

My Project

Generated by Doxygen 1.9.5

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	7
3.1 Class List	7
4 File Index	11
4.1 File List	11
5 Namespace Documentation	13
5.1 Algebra Namespace Reference	13
5.1.1 Enumeration Type Documentation	13
5.1.1.1 Solvers	13
5.2 corenc Namespace Reference	14
5.2.1 Detailed Description	14
5.2.2 Typedef Documentation	15
5.2.2.1 scalar_func	15
5.2.2.2 vector_func	15
5.2.3 Enumeration Type Documentation	15
5.2.3.1 Parameters	15
5.2.3.2 Terms	15
5.3 corenc::color Namespace Reference	16
5.3.1 Variable Documentation	16
5.3.1.1 BBLACK	16
5.3.1.2 BBLUE	16
5.3.1.3 BCYAN	16
5.3.1.4 BGREEN	17
5.3.1.5 BLACK	17
5.3.1.6 BLUE	17
5.3.1.7 BMAGENTA	17
5.3.1.8 BRED	17
5.3.1.9 BWHITE	17
5.3.1.10 BYELLOW	17
5.3.1.11 CYAN	17
5.3.1.12 ESCAPE	18
5.3.1.13 GREEN	18
5.3.1.14 MAGENTA	18
5.3.1.15 PURPLE	18
5.3.1.16 RED	18
5.3.1.17 WHITE	18
5.3.1.18 YELLOW	18

5.4 corenc::Mesh Namespace Reference	19
5.4.1 Typedef Documentation	20
5.4.1.1 function_dp	20
5.4.2 Enumeration Type Documentation	20
5.4.2.1 Elements	20
5.4.2.2 Meshes	20
5.4.2.3 NODES	21
5.5 corenc::method Namespace Reference	21
5.5.1 Enumeration Type Documentation	21
5.5.1.1 BoundaryType	21
5.5.1.2 DGFlux	22
5.5.1.3 FVFlux	22
5.6 corenc::solvers Namespace Reference	23
5.7 corenc::tests Namespace Reference	23
5.8 Methods Namespace Reference	23
5.9 wtf Namespace Reference	23
5.9.1 Function Documentation	23
5.9.1.1 center_point()	23
5.9.1.2 mid_point()	24
5.9.1.3 s_point()	24
6 Class Documentation	25
6.1 corenc::CBurgersScalar Class Reference	25
6.1.1 Constructor & Destructor Documentation	25
6.1.1.1 CBurgersScalar()	25
6.1.1.2 ~CBurgersScalar()	26
6.1.2 Member Function Documentation	26
6.1.2.1 addTerm()	26
6.1.2.2 getFlux()	26
6.1.2.3 getNumberOfTerms()	26
6.1.2.4 getTerm()	26
6.1.2.5 load_parameters()	26
6.1.2.6 removeTerm()	27
6.1.2.7 setTerm()	27
6.2 corenc::Mesh::CCube Class Reference	27
6.2.1 Constructor & Destructor Documentation	28
6.2.1.1 CCube() [1/6]	28
6.2.1.2 CCube() [2/6]	28
6.2.1.3 CCube() [3/6]	28
6.2.1.4 CCube() [4/6]	29
6.2.1.5 CCube() [5/6]	29
6.2.1.6 CCube() [6/6]	29

6.2.1.7 <code>~CCube()</code>	29
6.2.2 Member Function Documentation	29
6.2.2.1 <code>GetEdge()</code>	29
6.2.2.2 <code>GetFacet()</code>	29
6.2.2.3 <code>GetNode()</code> [1/2]	30
6.2.2.4 <code>GetNode()</code> [2/2]	30
6.2.2.5 <code>GetNumberOfEdges()</code>	30
6.2.2.6 <code>GetNumberOfFacets()</code>	30
6.2.2.7 <code>GetNumberOfNodes()</code>	30
6.2.2.8 <code>IncreaseOrder()</code>	30
6.2.2.9 <code>Integrate()</code> [1/3]	31
6.2.2.10 <code>Integrate()</code> [2/3]	31
6.2.2.11 <code>Integrate()</code> [3/3]	31
6.2.2.12 <code>operator=()</code>	31
6.2.2.13 <code>operator==(())</code>	31
6.2.2.14 <code>operator>>()</code>	31
6.2.2.15 <code>SetEdge()</code>	32
6.2.2.16 <code>SetFacet()</code>	32
6.2.2.17 <code>SetNode()</code>	32
6.2.2.18 <code>SetOrder()</code>	32
6.3 <code>corenc::Mesh::CCubeBasis</code> Class Reference	32
6.3.1 Constructor & Destructor Documentation	33
6.3.1.1 <code>CCubeBasis()</code> [1/4]	33
6.3.1.2 <code>CCubeBasis()</code> [2/4]	33
6.3.1.3 <code>CCubeBasis()</code> [3/4]	33
6.3.1.4 <code>CCubeBasis()</code> [4/4]	34
6.3.1.5 <code>~CCubeBasis()</code>	34
6.3.2 Member Function Documentation	34
6.3.2.1 <code>GetGradShapeFunction()</code>	34
6.3.2.2 <code>GetMeasure()</code>	34
6.3.2.3 <code>GetNormal()</code>	34
6.3.2.4 <code>GetNumberOfShapeFunctions()</code>	34
6.3.2.5 <code>GetShapeFunction()</code>	35
6.3.2.6 <code>GetValue()</code>	35
6.3.2.7 <code>GetWeight()</code>	35
6.3.2.8 <code>IncreaseOrder()</code>	35
6.3.2.9 <code>operator=()</code>	35
6.3.2.10 <code>ReverseNormal()</code>	35
6.4 <code>corenc::method::CDGMethod< Type ></code> Class Template Reference	36
6.4.1 Constructor & Destructor Documentation	36
6.4.1.1 <code>CDGMethod()</code>	36
6.4.1.2 <code>~CDGMethod()</code>	36

6.4.2 Member Function Documentation	36
6.4.2.1 Assemble()	36
6.4.2.2 GetMaxSolution()	36
6.4.2.3 GetMinSolution()	37
6.4.2.4 GetSolution() [1/2]	37
6.4.2.5 GetSolution() [2/2]	37
6.5 corenc::method::CDGMethodZero< Type > Class Template Reference	37
6.5.1 Constructor & Destructor Documentation	37
6.5.1.1 CDGMethodZero()	38
6.5.1.2 ~CDGMethodZero()	38
6.5.2 Member Function Documentation	38
6.5.2.1 Assemble()	38
6.5.2.2 GetMaxSolution()	38
6.5.2.3 GetMinSolution()	38
6.5.2.4 GetSolution() [1/2]	38
6.5.2.5 GetSolution() [2/2]	39
6.6 corenc::CDiffusionScalar Class Reference	39
6.6.1 Constructor & Destructor Documentation	40
6.6.1.1 CDiffusionScalar()	40
6.6.1.2 ~CDiffusionScalar()	40
6.6.2 Member Function Documentation	40
6.6.2.1 add_boundary_parameter() [1/2]	40
6.6.2.2 add_boundary_parameter() [2/2]	40
6.6.2.3 add_parameter() [1/3]	41
6.6.2.4 add_parameter() [2/3]	41
6.6.2.5 add_parameter() [3/3]	41
6.6.2.6 addTerm()	41
6.6.2.7 findTerm()	41
6.6.2.8 get_boundary_parameter() [1/3]	41
6.6.2.9 get_boundary_parameter() [2/3]	42
6.6.2.10 get_boundary_parameter() [3/3]	42
6.6.2.11 get_boundary_type()	42
6.6.2.12 get_number_of_boundaries()	42
6.6.2.13 get_parameter() [1/5]	42
6.6.2.14 get_parameter() [2/5]	42
6.6.2.15 get_parameter() [3/5]	43
6.6.2.16 get_parameter() [4/5]	43
6.6.2.17 get_parameter() [5/5]	43
6.6.2.18 get_point_source()	43
6.6.2.19 get_total_sources()	43
6.6.2.20 getNumberOfTerms()	43
6.6.2.21 getTerm()	44

6.6.2.22 load_parameters()	44
6.6.2.23 removeTerm()	44
6.6.2.24 set_boundary_parameter()	44
6.6.2.25 set_parameter() [1/2]	44
6.6.2.26 set_parameter() [2/2]	44
6.6.2.27 set_point_source()	45
6.6.2.28 setTerm()	45
6.7 corenc::Mesh::CEdge Class Reference	45
6.7.1 Constructor & Destructor Documentation	46
6.7.1.1 CEdge() [1/4]	46
6.7.1.2 CEdge() [2/4]	46
6.7.1.3 CEdge() [3/4]	46
6.7.1.4 CEdge() [4/4]	46
6.7.1.5 ~CEdge()	46
6.7.2 Member Function Documentation	46
6.7.2.1 GetNode() [1/2]	47
6.7.2.2 GetNode() [2/2]	47
6.7.2.3 GetNumberOfNodes()	47
6.7.2.4 IncreaseOrder()	47
6.7.2.5 Integrate() [1/3]	47
6.7.2.6 Integrate() [2/3]	47
6.7.2.7 Integrate() [3/3]	48
6.7.2.8 operator=()	48
6.7.2.9 SetNode()	48
6.7.3 Friends And Related Function Documentation	48
6.7.3.1 operator==	48
6.7.3.2 operator>>	48
6.8 corenc::Mesh::CEdge2ndBasis Class Reference	49
6.8.1 Constructor & Destructor Documentation	49
6.8.1.1 CEdge2ndBasis() [1/4]	49
6.8.1.2 CEdge2ndBasis() [2/4]	49
6.8.1.3 CEdge2ndBasis() [3/4]	50
6.8.1.4 CEdge2ndBasis() [4/4]	50
6.8.1.5 ~CEdge2ndBasis()	50
6.8.2 Member Function Documentation	50
6.8.2.1 GetGradShapeFunction()	50
6.8.2.2 GetMeasure()	50
6.8.2.3 GetNormal()	50
6.8.2.4 GetNumberOfShapeFunctions()	51
6.8.2.5 GetShapeFunction()	51
6.8.2.6 GetWeight()	51
6.8.2.7 IncreaseOrder()	51

