeElementsPtr_Type edgeElements, ElementsPtr_Type elements, int indexElement)
- void addMidpoint (EdgeElementsPtr_Type edgeElements, int i)
- int determineLongestEdge (EdgeElementsPtr_Type edgeElements, vec_int_Type edgeVec, vec2D_dbl_ptr↵ _Type points)
- void buildEdgeMap (MapConstPtr_Type mapGlobalProc, MapConstPtr_Type mapProc)
- void buildNodeMap (EdgeElementsPtr_Type edgeElements, MapConstPtr_Type mapGlobalProc, Map↵ ConstPtr_Type mapProc, int newPoints, int newPointsRepeated)
- void updateElementsOfEdgesLocalAndGlobal (int maxRank, MapConstPtr_Type edgeMap)
- void **updateElementsOfSurfaceLocalAndGlobal** (EdgeElementsPtr_Type edgeElements)
- vec_bool_Type **checkInterfaceSurface** (EdgeElementsPtr_Type edgeElements, vec_int_Type originFlag, vec_int_Type edgeNumbers, int indexElement)
- void refinementRestrictions (MeshUnstrPtr_Type meshP1, ElementsPtr_Type elements, EdgeElementsPtr↵ _Type edgeElements, int iteration, int &newPoints, int &newPointsCommon, vec_GO_Type &globalInterface↵ IDsTagged, MapConstPtr_Type mapInterfaceEdges, string restriction, int &newElements)
- void refineIrregular (ElementsPtr_Type elements, EdgeElementsPtr_Type edgeElements, int &newElements, MapConstPtr_Type edgeMap, SurfaceElementsPtr_Type surfaceTriangleElements)
- void buildSurfaceTriangleElements (ElementsPtr_Type elements, EdgeElementsPtr_Type edgeElements, SurfaceElementsPtr_Type surfaceTriangleElements, MapConstPtr_Type edgeMap, MapConstPtr_Type elementMap)
- void **setErrorEstimate** (vec_dbl_Type errorElements)
- vec_dbl_Type **getErrorEstimate** ()

Data Fields

- string **refinementRestriction_** = "none"
- string **markingStrategy_** = "Maximum"
- double **theta_** = 0.5
- bool **meshQualityPrint_** = "false"
- bool **timeTablePrint_** = "false"
- int **refinement3DDiagonal_** = 0

Protected Attributes

- vec_GO_Type **globalInterfaceIDs_**
- vec_dbl_Type **errorEstimation_**
- vec_dbl_Type **areaTriangles_**
- vec_dbl_Type **volTetraeders_**
- vec_dbl_Type **h_T_diam_E_**
- vec_dbl_Type **h_T_min_**
- MapConstPtr_Type **surfaceTriangleMap_**

4.4.1 Constructor & Destructor Documentation

4.4.1.1 RefinementFactory() [1/2]

```
RefinementFactory (
              CommConstPtr_Type comm,
              int volumeID = 10 )
```

Parameters

in	*comm*	
in	*volumeID*	

4.4.1.2 RefinementFactory() [2/2]

```
RefinementFactory (
              CommConstPtr_Type comm,
              int volumeID,
              string refinementRestriction,
              int refinement3DDiagonal = 0 )
```

Parameters

in	*comm*	
in	*volumeID*	
in	*refinementRestriction*	for repeated refinement steps
in	*refinement3DDiagonal*	

4.4.2 Member Function Documentation

4.4.2.1 addMidpoint()

```
void addMidpoint (
              EdgeElementsPtr_Type edgeElements,
              int edgeID )
```

adding a Midpoint on an edge

Parameters

in	*edgeElements*	
in	*edgeID*	

4.4.2.2 assignEdgeFlags()

```
void assignEdgeFlags (
            MeshUnstrPtr_Type meshP1,
            EdgeElementsPtr_Type edgeElements )
```

Not all edges are marked with a flag in the beginning. In order to set the correct flags to new points we assign the edge flag of the edge they originated from, similar to the function determineEdgeFlagP2New, but uses the edgeMap.

Parameters

in	*meshP1*	inputMesh
in	*edgeElements*	that receive flags

4.4.2.3 buildEdgeMap()

```
void buildEdgeMap (
            MapConstPtr_Type mapGlobalProc,
            MapConstPtr_Type mapProc )
```

building edgeMap after refinement

Parameters

in	*mapGlobalProc*	
in	*mapProc*	

4.4.2.4 buildNodeMap()

```
void buildNodeMap (
            EdgeElementsPtr_Type edgeElements,
            MapConstPtr_Type mapGlobalProc,
            MapConstPtr_Type mapProc,
            int newPoints,
            int newPointsRepeated )
```

building nodemap after refinement

Parameters

in	*edgeElements*	
in	*mapGlobalProc*	
in	*mapProc*	
in	*newPoints*	
in	*newPointsRepeated*	

4.4.2.5 buildSurfaceTriangleElements()

```
void buildSurfaceTriangleElements (
            ElementsPtr_Type elements,
            EdgeElementsPtr_Type edgeElements,
            SurfaceElementsPtr_Type surfaceTriangleElements,
            MapConstPtr_Type edgeMap,
            MapConstPtr_Type elementMap )
```

building surface triangle elements, as they are not originally part of the mesh information provided by mesh partitioner

Parameters

in	*elements*	
in	*edgeElements*	
in	*surfaceTriangleElements*	pointer which will be filled with surfaceTriangleElements
in	*edgeMap*	
in	*elementMap*	

4.4.2.6 determineLongestEdge()

```
int determineLongestEdge (
            EdgeElementsPtr_Type edgeElements,
            vec_int_Type edgeVec,
            vec2D_dbl_ptr_Type points )
```

determine longest edge in triangle

Parameters

in	*edgeElements*	
in	*edgeVec*	
in	*points*	
out	*local*	edgeID of the longest edge

4.4.2.7 refineBlue()

```
void refineBlue (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

2D blue refinement: refining element according to blue refinement scheme - connecting nodes of shorter edge with midpoint of longer tagged edge and connect that with opposite corner

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.8 refineGreen()

```
void refineGreen (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

2D green refinement: refining the element according to green scheme - connecting node on refined edge with the opposite node

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.9 refineIrregular()

```
void refineIrregular (
            ElementsPtr_Type elements,
            EdgeElementsPtr_Type edgeElements,
            int & newElements,
            MapConstPtr_Type edgeMap,
            SurfaceElementsPtr_Type surfaceTriangleElements )
```

irregular refinement performed according to set rules determined by Bey or Verfürth

Parameters

in	*elements*	
in	*edgeElements*	
in	*newElements*	
in	*edgeMap*	
in	*surfaceTriangleElements*	

4.4.2.10 refinementRestrictions()

```
void refinementRestrictions (
            MeshUnstrPtr_Type meshP1,
            ElementsPtr_Type elements,
            EdgeElementsPtr_Type edgeElements,
            int iteration,
            int & newPoints,
            int & newPointsCommon,
            vec_GO_Type & globalInterfaceIDsTagged,
            MapConstPtr_Type mapInterfaceEdges,
            string restriction,
            int & newElements )
```

Refinement Restrictions In 2D we can add some Restrictions to the Mesh Refinement: KeepRegularity: this will keep the regularity of the Mesh by only refining whith a irregular strategy when the longest edge is involved. If not we add a node to the longest edge, whereby the irregular refinement strategy is changed CheckGreenTags: this will only check tagged green Elements, if its irregular refinement tag from the previous refinement is 'green' and if so not refine it green again but add a node to the longest edge and thus refine it blue In the 3D Case we simply never refine an element irregularly twice, this strategy is called simply 'Bey'. If an element is refined regular, its refinement tag changes from eventually 'irregular' to regular. If those elements should still not be refined irregular we use the strategy 'BeyIrreuglar.

Furthermore if there is no fitting irrregular refinement strategy (Type(1)-Type(4) don't fit) we refine regular instead.

Parameters

in	*meshP1*	
in	*elements*	
in	*edgeElements*	
in	*iteration*	
in	*newPoints*	
in	*newPointsCommon*	
in	*globalInterfaceIDsTagged*	
in	*mapInterfaceEdges*	
in	*restriction*	
in	*newElements*	

4.4.2.11 refineMesh()

```
void refineMesh (
            MeshUnstrPtr_Type meshP1,
            int iteration,
            MeshUnstrPtr_Type outputMesh )
```

main function of RefinementFactory, performs one complete mesh refinement

Parameters

in	*meshP1*	inputMesh
in	*iteration*	current Iteration
in	*outputMesh*	refined mesh

4.4.2.12 refineRed()

```
void refineRed (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

2D red refinement: refining the element red by connecting all tagged edges midpoints. one element is refined into 4

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.13 refineRegular()

```
void refineRegular (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

2D and 3D regular refinement. Chosen by error estimator or otherwise elements are refined regular by connecting edge midpoints.

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.14 refineType1()

```
void refineType1 (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

3D Type(1) refinement

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.15 refineType2()

```
void refineType2 (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

3D Type(2) refinement

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.16 refineType3()

```
void refineType3 (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

3D Type(3) refinement

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.17 refineType4()

```
void refineType4 (
            EdgeElementsPtr_Type edgeElements,
            ElementsPtr_Type elements,
            int indexElement )
```

3D Type(4) refinement

Parameters

in	*edgeElements*	
in	*elements*	
in	*indexELement*	

4.4.2.18 updateElementsOfEdgesLocalAndGlobal()

```
void updateElementsOfEdgesLocalAndGlobal (
            int maxRank,
            MapConstPtr_Type edgeMap )
```

Updating ElementsOfEdgesLocal and ElementsOfEdgesGlobal.

Parameters

in	*maxRank*	
in	*edgeMap*	

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

- RefinementFactory_decl.hpp
- RefinementFactory_def.hpp