6.8.2.8 operator=()	51
6.8.2.9 ReverseNormal()	51
6.9 corenc::Mesh::CEdgeConstantBasis Class Reference	52
6.9.1 Constructor & Destructor Documentation	52
6.9.1.1 CEdgeConstantBasis() [1/4]	52
6.9.1.2 CEdgeConstantBasis() [2/4]	52
6.9.1.3 CEdgeConstantBasis() [3/4]	53
6.9.1.4 CEdgeConstantBasis() [4/4]	53
6.9.1.5 ~CEdgeConstantBasis()	53
6.9.2 Member Function Documentation	53
6.9.2.1 GetGradShapeFunction()	53
6.9.2.2 GetMeasure()	53
6.9.2.3 GetNormal()	53
6.9.2.4 GetNumberOfShapeFunctions()	54
6.9.2.5 GetShapeFunction()	54
6.9.2.6 GetWeight()	54
6.9.2.7 IncreaseOrder()	54
6.9.2.8 operator=()	54
6.9.2.9 ReverseNormal()	54
6.10 corenc::Mesh::CEdgeHermiteBasis Class Reference	55
6.10.1 Constructor & Destructor Documentation	55
6.10.1.1 CEdgeHermiteBasis() [1/4]	55
6.10.1.2 CEdgeHermiteBasis() [2/4]	55
6.10.1.3 CEdgeHermiteBasis() [3/4]	56
6.10.1.4 CEdgeHermiteBasis() [4/4]	56
6.10.1.5 ~CEdgeHermiteBasis()	56
6.10.2 Member Function Documentation	56
6.10.2.1 GetGradShapeFunction()	56
6.10.2.2 GetMeasure()	56
6.10.2.3 GetNormal()	56
6.10.2.4 GetNumberOfShapeFunctions()	57
6.10.2.5 GetShapeFunction()	57
6.10.2.6 GetWeight()	57
6.10.2.7 IncreaseOrder()	57
6.10.2.8 operator=()	57
6.10.2.9 ReverseNormal()	57
6.11 corenc::Mesh::CEdgeLinearBasis Class Reference	58
6.11.1 Constructor & Destructor Documentation	58
6.11.1.1 CEdgeLinearBasis() [1/4]	58
6.11.1.2 CEdgeLinearBasis() [2/4]	58
6.11.1.3 CEdgeLinearBasis() [3/4]	59
6.11.1.4 CEdgeLinearBasis() [4/4]	59

6.11.1.5 <code>~CEdgeLinearBasis()</code>	59
6.11.2 Member Function Documentation	59
6.11.2.1 <code>GetGradShapeFunction()</code>	59
6.11.2.2 <code>GetMeasure()</code>	59
6.11.2.3 <code>GetNormal()</code>	59
6.11.2.4 <code>GetNumberOfShapeFunctions()</code>	60
6.11.2.5 <code>GetShapeFunction()</code>	60
6.11.2.6 <code>GetWeight()</code>	60
6.11.2.7 <code>IncreaseOrder()</code>	60
6.11.2.8 <code>operator=()</code>	60
6.11.2.9 <code>ReverseNormal()</code>	60
6.12 <code>corenc::Mesh::CEdgeMultiBasis</code> Class Reference	61
6.12.1 Constructor & Destructor Documentation	61
6.12.1.1 <code>CEdgeMultiBasis()</code> [1/4]	61
6.12.1.2 <code>CEdgeMultiBasis()</code> [2/4]	61
6.12.1.3 <code>CEdgeMultiBasis()</code> [3/4]	62
6.12.1.4 <code>CEdgeMultiBasis()</code> [4/4]	62
6.12.1.5 <code>~CEdgeMultiBasis()</code>	62
6.12.2 Member Function Documentation	62
6.12.2.1 <code>GetGradShapeFunction()</code>	62
6.12.2.2 <code>GetMeasure()</code>	62
6.12.2.3 <code>GetNormal()</code>	62
6.12.2.4 <code>GetNumberOfShapeFunctions()</code>	63
6.12.2.5 <code>GetShapeFunction()</code>	63
6.12.2.6 <code>GetWeight()</code>	63
6.12.2.7 <code>IncreaseOrder()</code>	63
6.12.2.8 <code>operator=()</code>	63
6.12.2.9 <code>ReverseNormal()</code>	63
6.13 <code>corenc::Mesh::CElement< T ></code> Class Template Reference	64
6.13.1 Constructor & Destructor Documentation	64
6.13.1.1 <code>CElement()</code>	64
6.13.1.2 <code>~CElement()</code>	65
6.13.2 Member Function Documentation	65
6.13.2.1 <code>Clone()</code>	65
6.13.2.2 <code>GetDoFs()</code>	65
6.13.2.3 <code>GetGradShapeFunction()</code>	65
6.13.2.4 <code>GetMeasure()</code>	65
6.13.2.5 <code>GetNeighbour()</code>	66
6.13.2.6 <code>GetNode()</code>	66
6.13.2.7 <code>GetNormal()</code>	66
6.13.2.8 <code>GetNumberOfNodes()</code>	66
6.13.2.9 <code>GetShapeFunction()</code>	66

6.13.2.10 GetType()	67
6.13.2.11 GetWeight()	67
6.13.2.12 IncreaseOrder()	67
6.13.2.13 Integrate() [1/3]	67
6.13.2.14 Integrate() [2/3]	67
6.13.2.15 Integrate() [3/3]	68
6.13.2.16 ReverseNormal()	68
6.13.2.17 SetNeighbour()	68
6.13.2.18 SetNode()	68
6.13.2.19 SetType()	69
6.14 corenc::Mesh::CElement2D< T > Class Template Reference	69
6.14.1 Constructor & Destructor Documentation	69
6.14.1.1 CElement2D()	70
6.14.1.2 ~CElement2D()	70
6.14.2 Member Function Documentation	70
6.14.2.1 Clone()	70
6.14.2.2 GetDoFs()	70
6.14.2.3 GetGradShapeFunction()	70
6.14.2.4 GetMeasure()	71
6.14.2.5 GetNeighbour()	71
6.14.2.6 GetNode()	71
6.14.2.7 GetNormal()	71
6.14.2.8 GetNumberOfNodes()	71
6.14.2.9 GetShapeFunction()	72
6.14.2.10 GetType()	72
6.14.2.11 GetWeight()	72
6.14.2.12 IncreaseOrder()	72
6.14.2.13 Integrate() [1/3]	72
6.14.2.14 Integrate() [2/3]	73
6.14.2.15 Integrate() [3/3]	73
6.14.2.16 ReverseNormal()	73
6.14.2.17 SetNeighbour()	73
6.14.2.18 SetNode()	73
6.14.2.19 SetOrder()	74
6.14.2.20 SetType()	74
6.15 corenc::Mesh::CElement2D< bool > Class Reference	74
6.15.1 Constructor & Destructor Documentation	75
6.15.1.1 CElement2D()	75
6.15.1.2 ~CElement2D()	75
6.15.2 Member Function Documentation	75
6.15.2.1 Clone()	75
6.15.2.2 GetDoFs()	75

6.15.2.3 GetGradShapeFunction()	75
6.15.2.4 GetMeasure()	75
6.15.2.5 GetNeighbour()	76
6.15.2.6 GetNode()	76
6.15.2.7 GetNormal()	76
6.15.2.8 GetNumberOfNodes()	76
6.15.2.9 GetShapeFunction()	76
6.15.2.10 GetType()	76
6.15.2.11 GetWeight()	76
6.15.2.12 IncreaseOrder()	77
6.15.2.13 Integrate() [1/3]	77
6.15.2.14 Integrate() [2/3]	77
6.15.2.15 Integrate() [3/3]	77
6.15.2.16 ReverseNormal()	77
6.15.2.17 SetNeighbour()	77
6.15.2.18 SetNode()	78
6.15.2.19 SetOrder()	78
6.15.2.20 SetType()	78
6.16 corenc::Mesh::CElement< bool > Class Reference	78
6.16.1 Constructor & Destructor Documentation	79
6.16.1.1 CElement()	79
6.16.1.2 ~CElement()	79
6.16.2 Member Function Documentation	79
6.16.2.1 Clone()	79
6.16.2.2 GetDoFs()	79
6.16.2.3 GetGradShapeFunction()	79
6.16.2.4 GetMeasure()	79
6.16.2.5 GetNeighbour()	80
6.16.2.6 GetNode()	80
6.16.2.7 GetNormal()	80
6.16.2.8 GetNumberOfNodes()	80
6.16.2.9 GetShapeFunction()	80
6.16.2.10 GetType()	80
6.16.2.11 GetWeight()	80
6.16.2.12 IncreaseOrder()	81
6.16.2.13 Integrate() [1/3]	81
6.16.2.14 Integrate() [2/3]	81
6.16.2.15 Integrate() [3/3]	81
6.16.2.16 ReverseNormal()	81
6.16.2.17 SetNeighbour()	81
6.16.2.18 SetNode()	82
6.16.2.19 SetType()	82

6.17 corenc::method::CFEMethod< Type > Class Template Reference	82
6.17.1 Constructor & Destructor Documentation	82
6.17.1.1 CFEMethod()	82
6.17.1.2 ~CFEMethod()	82
6.17.2 Member Function Documentation	83
6.17.2.1 Assemble()	83
6.17.2.2 GetMaxSolution()	83
6.17.2.3 GetMinSolution()	83
6.17.2.4 GetSolution() [1/2]	83
6.17.2.5 GetSolution() [2/2]	83
6.18 corenc::method::CFEMethodZero< Type > Class Template Reference	83
6.18.1 Constructor & Destructor Documentation	84
6.18.1.1 CFEMethodZero()	84
6.18.1.2 ~CFEMethodZero()	84
6.18.2 Member Function Documentation	84
6.18.2.1 Assemble()	84
6.18.2.2 GetMaxSolution()	84
6.18.2.3 GetMinSolution()	85
6.18.2.4 GetSolution() [1/2]	85
6.18.2.5 GetSolution() [2/2]	85
6.19 corenc::CFESolution Class Reference	85
6.19.1 Constructor & Destructor Documentation	86
6.19.1.1 CFESolution() [1/3]	86
6.19.1.2 ~CFESolution()	86
6.19.1.3 CFESolution() [2/3]	86
6.19.1.4 CFESolution() [3/3]	86
6.19.2 Member Function Documentation	86
6.19.2.1 operator double()	86
6.19.2.2 operator!=()	87
6.19.2.3 operator*=()	87
6.19.2.4 operator+=()	87
6.19.2.5 operator-=()	87
6.19.2.6 operator/=()	87
6.19.2.7 operator=() [1/2]	87
6.19.2.8 operator=() [2/2]	87
6.19.2.9 operator==()	88
6.19.3 Friends And Related Function Documentation	88
6.19.3.1 operator* [1/3]	88
6.19.3.2 operator* [2/3]	88
6.19.3.3 operator* [3/3]	88
6.19.3.4 operator+	88
6.19.3.5 operator-	88

6.19.3.6 operator/	89
6.20 corenc::CFEweights Class Reference	89
6.20.1 Constructor & Destructor Documentation	89
6.20.1.1 CFEweights()	89
6.20.1.2 ~CFEweights()	89
6.20.2 Member Function Documentation	89
6.20.2.1 getWeight()	89
6.20.2.2 updateWeight()	90
6.21 corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T > Class Template Reference	90
6.21.1 Constructor & Destructor Documentation	91
6.21.1.1 CFiniteElement() [1/7]	91
6.21.1.2 CFiniteElement() [2/7]	91
6.21.1.3 CFiniteElement() [3/7]	91
6.21.1.4 CFiniteElement() [4/7]	92
6.21.1.5 CFiniteElement() [5/7]	92
6.21.1.6 CFiniteElement() [6/7]	92
6.21.1.7 CFiniteElement() [7/7]	92
6.21.1.8 ~CFiniteElement()	92
6.21.2 Member Function Documentation	92
6.21.2.1 Clone()	93
6.21.2.2 GetDoF()	93
6.21.2.3 GetDoFs()	93
6.21.2.4 GetGradShapeFunction()	93
6.21.2.5 GetMeasure()	93
6.21.2.6 GetNeighbour()	94
6.21.2.7 GetNode()	94
6.21.2.8 GetNormal()	94
6.21.2.9 GetNumberOfNodes()	94
6.21.2.10 GetShape()	94
6.21.2.11 GetShapeFunction()	95
6.21.2.12 GetShapeFunctions()	95
6.21.2.13 GetType()	95
6.21.2.14 GetWeight()	95
6.21.2.15 IncreaseOrder()	95
6.21.2.16 Integrate() [1/3]	96
6.21.2.17 Integrate() [2/3]	96
6.21.2.18 Integrate() [3/3]	96
6.21.2.19 operator=()	96
6.21.2.20 ReverseNormal()	96
6.21.2.21 SetDoF()	97
6.21.2.22 SetNeighbour()	97
6.21.2.23 SetNode()	97

6.21.2.24 SetShape()	97
6.21.2.25 SetShapeFunction()	97
6.21.2.26 SetType()	98
6.21.3 Friends And Related Function Documentation	98
6.21.3.1 operator==	98
6.21.3.2 operator>>	98
6.22 corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction > Class Template Reference	98
6.22.1 Constructor & Destructor Documentation	99
6.22.1.1 CFiniteElement2D() [1/8]	100
6.22.1.2 CFiniteElement2D() [2/8]	100
6.22.1.3 CFiniteElement2D() [3/8]	100
6.22.1.4 CFiniteElement2D() [4/8]	100
6.22.1.5 CFiniteElement2D() [5/8]	100
6.22.1.6 CFiniteElement2D() [6/8]	101
6.22.1.7 CFiniteElement2D() [7/8]	101
6.22.1.8 CFiniteElement2D() [8/8]	101
6.22.1.9 ~CFiniteElement2D()	101
6.22.2 Member Function Documentation	101
6.22.2.1 Clone()	101
6.22.2.2 GetDoFs()	102
6.22.2.3 GetGradShapeFunction()	102
6.22.2.4 GetMeasure()	102
6.22.2.5 GetNeighbour()	102
6.22.2.6 GetNode()	102
6.22.2.7 GetNormal()	103
6.22.2.8 GetNumberOfNodes()	103
6.22.2.9 GetShape()	103
6.22.2.10 GetShapeFunction()	103
6.22.2.11 GetShapeFunctions()	103
6.22.2.12 GetType()	103
6.22.2.13 GetWeight()	104
6.22.2.14 IncreaseOrder()	104
6.22.2.15 Integrate() [1/3]	104
6.22.2.16 Integrate() [2/3]	104
6.22.2.17 Integrate() [3/3]	104
6.22.2.18 operator=()	105
6.22.2.19 ReverseNormal()	105
6.22.2.20 SetNeighbour()	105
6.22.2.21 SetNode()	105
6.22.2.22 SetOrder()	105
6.22.2.23 SetShape()	106
6.22.2.24 SetShapeFunction()	106

6.22.2.25 SetType()	106
6.22.3 Friends And Related Function Documentation	106
6.22.3.1 operator==	106
6.22.3.2 operator>>	106
6.23 corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool > Class Template Reference	107
6.23.1 Constructor & Destructor Documentation	108
6.23.1.1 CFiniteElement() [1/8]	108
6.23.1.2 CFiniteElement() [2/8]	108
6.23.1.3 CFiniteElement() [3/8]	108
6.23.1.4 CFiniteElement() [4/8]	108
6.23.1.5 CFiniteElement() [5/8]	109
6.23.1.6 CFiniteElement() [6/8]	109
6.23.1.7 CFiniteElement() [7/8]	109
6.23.1.8 CFiniteElement() [8/8]	109
6.23.1.9 ~CFiniteElement()	109
6.23.2 Member Function Documentation	109
6.23.2.1 Clone()	110
6.23.2.2 GetDoFs()	110
6.23.2.3 GetGradShapeFunction()	110
6.23.2.4 GetMeasure()	110
6.23.2.5 GetNeighbour()	110
6.23.2.6 GetNode()	111
6.23.2.7 GetNormal()	111
6.23.2.8 GetNumberOfNodes()	111
6.23.2.9 GetShape()	111
6.23.2.10 GetShapeFunction()	111
6.23.2.11 GetShapeFunctions()	112
6.23.2.12 GetType()	112
6.23.2.13 GetWeight()	112
6.23.2.14 IncreaseOrder()	112
6.23.2.15 Integrate() [1/3]	112
6.23.2.16 Integrate() [2/3]	113
6.23.2.17 Integrate() [3/3]	113
6.23.2.18 operator=()	113
6.23.2.19 ReverseNormal()	113
6.23.2.20 SetNeighbour()	113
6.23.2.21 SetNode()	114
6.23.2.22 SetShape()	114
6.23.2.23 SetShapeFunction()	114
6.23.2.24 SetType()	114
6.23.3 Friends And Related Function Documentation	114
6.23.3.1 operator==	114

6.23.3.2 operator>>	115
6.24 corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool > Class Template Reference	115
6.24.1 Constructor & Destructor Documentation	116
6.24.1.1 CFiniteElement() [1/7]	116
6.24.1.2 CFiniteElement() [2/7]	116
6.24.1.3 CFiniteElement() [3/7]	116
6.24.1.4 CFiniteElement() [4/7]	117
6.24.1.5 CFiniteElement() [5/7]	117
6.24.1.6 CFiniteElement() [6/7]	117
6.24.1.7 CFiniteElement() [7/7]	117
6.24.1.8 ~CFiniteElement()	117
6.24.2 Member Function Documentation	117
6.24.2.1 Clone()	118
6.24.2.2 GetDoF()	118
6.24.2.3 GetDoFs()	118
6.24.2.4 GetGradShapeFunction()	118
6.24.2.5 GetMeasure()	118
6.24.2.6 GetNeighbour()	119
6.24.2.7 GetNode()	119
6.24.2.8 GetNormal()	119
6.24.2.9 GetNumberOfNodes()	119
6.24.2.10 GetShape()	119
6.24.2.11 GetShapeFunction()	120
6.24.2.12 GetShapeFunctions()	120
6.24.2.13 GetType()	120
6.24.2.14 GetWeight()	120
6.24.2.15 IncreaseOrder()	120
6.24.2.16 Integrate() [1/3]	121
6.24.2.17 Integrate() [2/3]	121
6.24.2.18 Integrate() [3/3]	121
6.24.2.19 operator=()	121
6.24.2.20 ReverseNormal()	121
6.24.2.21 SetDoF()	122
6.24.2.22 SetNeighbour()	122
6.24.2.23 SetNode()	122
6.24.2.24 SetShape()	122
6.24.2.25 SetShapeFunction()	122
6.24.2.26 SetType()	123
6.24.3 Friends And Related Function Documentation	123
6.24.3.1 operator==	123
6.24.3.2 operator>>	123
6.25 corenc::CFiniteSolver< Method, Mesh, Solver > Class Template Reference	123

6.25.1 Constructor & Destructor Documentation	123
6.25.1.1 CFiniteSolver()	124
6.25.1.2 ~CFiniteSolver()	124
6.25.2 Member Function Documentation	124
6.25.2.1 Solve()	124
6.26 corenc::Mesh::CMesh< T > Class Template Reference	124
6.26.1 Constructor & Destructor Documentation	125
6.26.1.1 CMesh()	125
6.26.1.2 ~CMesh()	125
6.26.2 Member Function Documentation	125
6.26.2.1 FindElement()	125
6.26.2.2 GetBoundary()	125
6.26.2.3 GetElement()	125
6.26.2.4 getMinSize()	126
6.26.2.5 GetNode()	126
6.26.2.6 GetNumberOfBoundaries()	126
6.26.2.7 GetNumberOfElements()	126
6.26.2.8 GetNumberOfNodes()	126
6.26.2.9 getParameter() [1/2]	127
6.26.2.10 getParameter() [2/2]	127
6.26.2.11 getSolution() [1/2]	127
6.26.2.12 getSolution() [2/2]	127
6.26.2.13 setParameter()	127
6.26.2.14 updateSolution() [1/4]	128
6.26.2.15 updateSolution() [2/4]	128
6.26.2.16 updateSolution() [3/4]	128
6.26.2.17 updateSolution() [4/4]	128
6.27 corenc::Mesh::CMesh1D Class Reference	129
6.27.1 Constructor & Destructor Documentation	129
6.27.1.1 CMesh1D() [1/6]	130
6.27.1.2 CMesh1D() [2/6]	130
6.27.1.3 CMesh1D() [3/6]	130
6.27.1.4 CMesh1D() [4/6]	130
6.27.1.5 CMesh1D() [5/6]	130
6.27.1.6 CMesh1D() [6/6]	130
6.27.1.7 ~CMesh1D()	131
6.27.2 Member Function Documentation	131
6.27.2.1 FindElement()	131
6.27.2.2 GetBoundary() [1/2]	131
6.27.2.3 GetBoundary() [2/2]	131
6.27.2.4 GetElement()	131
6.27.2.5 GetElements()	131

6.27.2.6	getMinSize()	132
6.27.2.7	GetNode()	132
6.27.2.8	GetNumberOfBoundaries()	132
6.27.2.9	GetNumberOfElements()	132
6.27.2.10	GetNumberOfNodes()	132
6.27.2.11	getParameter() [1/2]	132
6.27.2.12	getParameter() [2/2]	133
6.27.2.13	getSolution() [1/2]	133
6.27.2.14	getSolution() [2/2]	133
6.27.2.15	operator=()	133
6.27.2.16	setParameter()	133
6.27.2.17	updateSolution() [1/4]	134
6.27.2.18	updateSolution() [2/4]	134
6.27.2.19	updateSolution() [3/4]	134
6.27.2.20	updateSolution() [4/4]	134
6.28	corenc::Mesh::CMesh< bool > Class Reference	134
6.28.1	Constructor & Destructor Documentation	135
6.28.1.1	CMesh()	135
6.28.1.2	~CMesh()	135
6.28.2	Member Function Documentation	135
6.28.2.1	FindElement()	135
6.28.2.2	GetBoundary()	136
6.28.2.3	GetElement()	136
6.28.2.4	getMinSize()	136
6.28.2.5	GetNode()	136
6.28.2.6	GetNumberOfBoundaries()	136
6.28.2.7	GetNumberOfElements()	136
6.28.2.8	GetNumberOfNodes()	136
6.28.2.9	getParameter() [1/2]	137
6.28.2.10	getParameter() [2/2]	137
6.28.2.11	getSolution() [1/2]	137
6.28.2.12	getSolution() [2/2]	137
6.28.2.13	setParameter()	137
6.28.2.14	updateSolution() [1/3]	137
6.28.2.15	updateSolution() [2/3]	138
6.28.2.16	updateSolution() [3/3]	138
6.29	corenc::Mesh::CNode Class Reference	138
6.29.1	Constructor & Destructor Documentation	139
6.29.1.1	CNode() [1/4]	139
6.29.1.2	CNode() [2/4]	139
6.29.1.3	CNode() [3/4]	139
6.29.1.4	CNode() [4/4]	139

6.29.1.5 <code>~CNode()</code>	139
6.29.2 Member Function Documentation	139
6.29.2.1 <code>GetNode()</code> [1/2]	140
6.29.2.2 <code>GetNode()</code> [2/2]	140
6.29.2.3 <code>GetNumberOfNodes()</code>	140
6.29.2.4 <code>IncreaseOrder()</code>	140
6.29.2.5 <code>Integrate()</code> [1/3]	140
6.29.2.6 <code>Integrate()</code> [2/3]	140
6.29.2.7 <code>Integrate()</code> [3/3]	141
6.29.2.8 <code>operator=()</code>	141
6.29.2.9 <code>SetNode()</code>	141
6.29.3 Friends And Related Function Documentation	141
6.29.3.1 <code>operator==</code>	141
6.29.3.2 <code>operator>></code>	141
6.30 <code>corenc::Mesh::CNodeBasis</code> Class Reference	142
6.30.1 Constructor & Destructor Documentation	142
6.30.1.1 <code>CNodeBasis()</code> [1/3]	142
6.30.1.2 <code>CNodeBasis()</code> [2/3]	142
6.30.1.3 <code>CNodeBasis()</code> [3/3]	143
6.30.1.4 <code>~CNodeBasis()</code>	143
6.30.2 Member Function Documentation	143
6.30.2.1 <code>GetGradShapeFunction()</code>	143
6.30.2.2 <code>GetMeasure()</code>	143
6.30.2.3 <code>GetNormal()</code>	143
6.30.2.4 <code>GetNumberOfShapeFunctions()</code>	143
6.30.2.5 <code>GetShapeFunction()</code>	144
6.30.2.6 <code>GetWeight()</code>	144
6.30.2.7 <code>IncreaseOrder()</code>	144
6.30.2.8 <code>operator=()</code>	144
6.30.2.9 <code>ReverseNormal()</code>	144
6.31 <code>corenc::Mesh::CParameter</code> Class Reference	144
6.31.1 Constructor & Destructor Documentation	145
6.31.1.1 <code>CParameter()</code> [1/3]	145
6.31.1.2 <code>CParameter()</code> [2/3]	145
6.31.1.3 <code>CParameter()</code> [3/3]	145
6.31.1.4 <code>~CParameter()</code>	145
6.31.2 Member Function Documentation	145
6.31.2.1 <code>GetAdvection()</code> [1/2]	146
6.31.2.2 <code>GetAdvection()</code> [2/2]	146
6.31.2.3 <code>GetDiffusion()</code> [1/2]	146
6.31.2.4 <code>GetDiffusion()</code> [2/2]	146
6.31.2.5 <code>GetMass()</code> [1/2]	146

6.31.2.6 GetMass() [2/2]	146
6.32 corenc::CProblem Class Reference	147
6.32.1 Constructor & Destructor Documentation	147
6.32.1.1 CProblem()	147
6.32.1.2 ~CProblem()	147
6.32.2 Member Function Documentation	147
6.32.2.1 addTerm()	147
6.32.2.2 getNumberOfTerms()	148
6.32.2.3 getTerm()	148
6.32.2.4 load_parameters()	148
6.32.2.5 setTerm()	148
6.33 corenc::Mesh::CRectangle Class Reference	148
6.33.1 Constructor & Destructor Documentation	149
6.33.1.1 CRectangle() [1/6]	149
6.33.1.2 CRectangle() [2/6]	149
6.33.1.3 CRectangle() [3/6]	150
6.33.1.4 CRectangle() [4/6]	150
6.33.1.5 CRectangle() [5/6]	150
6.33.1.6 CRectangle() [6/6]	150
6.33.1.7 ~CRectangle()	150
6.33.2 Member Function Documentation	150
6.33.2.1 GetEdge()	151
6.33.2.2 GetFacet()	151
6.33.2.3 GetNode() [1/2]	151
6.33.2.4 GetNode() [2/2]	151
6.33.2.5 GetNumberOfEdges()	151
6.33.2.6 GetNumberOfFacets()	151
6.33.2.7 GetNumberOfNodes()	152
6.33.2.8 IncreaseOrder()	152
6.33.2.9 Integrate() [1/3]	152
6.33.2.10 Integrate() [2/3]	152
6.33.2.11 Integrate() [3/3]	152
6.33.2.12 operator=()	152
6.33.2.13 operator==(())	153
6.33.2.14 operator>>()	153
6.33.2.15 SetEdge()	153
6.33.2.16 SetFacet()	153
6.33.2.17 SetNode()	153
6.33.2.18 SetOrder()	154
6.34 corenc::Mesh::CRectangleBasis Class Reference	154
6.34.1 Constructor & Destructor Documentation	154
6.34.1.1 CRectangleBasis() [1/4]	155

6.34.1.2 CRectangleBasis() [2 / 4]	155
6.34.1.3 CRectangleBasis() [3 / 4]	155
6.34.1.4 CRectangleBasis() [4 / 4]	155
6.34.1.5 ~CRectangleBasis()	155
6.34.2 Member Function Documentation	155
6.34.2.1 GetGradShapeFunction()	155
6.34.2.2 GetMeasure()	156
6.34.2.3 GetNormal()	156
6.34.2.4 GetNumberOfShapeFunctions()	156
6.34.2.5 GetShapeFunction()	156
6.34.2.6 GetValue()	156
6.34.2.7 GetWeight()	156
6.34.2.8 IncreaseOrder()	157
6.34.2.9 operator=()	157
6.34.2.10 ReverseNormal()	157
6.35 corenc::Mesh::CRectangleBasis2 Class Reference	157
6.35.1 Constructor & Destructor Documentation	158
6.35.1.1 CRectangleBasis2() [1 / 4]	158
6.35.1.2 CRectangleBasis2() [2 / 4]	158
6.35.1.3 CRectangleBasis2() [3 / 4]	158
6.35.1.4 CRectangleBasis2() [4 / 4]	158
6.35.1.5 ~CRectangleBasis2()	158
6.35.2 Member Function Documentation	158
6.35.2.1 GetGradShapeFunction()	159
6.35.2.2 GetMeasure()	159
6.35.2.3 GetNormal()	159
6.35.2.4 GetNumberOfShapeFunctions()	159
6.35.2.5 GetShapeFunction()	159
6.35.2.6 GetValue()	159
6.35.2.7 GetWeight()	160
6.35.2.8 IncreaseOrder()	160
6.35.2.9 operator=()	160
6.35.2.10 ReverseNormal()	160
6.36 corenc::Mesh::CRectangleBasis2x Class Reference	160
6.36.1 Constructor & Destructor Documentation	161
6.36.1.1 CRectangleBasis2x() [1 / 4]	161
6.36.1.2 CRectangleBasis2x() [2 / 4]	161
6.36.1.3 CRectangleBasis2x() [3 / 4]	161
6.36.1.4 CRectangleBasis2x() [4 / 4]	162
6.36.1.5 ~CRectangleBasis2x()	162
6.36.2 Member Function Documentation	162
6.36.2.1 GetGradShapeFunction()	162

6.36.2.2	GetMeasure()	162
6.36.2.3	GetNormal()	162
6.36.2.4	GetNumberOfShapeFunctions()	162
6.36.2.5	GetShapeFunction()	163
6.36.2.6	GetValue()	163
6.36.2.7	GetWeight()	163
6.36.2.8	IncreaseOrder()	163
6.36.2.9	operator=()	163
6.36.2.10	ReverseNormal()	163
6.37	corenc::Mesh::CRectangleBasis2y Class Reference	164
6.37.1	Constructor & Destructor Documentation	164
6.37.1.1	CRectangleBasis2y() [1/4]	164
6.37.1.2	CRectangleBasis2y() [2/4]	164
6.37.1.3	CRectangleBasis2y() [3/4]	165
6.37.1.4	CRectangleBasis2y() [4/4]	165
6.37.1.5	~CRectangleBasis2y()	165
6.37.2	Member Function Documentation	165
6.37.2.1	GetGradShapeFunction()	165
6.37.2.2	GetMeasure()	165
6.37.2.3	GetNormal()	165
6.37.2.4	GetNumberOfShapeFunctions()	166
6.37.2.5	GetShapeFunction()	166
6.37.2.6	GetValue()	166
6.37.2.7	GetWeight()	166
6.37.2.8	IncreaseOrder()	166
6.37.2.9	operator=()	166
6.37.2.10	ReverseNormal()	167
6.38	corenc::Mesh::CRectangleConstantBasis Class Reference	167
6.38.1	Constructor & Destructor Documentation	167
6.38.1.1	CRectangleConstantBasis() [1/4]	167
6.38.1.2	CRectangleConstantBasis() [2/4]	168
6.38.1.3	CRectangleConstantBasis() [3/4]	168
6.38.1.4	CRectangleConstantBasis() [4/4]	168
6.38.1.5	~CRectangleConstantBasis()	168
6.38.2	Member Function Documentation	168
6.38.2.1	GetGradShapeFunction()	168
6.38.2.2	GetMeasure()	169
6.38.2.3	GetNormal()	169
6.38.2.4	GetNumberOfShapeFunctions()	169
6.38.2.5	GetShapeFunction()	169
6.38.2.6	GetValue()	169
6.38.2.7	IncreaseOrder()	169

6.38.2.8 operator=()	170
6.38.2.9 ReverseNormal()	170
6.39 corenc::Mesh::CRectangleHBasis Class Reference	170
6.39.1 Constructor & Destructor Documentation	171
6.39.1.1 CRectangleHBasis() [1/6]	171
6.39.1.2 CRectangleHBasis() [2/6]	171
6.39.1.3 CRectangleHBasis() [3/6]	171
6.39.1.4 CRectangleHBasis() [4/6]	171
6.39.1.5 CRectangleHBasis() [5/6]	171
6.39.1.6 CRectangleHBasis() [6/6]	172
6.39.1.7 ~CRectangleHBasis()	172
6.39.2 Member Function Documentation	172
6.39.2.1 GetGradShapeFunction()	172
6.39.2.2 GetMeasure()	172
6.39.2.3 GetNormal()	172
6.39.2.4 GetNumberOfShapeFunctions()	172
6.39.2.5 GetShapeFunction()	173
6.39.2.6 GetValue()	173
6.39.2.7 GetWeight()	173
6.39.2.8 IncreaseOrder()	173
6.39.2.9 operator=()	173
6.39.2.10 ReverseNormal()	173
6.39.2.11 SetOrder()	174
6.40 corenc::Mesh::CRegularMesh Class Reference	174
6.40.1 Constructor & Destructor Documentation	175
6.40.1.1 CRegularMesh() [1/6]	175
6.40.1.2 CRegularMesh() [2/6]	175
6.40.1.3 CRegularMesh() [3/6]	175
6.40.1.4 CRegularMesh() [4/6]	175
6.40.1.5 CRegularMesh() [5/6]	175
6.40.1.6 CRegularMesh() [6/6]	176
6.40.1.7 ~CRegularMesh()	176
6.40.2 Member Function Documentation	176
6.40.2.1 Clone()	176
6.40.2.2 FindElement()	176
6.40.2.3 GetBoundary() [1/2]	176
6.40.2.4 GetBoundary() [2/2]	176
6.40.2.5 GetElement()	177
6.40.2.6 GetElements()	177
6.40.2.7 getMinSize()	177
6.40.2.8 GetNode()	177
6.40.2.9 GetNumberOfBoundaries()	177

6.40.2.10	GetNumberOfElements()	177
6.40.2.11	GetNumberOfNodes()	177
6.40.2.12	GetNumberOfNodes()	178
6.40.2.13	getParameter() [1/2]	178
6.40.2.14	getParameter() [2/2]	178
6.40.2.15	getSolution() [1/2]	178
6.40.2.16	getSolution() [2/2]	178
6.40.2.17	interpolate()	178
6.40.2.18	operator=()	179
6.40.2.19	refine_h()	179
6.40.2.20	refine_hp()	179
6.40.2.21	refine_hx()	179
6.40.2.22	refine_hy()	179
6.40.2.23	refine_p()	179
6.40.2.24	setParameter() [1/2]	179
6.40.2.25	setParameter() [2/2]	180
6.40.2.26	updateSolution() [1/4]	180
6.40.2.27	updateSolution() [2/4]	180
6.40.2.28	updateSolution() [3/4]	180
6.40.2.29	updateSolution() [4/4]	180
6.41	corenc::Mesh::CRegularMesh3D Class Reference	180
6.41.1	Constructor & Destructor Documentation	181
6.41.1.1	CRegularMesh3D() [1/6]	181
6.41.1.2	CRegularMesh3D() [2/6]	182
6.41.1.3	CRegularMesh3D() [3/6]	182
6.41.1.4	CRegularMesh3D() [4/6]	182
6.41.1.5	CRegularMesh3D() [5/6]	182
6.41.1.6	CRegularMesh3D() [6/6]	182
6.41.1.7	~CRegularMesh3D()	183
6.41.2	Member Function Documentation	183
6.41.2.1	Clone()	183
6.41.2.2	FindElement()	183
6.41.2.3	GetBoundary() [1/2]	183
6.41.2.4	GetBoundary() [2/2]	183
6.41.2.5	GetElement()	183
6.41.2.6	GetElements()	183
6.41.2.7	getMinSize()	184
6.41.2.8	GetNode()	184
6.41.2.9	GetNumberOfBoundaries()	184
6.41.2.10	GetNumberOfElements()	184
6.41.2.11	GetNumberOfNodes()	184
6.41.2.12	GetNumberOfNodes()	184

6.41.2.13	getParameter() [1/2]	184
6.41.2.14	getParameter() [2/2]	185
6.41.2.15	getSolution() [1/2]	185
6.41.2.16	getSolution() [2/2]	185
6.41.2.17	interpolate()	185
6.41.2.18	operator=()	185
6.41.2.19	refine_h()	185
6.41.2.20	refine_hp()	185
6.41.2.21	refine_hx()	186
6.41.2.22	refine_hy()	186
6.41.2.23	refine_p()	186
6.41.2.24	setParameter() [1/2]	186
6.41.2.25	setParameter() [2/2]	186
6.41.2.26	updateSolution() [1/4]	186
6.41.2.27	updateSolution() [2/4]	186
6.41.2.28	updateSolution() [3/4]	187
6.41.2.29	updateSolution() [4/4]	187
6.42	corenc::CShallowWater Class Reference	187
6.42.1	Constructor & Destructor Documentation	188
6.42.1.1	CShallowWater()	188
6.42.1.2	~CShallowWater()	188
6.42.2	Member Function Documentation	188
6.42.2.1	add_boundary_parameter() [1/2]	188
6.42.2.2	add_boundary_parameter() [2/2]	188
6.42.2.3	add_parameter()	189
6.42.2.4	addTerm()	189
6.42.2.5	get_boundary_parameter() [1/2]	189
6.42.2.6	get_boundary_parameter() [2/2]	189
6.42.2.7	get_boundary_type()	189
6.42.2.8	get_number_of_boundaries()	189
6.42.2.9	get_parameter() [1/2]	190
6.42.2.10	get_parameter() [2/2]	190
6.42.2.11	get_solution()	190
6.42.2.12	getNumberOfTerms()	190
6.42.2.13	getTerm()	190
6.42.2.14	load_parameters()	191
6.42.2.15	removeTerm()	191
6.42.2.16	set_boundary_parameter()	191
6.42.2.17	set_parameter()	191
6.42.2.18	setTerm()	191
6.43	corenc::Mesh::CShape Class Reference	192
6.43.1	Constructor & Destructor Documentation	192

6.43.1.1 CShape() [1/2]	192
6.43.1.2 CShape() [2/2]	192
6.43.1.3 ~CShape()	193
6.43.2 Member Function Documentation	193
6.43.2.1 GetEdge()	193
6.43.2.2 GetFacet()	193
6.43.2.3 GetNode() [1/2]	193
6.43.2.4 GetNode() [2/2]	193
6.43.2.5 GetNumberOfEdges()	194
6.43.2.6 GetNumberOfFacets()	194
6.43.2.7 GetNumberOfNodes()	194
6.43.2.8 Integrate() [1/3]	194
6.43.2.9 Integrate() [2/3]	194
6.43.2.10 Integrate() [3/3]	194
6.43.2.11 SetEdge()	195
6.43.2.12 SetFacet()	195
6.43.2.13 SetNode()	195
6.44 corenc::Mesh::CShapeFunction< Type > Class Template Reference	195
6.44.1 Constructor & Destructor Documentation	196
6.44.1.1 CShapeFunction() [1/2]	196
6.44.1.2 CShapeFunction() [2/2]	196
6.44.1.3 ~CShapeFunction()	196
6.44.2 Member Function Documentation	196
6.44.2.1 GetGradShapeFunction()	196
6.44.2.2 GetMeasure()	197
6.44.2.3 GetNormal()	197
6.44.2.4 GetNumberOfShapeFunctions()	197
6.44.2.5 GetShapeFunction()	197
6.44.2.6 ReverseNormal()	198
6.45 Methods::CSMethod Class Reference	198
6.45.1 Constructor & Destructor Documentation	198
6.45.1.1 CSMethod()	198
6.45.1.2 ~CSMethod()	198
6.46 corenc::CSolution Class Reference	199
6.46.1 Constructor & Destructor Documentation	199
6.46.1.1 CSolution()	199
6.46.1.2 ~CSolution()	199
6.47 corenc::Mesh::CTriangle Class Reference	199
6.47.1 Constructor & Destructor Documentation	200
6.47.1.1 CTriangle() [1/6]	200
6.47.1.2 CTriangle() [2/6]	200
6.47.1.3 CTriangle() [3/6]	201

6.47.1.4 CTriangle() [4/6]	201
6.47.1.5 CTriangle() [5/6]	201
6.47.1.6 CTriangle() [6/6]	201
6.47.1.7 ~CTriangle()	201
6.47.2 Member Function Documentation	201
6.47.2.1 GetEdge()	202
6.47.2.2 GetFacet()	202
6.47.2.3 GetNode() [1/2]	202
6.47.2.4 GetNode() [2/2]	202
6.47.2.5 GetNumberOfEdges()	202
6.47.2.6 GetNumberOfFacets()	202
6.47.2.7 GetNumberOfNodes()	203
6.47.2.8 IncreaseOrder()	203
6.47.2.9 Integrate() [1/3]	203
6.47.2.10 Integrate() [2/3]	203
6.47.2.11 Integrate() [3/3]	203
6.47.2.12 operator=()	203
6.47.2.13 operator==()	204
6.47.2.14 operator>>()	204
6.47.2.15 SetEdge()	204
6.47.2.16 SetFacet()	204
6.47.2.17 SetNode()	204
6.48 corenc::Mesh::CTriangleBasis Class Reference	205
6.48.1 Constructor & Destructor Documentation	205
6.48.1.1 CTriangleBasis() [1/8]	205
6.48.1.2 CTriangleBasis() [2/8]	206
6.48.1.3 CTriangleBasis() [3/8]	206
6.48.1.4 CTriangleBasis() [4/8]	206
6.48.1.5 ~CTriangleBasis() [1/2]	206
6.48.1.6 CTriangleBasis() [5/8]	206
6.48.1.7 CTriangleBasis() [6/8]	206
6.48.1.8 CTriangleBasis() [7/8]	207
6.48.1.9 CTriangleBasis() [8/8]	207
6.48.1.10 ~CTriangleBasis() [2/2]	207
6.48.2 Member Function Documentation	207
6.48.2.1 GetGradShapeFunction() [1/2]	207
6.48.2.2 GetGradShapeFunction() [2/2]	207
6.48.2.3 GetMeasure()	207
6.48.2.4 GetNormal() [1/2]	208
6.48.2.5 GetNormal() [2/2]	208
6.48.2.6 GetNumberOfShapeFunctions() [1/2]	208
6.48.2.7 GetNumberOfShapeFunctions() [2/2]	208

6.48.2.8	GetShapeFunction() [1/2]	208
6.48.2.9	GetShapeFunction() [2/2]	208
6.48.2.10	GetValue() [1/2]	209
6.48.2.11	GetValue() [2/2]	209
6.48.2.12	GetWeight()	209
6.48.2.13	IncreaseOrder()	209
6.48.2.14	operator=() [1/2]	209
6.48.2.15	operator=() [2/2]	209
6.48.2.16	ReverseNormal() [1/2]	209
6.48.2.17	ReverseNormal() [2/2]	210
6.49	corenc::Mesh::CTriangleLagrangeBasis Class Reference	210
6.49.1	Constructor & Destructor Documentation	210
6.49.1.1	CTriangleLagrangeBasis() [1/4]	211
6.49.1.2	CTriangleLagrangeBasis() [2/4]	211
6.49.1.3	CTriangleLagrangeBasis() [3/4]	211
6.49.1.4	CTriangleLagrangeBasis() [4/4]	211
6.49.1.5	~CTriangleLagrangeBasis()	211
6.49.2	Member Function Documentation	211
6.49.2.1	GetAlpha()	211
6.49.2.2	GetGradShapeFunction()	212
6.49.2.3	GetMeasure()	212
6.49.2.4	GetNormal()	212
6.49.2.5	GetNumberOfShapeFunctions()	212
6.49.2.6	GetShapeFunction()	212
6.49.2.7	GetValue()	212
6.49.2.8	GetWeight()	213
6.49.2.9	IncreaseOrder()	213
6.49.2.10	operator=()	213
6.49.2.11	ReverseNormal()	213
6.50	corenc::Mesh::CTriangleLinear Class Reference	213
6.50.1	Constructor & Destructor Documentation	214
6.50.1.1	CTriangleLinear() [1/6]	214
6.50.1.2	CTriangleLinear() [2/6]	214
6.50.1.3	CTriangleLinear() [3/6]	215
6.50.1.4	CTriangleLinear() [4/6]	215
6.50.1.5	CTriangleLinear() [5/6]	215
6.50.1.6	CTriangleLinear() [6/6]	215
6.50.1.7	~CTriangleLinear()	215
6.50.2	Member Function Documentation	215
6.50.2.1	GetEdge()	215
6.50.2.2	GetFacet()	216
6.50.2.3	GetNode() [1/2]	216

6.50.2.4	GetNode()	[2 / 2]	216
6.50.2.5	GetNumberOfEdges()		216
6.50.2.6	GetNumberOfFacets()		216
6.50.2.7	GetNumberOfNodes()		216
6.50.2.8	IncreaseOrder()		217
6.50.2.9	Integrate()	[1 / 3]	217
6.50.2.10	Integrate()	[2 / 3]	217
6.50.2.11	Integrate()	[3 / 3]	217
6.50.2.12	operator=()		217
6.50.2.13	operator==()		217
6.50.2.14	operator>>()		218
6.50.2.15	SetEdge()		218
6.50.2.16	SetFacet()		218
6.50.2.17	SetNode()		218
6.51	corenc::Mesh::CTriangleLinearBasis Class Reference		218
6.51.1	Constructor & Destructor Documentation		219
6.51.1.1	CTriangleLinearBasis()	[1 / 4]	219
6.51.1.2	CTriangleLinearBasis()	[2 / 4]	219
6.51.1.3	CTriangleLinearBasis()	[3 / 4]	219
6.51.1.4	CTriangleLinearBasis()	[4 / 4]	219
6.51.1.5	~CTriangleLinearBasis()		220
6.51.2	Member Function Documentation		220
6.51.2.1	GetGradShapeFunction()		220
6.51.2.2	GetMeasure()		220
6.51.2.3	GetNormal()		220
6.51.2.4	GetNumberOfShapeFunctions()		220
6.51.2.5	GetShapeFunction()		220
6.51.2.6	GetValue()		221
6.51.2.7	IncreaseOrder()		221
6.51.2.8	operator=()		221
6.51.2.9	ReverseNormal()		221
6.52	corenc::Mesh::CTriangularMesh Class Reference		221
6.52.1	Constructor & Destructor Documentation		222
6.52.1.1	CTriangularMesh()	[1 / 4]	222
6.52.1.2	CTriangularMesh()	[2 / 4]	223
6.52.1.3	CTriangularMesh()	[3 / 4]	223
6.52.1.4	CTriangularMesh()	[4 / 4]	223
6.52.1.5	~CTriangularMesh()		223
6.52.2	Member Function Documentation		223
6.52.2.1	Clone()		223
6.52.2.2	FindElement()		223
6.52.2.3	GetBoundary()	[1 / 2]	224

6.52.2.4	GetBoundary() [2/2]	224
6.52.2.5	GetElement()	224
6.52.2.6	GetElements()	224
6.52.2.7	getMinSize()	224
6.52.2.8	GetNode()	224
6.52.2.9	GetNumberOfBoundaries()	225
6.52.2.10	GetNumberOfElements()	225
6.52.2.11	GetNumberOfNodes()	225
6.52.2.12	GetNumberOfNodes()	225
6.52.2.13	getParameter() [1/2]	225
6.52.2.14	getParameter() [2/2]	225
6.52.2.15	getSolution() [1/2]	226
6.52.2.16	getSolution() [2/2]	226
6.52.2.17	interpolate()	226
6.52.2.18	operator=()	226
6.52.2.19	refine_h()	226
6.52.2.20	refine_hp()	226
6.52.2.21	refine_p()	226
6.52.2.22	set2ndOrder()	227
6.52.2.23	set3rdOrder()	227
6.52.2.24	set4thOrder()	227
6.52.2.25	setParameter() [1/2]	227
6.52.2.26	setParameter() [2/2]	227
6.52.2.27	updateSolution() [1/4]	227
6.52.2.28	updateSolution() [2/4]	228
6.52.2.29	updateSolution() [3/4]	228
6.52.2.30	updateSolution() [4/4]	228
6.53	corenc::Mesh::CTriangularMeshLinear Class Reference	228
6.53.1	Constructor & Destructor Documentation	229
6.53.1.1	CTriangularMeshLinear() [1/3]	229
6.53.1.2	CTriangularMeshLinear() [2/3]	229
6.53.1.3	CTriangularMeshLinear() [3/3]	229
6.53.1.4	~CTriangularMeshLinear()	230
6.53.2	Member Function Documentation	230
6.53.2.1	FindElement()	230
6.53.2.2	GetBoundary() [1/2]	230
6.53.2.3	GetBoundary() [2/2]	230
6.53.2.4	GetElement()	230
6.53.2.5	GetElements()	230
6.53.2.6	getMinSize()	231
6.53.2.7	GetNode()	231
6.53.2.8	GetNumberOfBoundaries()	231

6.53.2.9	GetNumberOfElements()	231
6.53.2.10	GetNumberOfNodes()	231
6.53.2.11	getParameter() [1/2]	231
6.53.2.12	getParameter() [2/2]	232
6.53.2.13	getSolution() [1/2]	232
6.53.2.14	getSolution() [2/2]	232
6.53.2.15	refine_h()	232
6.53.2.16	setParameter() [1/2]	232
6.53.2.17	setParameter() [2/2]	232
6.53.2.18	updateSolution() [1/4]	233
6.53.2.19	updateSolution() [2/4]	233
6.53.2.20	updateSolution() [3/4]	233
6.53.2.21	updateSolution() [4/4]	233
6.54	corenc::CVecSolution Class Reference	233
6.54.1	Constructor & Destructor Documentation	234
6.54.1.1	CVecSolution()	234
6.54.1.2	~CVecSolution()	234
6.54.2	Member Data Documentation	234
6.54.2.1	m_w	234
6.55	corenc::solvers::dg_shallow_water< Mesh > Class Template Reference	234
6.55.1	Constructor & Destructor Documentation	235
6.55.1.1	dg_shallow_water()	235
6.55.1.2	~dg_shallow_water()	235
6.55.2	Member Function Documentation	235
6.55.2.1	solve() [1/2]	235
6.55.2.2	solve() [2/2]	236
6.56	corenc::solvers::dg_solver< _Problem, _Mesh, _Result > Class Template Reference	236
6.56.1	Constructor & Destructor Documentation	236
6.56.1.1	dg_solver()	236
6.56.1.2	~dg_solver()	237
6.56.2	Member Function Documentation	237
6.56.2.1	elliptic_solver()	237
6.56.2.2	get_gradvalue() [1/2]	237
6.56.2.3	get_gradvalue() [2/2]	237
6.56.2.4	get_value() [1/3]	237
6.56.2.5	get_value() [2/3]	238
6.56.2.6	get_value() [3/3]	238
6.57	corenc::solvers::dg_solver_shallow_water Class Reference	238
6.57.1	Constructor & Destructor Documentation	238
6.57.1.1	dg_solver_shallow_water()	238
6.57.1.2	~dg_solver_shallow_water()	239
6.57.2	Member Function Documentation	239

6.57.2.1 solve() [1/2]	239
6.57.2.2 solve() [2/2]	239
6.58 corenc::method::DGMethod< Problem, Grid, Matrix > Class Template Reference	239
6.58.1 Constructor & Destructor Documentation	240
6.58.1.1 DGMethod() [1/6]	240
6.58.1.2 DGMethod() [2/6]	241
6.58.1.3 DGMethod() [3/6]	241
6.58.1.4 DGMethod() [4/6]	241
6.58.1.5 DGMethod() [5/6]	241
6.58.1.6 DGMethod() [6/6]	241
6.58.1.7 ~DGMethod()	241
6.58.2 Member Function Documentation	242
6.58.2.1 Discretization()	242
6.58.2.2 GetEffective()	242
6.58.2.3 GetGlobalMatrix()	242
6.58.2.4 GetGradSolution() [1/2]	242
6.58.2.5 GetGradSolution() [2/2]	242
6.58.2.6 GetMesh()	243
6.58.2.7 GetRightVector()	243
6.58.2.8 GetSolution() [1/3]	243
6.58.2.9 GetSolution() [2/3]	243
6.58.2.10 GetSolution() [3/3]	243
6.58.2.11 GetValue() [1/3]	243
6.58.2.12 GetValue() [2/3]	244
6.58.2.13 GetValue() [3/3]	244
6.58.2.14 LoadSolution()	244
6.58.2.15 OutDatFormat()	244
6.58.2.16 OutMeshFormat()	244
6.58.2.17 OutMeshTimeFormat()	245
6.58.2.18 ProjectSolution() [1/2]	245
6.58.2.19 ProjectSolution() [2/2]	245
6.58.2.20 Rediscretization() [1/2]	245
6.58.2.21 Rediscretization() [2/2]	245
6.58.2.22 SetSolution()	246
6.58.2.23 SetTimeStep()	246
6.59 corenc::method::DGMethodZero< Problem, Grid, Matrix > Class Template Reference	246
6.59.1 Constructor & Destructor Documentation	247
6.59.1.1 DGMethodZero() [1/6]	247
6.59.1.2 DGMethodZero() [2/6]	247
6.59.1.3 DGMethodZero() [3/6]	247
6.59.1.4 DGMethodZero() [4/6]	248
6.59.1.5 DGMethodZero() [5/6]	248

6.59.1.6 DGMethodZero() [6/6]	248
6.59.1.7 ~DGMethodZero()	248
6.59.2 Member Function Documentation	248
6.59.2.1 Discretization()	248
6.59.2.2 GetEffective()	248
6.59.2.3 GetGlobalMatrix()	249
6.59.2.4 GetGradSolution() [1/2]	249
6.59.2.5 GetGradSolution() [2/2]	249
6.59.2.6 GetMesh()	249
6.59.2.7 GetRightVector()	249
6.59.2.8 GetSolution() [1/3]	249
6.59.2.9 GetSolution() [2/3]	250
6.59.2.10 GetSolution() [3/3]	250
6.59.2.11 GetValue() [1/3]	250
6.59.2.12 GetValue() [2/3]	250
6.59.2.13 GetValue() [3/3]	250
6.59.2.14 LoadSolution()	251
6.59.2.15 OutDatFormat()	251
6.59.2.16 OutMeshFormat()	251
6.59.2.17 OutMeshTimeFormat()	251
6.59.2.18 ProjectSolution() [1/2]	251
6.59.2.19 ProjectSolution() [2/2]	252
6.59.2.20 Rediscretization() [1/2]	252
6.59.2.21 Rediscretization() [2/2]	252
6.59.2.22 SetSolution()	252
6.59.2.23 SetTimeStep()	252
6.60 corenc::method::DGSolution< Grid > Class Template Reference	253
6.60.1 Constructor & Destructor Documentation	253
6.60.1.1 DGSolution() [1/3]	253
6.60.1.2 DGSolution() [2/3]	253
6.60.1.3 DGSolution() [3/3]	253
6.60.1.4 ~DGSolution()	253
6.60.2 Member Function Documentation	254
6.60.2.1 getWeight()	254
6.60.2.2 getWeights()	254
6.60.2.3 operator=()	254
6.60.2.4 updateWeight()	254
6.61 corenc::solvers::eigen_solver< Matrix, Solver > Class Template Reference	254
6.61.1 Constructor & Destructor Documentation	255
6.61.1.1 eigen_solver()	255
6.61.1.2 ~eigen_solver()	255
6.61.2 Member Function Documentation	255

6.61.2.1 rayleigh()	255
6.62 Algebra::ESolver Class Reference	255
6.62.1 Constructor & Destructor Documentation	256
6.62.1.1 ESolver() [1/3]	256
6.62.1.2 ESolver() [2/3]	256
6.62.1.3 ESolver() [3/3]	257
6.62.1.4 ~ESolver()	257
6.62.2 Member Function Documentation	257
6.62.2.1 BiCGStab() [1/2]	257
6.62.2.2 BiCGStab() [2/2]	257
6.62.2.3 BiCGStabPrecond()	257
6.62.2.4 Gauss() [1/5]	257
6.62.2.5 Gauss() [2/5]	258
6.62.2.6 Gauss() [3/5]	258
6.62.2.7 Gauss() [4/5]	258
6.62.2.8 Gauss() [5/5]	258
6.62.2.9 GetSolution() [1/3]	258
6.62.2.10 GetSolution() [2/3]	258
6.62.2.11 GetSolution() [3/3]	259
6.62.2.12 GMRES() [1/2]	259
6.62.2.13 GMRES() [2/2]	259
6.62.2.14 MatrixprodVector() [1/4]	259
6.62.2.15 MatrixprodVector() [2/4]	259
6.62.2.16 MatrixprodVector() [3/4]	259
6.62.2.17 MatrixprodVector() [4/4]	260
6.62.2.18 Pardiso()	260
6.62.2.19 Reload()	260
6.62.2.20 Solve() [1/3]	260
6.62.2.21 Solve() [2/3]	260
6.62.2.22 Solve() [3/3]	261
6.63 corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 > Class Template Reference	261
6.63.1 Constructor & Destructor Documentation	261
6.63.1.1 FEAnalysis()	261
6.63.1.2 ~FEAnalysis()	261
6.63.2 Member Function Documentation	261
6.63.2.1 L2Norm()	262
6.64 corenc::solvers::fem_solver< _Problem, _Mesh, _Result > Class Template Reference	262
6.64.1 Constructor & Destructor Documentation	262
6.64.1.1 fem_solver()	262
6.64.1.2 ~fem_solver()	262
6.64.2 Member Function Documentation	263
6.64.2.1 elliptic_solver()	263

6.64.2.2 elliptic_solver_gauss()	263
6.64.2.3 get_gradvalue() [1/2]	263
6.64.2.4 get_gradvalue() [2/2]	263
6.64.2.5 get_value() [1/4]	264
6.64.2.6 get_value() [2/4]	264
6.64.2.7 get_value() [3/4]	264
6.64.2.8 get_value() [4/4]	264
6.65 corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result > Class Template Reference	264
6.65.1 Constructor & Destructor Documentation	265
6.65.1.1 fem_solver_lib()	265
6.65.1.2 ~fem_solver_lib()	265
6.65.2 Member Function Documentation	265
6.65.2.1 elliptic_solver()	265
6.65.2.2 elliptic_solver_gauss()	266
6.65.2.3 get_gradvalue() [1/2]	266
6.65.2.4 get_gradvalue() [2/2]	266
6.65.2.5 get_value() [1/4]	266
6.65.2.6 get_value() [2/4]	266
6.65.2.7 get_value() [3/4]	267
6.65.2.8 get_value() [4/4]	267
6.66 corenc::method::FEMethod< Problem, Grid, Matrix > Class Template Reference	267
6.66.1 Constructor & Destructor Documentation	268
6.66.1.1 FEMethod() [1/6]	268
6.66.1.2 FEMethod() [2/6]	268
6.66.1.3 FEMethod() [3/6]	269
6.66.1.4 FEMethod() [4/6]	269
6.66.1.5 FEMethod() [5/6]	269
6.66.1.6 FEMethod() [6/6]	269
6.66.1.7 ~FEMethod()	269
6.66.2 Member Function Documentation	269
6.66.2.1 Discretization()	270
6.66.2.2 GetEffective()	270
6.66.2.3 GetGlobalMatrix()	270
6.66.2.4 GetGradSolution() [1/2]	270
6.66.2.5 GetGradSolution() [2/2]	270
6.66.2.6 GetMesh()	270
6.66.2.7 GetRightVector()	271
6.66.2.8 GetSolution() [1/3]	271
6.66.2.9 GetSolution() [2/3]	271
6.66.2.10 GetSolution() [3/3]	271
6.66.2.11 GetValue() [1/3]	271
6.66.2.12 GetValue() [2/3]	271

6.66.2.13 GetValue() [3/3]	272
6.66.2.14 LoadSolution()	272
6.66.2.15 operator=()	272
6.66.2.16 OutDatFormat()	272
6.66.2.17 OutMeshFormat()	272
6.66.2.18 OutMeshTimeFormat()	273
6.66.2.19 ProjectSolution() [1/2]	273
6.66.2.20 ProjectSolution() [2/2]	273
6.66.2.21 Rediscretization() [1/2]	273
6.66.2.22 Rediscretization() [2/2]	273
6.66.2.23 SetSolution()	274
6.66.2.24 SetTimeStep()	274
6.67 corenc::method::FEMethodZero< Problem, Grid, Matrix > Class Template Reference	274
6.67.1 Constructor & Destructor Documentation	275
6.67.1.1 FEMethodZero() [1/6]	275
6.67.1.2 FEMethodZero() [2/6]	275
6.67.1.3 FEMethodZero() [3/6]	275
6.67.1.4 FEMethodZero() [4/6]	276
6.67.1.5 FEMethodZero() [5/6]	276
6.67.1.6 FEMethodZero() [6/6]	276
6.67.1.7 ~FEMethodZero()	276
6.67.2 Member Function Documentation	276
6.67.2.1 Discretization()	276
6.67.2.2 GetEffective()	276
6.67.2.3 GetGlobalMatrix()	277
6.67.2.4 GetGradSolution() [1/2]	277
6.67.2.5 GetGradSolution() [2/2]	277
6.67.2.6 GetMesh()	277
6.67.2.7 GetRightVector()	277
6.67.2.8 GetSolution() [1/3]	277
6.67.2.9 GetSolution() [2/3]	278
6.67.2.10 GetSolution() [3/3]	278
6.67.2.11 GetValue() [1/3]	278
6.67.2.12 GetValue() [2/3]	278
6.67.2.13 GetValue() [3/3]	278
6.67.2.14 LoadSolution()	279
6.67.2.15 OutDatFormat()	279
6.67.2.16 OutMeshFormat()	279
6.67.2.17 OutMeshTimeFormat()	279
6.67.2.18 ProjectSolution() [1/2]	279
6.67.2.19 ProjectSolution() [2/2]	280
6.67.2.20 Rediscretization() [1/2]	280

6.67.2.21 Rediscretization() [2/2]	280
6.67.2.22 SetSolution()	280
6.67.2.23 SetTimeStep()	280
6.68 corenc::method::FVMethod1d Class Reference	281
6.68.1 Constructor & Destructor Documentation	281
6.68.1.1 FVMethod1d()	281
6.68.1.2 ~FVMethod1d()	281
6.68.2 Member Function Documentation	281
6.68.2.1 GetSolution()	281
6.68.2.2 Solve()	282
6.69 corenc::Mesh::Gauss1dim Struct Reference	282
6.69.1 Member Data Documentation	282
6.69.1.1 m_a	282
6.69.1.2 m_order	283
6.69.1.3 m_sqrt35	283
6.69.1.4 m_w	283
6.70 corenc::Mesh::Gauss1dimN< N > Struct Template Reference	283
6.70.1 Member Data Documentation	283
6.70.1.1 m_a	284
6.70.1.2 m_order	284
6.70.1.3 m_w	284
6.71 corenc::GaussianKernel Struct Reference	284
6.71.1 Constructor & Destructor Documentation	284
6.71.1.1 GaussianKernel()	285
6.71.2 Member Function Documentation	285
6.71.2.1 get_gp()	285
6.71.2.2 gpexp()	285
6.71.2.3 gpstep()	285
6.71.3 Member Data Documentation	285
6.71.3.1 _centrs	285
6.71.3.2 N	285
6.72 corenc::GaussianProcess Struct Reference	286
6.72.1 Constructor & Destructor Documentation	286
6.72.1.1 GaussianProcess()	286
6.72.2 Member Function Documentation	286
6.72.2.1 He()	286
6.72.2.2 phi()	287
6.72.3 Member Data Documentation	287
6.72.3.1 a	287
6.72.3.2 A	287
6.72.3.3 b	287
6.72.3.4 B	287

6.72.3.5 c	287
6.72.3.6 K	287
6.72.3.7 l	288
6.72.3.8 lambda	288
6.72.3.9 sigma2	288
6.73 corenc::Mesh::GaussRectangular Struct Reference	288
6.73.1 Member Data Documentation	288
6.73.1.1 m_a	288
6.73.1.2 m_b	289
6.73.1.3 m_c	289
6.73.1.4 m_ra	289
6.73.1.5 m_rb	289
6.73.1.6 m_rw	289
6.73.1.7 m_wa	289
6.73.1.8 m_wb	289
6.73.1.9 m_wc	290
6.74 corenc::Mesh::GaussRectangularCubic Struct Reference	290
6.74.1 Member Data Documentation	290
6.74.1.1 m_a	290
6.74.1.2 m_b	290
6.74.1.3 m_c	291
6.74.1.4 m_ra	291
6.74.1.5 m_rb	291
6.74.1.6 m_rc	291
6.74.1.7 m_rw	291
6.74.1.8 m_s	291
6.74.1.9 m_w1	291
6.74.1.10 m_w2	291
6.74.1.11 m_w3	292
6.74.1.12 m_w4	292
6.75 corenc::Mesh::GaussTetrahedron Struct Reference	292
6.75.1 Member Data Documentation	292
6.75.1.1 m_la	292
6.75.1.2 m_lb	293
6.75.1.3 m_lc	293
6.75.1.4 m_ld	293
6.75.1.5 m_msq	293
6.75.1.6 m_psq	293
6.75.1.7 m_w	293
6.76 corenc::Mesh::GaussTriangle Struct Reference	293
6.76.1 Member Data Documentation	294
6.76.1.1 m_order	294

6.76.1.2 m_sqrt15	294
6.76.1.3 m_tra	294
6.76.1.4 m_trb	294
6.76.1.5 m_trw	295
6.77 Algebra::Matrix Class Reference	295
6.77.1 Detailed Description	295
6.77.2 Constructor & Destructor Documentation	295
6.77.2.1 Matrix() [1/3]	296
6.77.2.2 Matrix() [2/3]	296
6.77.2.3 ~Matrix()	296
6.77.2.4 Matrix() [3/3]	296
6.77.3 Member Function Documentation	296
6.77.3.1 AddElement()	296
6.77.3.2 Create() [1/2]	296
6.77.3.3 Create() [2/2]	297
6.77.3.4 GetElement()	297
6.77.3.5 GetSize()	297
6.77.3.6 NullMatrix()	297
6.77.3.7 NullRow()	297
6.77.3.8 operator()()	297
6.77.3.9 operator=()	298
6.78 Algebra::MatrixDiag Class Reference	298
6.78.1 Detailed Description	298
6.78.2 Constructor & Destructor Documentation	298
6.78.2.1 MatrixDiag() [1/3]	298
6.78.2.2 MatrixDiag() [2/3]	299
6.78.2.3 ~MatrixDiag()	299
6.78.2.4 MatrixDiag() [3/3]	299
6.78.3 Member Function Documentation	299
6.78.3.1 AddElement()	299
6.78.3.2 Create()	299
6.78.3.3 GetSize()	299
6.78.3.4 NullMatrix()	300
6.78.3.5 NullRow()	300
6.78.3.6 operator()()	300
6.78.3.7 operator=()	300
6.79 Algebra::MatrixSkyline Class Reference	300
6.79.1 Detailed Description	301
6.79.2 Constructor & Destructor Documentation	301
6.79.2.1 MatrixSkyline() [1/3]	301
6.79.2.2 MatrixSkyline() [2/3]	301
6.79.2.3 ~MatrixSkyline()	301

6.79.2.4 MatrixSkyline() [3/3]	301
6.79.3 Member Function Documentation	301
6.79.3.1 AddElement()	302
6.79.3.2 Create()	302
6.79.3.3 diff_skymatrix() [1/2]	302
6.79.3.4 diff_skymatrix() [2/2]	302
6.79.3.5 GetElement()	302
6.79.3.6 GetSize()	302
6.79.3.7 NullMatrix()	303
6.79.3.8 NullRow()	303
6.79.3.9 operator>() [1/2]	303
6.79.3.10 operator>() [2/2]	303
6.79.3.11 operator=()	303
6.79.3.12 transpose_sky()	303
6.80 corenc::multi_vector< T > Class Template Reference	304
6.80.1 Constructor & Destructor Documentation	304
6.80.1.1 multi_vector() [1/3]	304
6.80.1.2 multi_vector() [2/3]	304
6.80.1.3 multi_vector() [3/3]	304
6.80.1.4 ~multi_vector()	305
6.80.2 Member Function Documentation	305
6.80.2.1 fill_inc()	305
6.80.2.2 get() [1/2]	305
6.80.2.3 get() [2/2]	305
6.80.2.4 resize() [1/2]	305
6.80.2.5 resize() [2/2]	305
6.80.2.6 set()	306
6.80.2.7 size()	306
6.80.2.8 totalsize()	306
6.81 corenc::Mesh::parameter< T > Class Template Reference	306
6.81.1 Member Typedef Documentation	307
6.81.1.1 cfunc	307
6.81.1.2 cfunc_old	307
6.81.2 Constructor & Destructor Documentation	307
6.81.2.1 parameter() [1/6]	307
6.81.2.2 parameter() [2/6]	307
6.81.2.3 parameter() [3/6]	307
6.81.2.4 parameter() [4/6]	308
6.81.2.5 parameter() [5/6]	308
6.81.2.6 parameter() [6/6]	308
6.81.2.7 ~parameter()	308
6.81.3 Member Function Documentation	308

6.81.3.1 get() [1/3]	308
6.81.3.2 get() [2/3]	308
6.81.3.3 get() [3/3]	309
6.81.3.4 set()	309
6.82 corenc::Mesh::Point Class Reference	309
6.82.1 Constructor & Destructor Documentation	310
6.82.1.1 Point() [1/4]	310
6.82.1.2 Point() [2/4]	310
6.82.1.3 Point() [3/4]	310
6.82.1.4 Point() [4/4]	310
6.82.2 Member Function Documentation	310
6.82.2.1 Jacobian()	310
6.82.2.2 operator*()	310
6.82.2.3 operator*=()	311
6.82.2.4 operator+=()	311
6.82.2.5 operator<()	311
6.82.2.6 operator=()	311
6.82.2.7 operator==()	311
6.82.3 Friends And Related Function Documentation	311
6.82.3.1 operator"!="	311
6.82.3.2 operator* [1/3]	312
6.82.3.3 operator* [2/3]	312
6.82.3.4 operator* [3/3]	312
6.82.3.5 operator+	312
6.82.3.6 operator-	312
6.82.4 Member Data Documentation	312
6.82.4.1 x	312
6.82.4.2 y	313
6.82.4.3 z	313
6.83 corenc::Mesh::point_source< T > Class Template Reference	313
6.83.1 Constructor & Destructor Documentation	313
6.83.1.1 point_source() [1/2]	313
6.83.1.2 point_source() [2/2]	313
6.83.2 Member Function Documentation	314
6.83.2.1 get_point()	314
6.83.2.2 get_value()	314
6.83.2.3 operator=()	314
6.84 corenc::method::RungeKutta< Problem, Type > Class Template Reference	314
6.84.1 Constructor & Destructor Documentation	315
6.84.1.1 RungeKutta() [1/2]	315
6.84.1.2 RungeKutta() [2/2]	315
6.84.1.3 ~RungeKutta()	315

6.84.2 Member Function Documentation	315
6.84.2.1 discretize()	315
6.84.2.2 explicitEuler()	315
6.84.2.3 updateTimestep()	316
6.85 corenc::method::STSolution< Grid > Class Template Reference	316
6.85.1 Constructor & Destructor Documentation	316
6.85.1.1 STSolution() [1/4]	316
6.85.1.2 STSolution() [2/4]	316
6.85.1.3 STSolution() [3/4]	317
6.85.1.4 STSolution() [4/4]	317
6.85.1.5 ~STSolution()	317
6.85.2 Member Function Documentation	317
6.85.2.1 addTimeLayer()	317
6.85.2.2 getWeight()	317
6.85.2.3 getWeights()	318
6.85.2.4 operator=()	318
6.85.2.5 updateWeight()	318
6.86 corenc::method::system_dg_method< Problem, Grid, Matrix > Class Template Reference	318
6.86.1 Constructor & Destructor Documentation	319
6.86.1.1 system_dg_method() [1/2]	319
6.86.1.2 system_dg_method() [2/2]	319
6.86.1.3 ~system_dg_method()	319
6.86.2 Member Function Documentation	319
6.86.2.1 Assemble()	319
6.86.2.2 changeFlux()	320
6.86.2.3 DGtostandard()	320
6.86.2.4 GetGlobalMatrix()	320
6.86.2.5 GetMaxSolution()	320
6.86.2.6 GetMinSolution()	320
6.86.2.7 GetSolution() [1/4]	320
6.86.2.8 GetSolution() [2/4]	321
6.86.2.9 GetSolution() [3/4]	321
6.86.2.10 GetSolution() [4/4]	321
6.86.2.11 toDGSolution()	321
6.86.2.12 updateWeights()	321
6.87 corenc::method::system_dg_method< Grid, bool, bool > Class Template Reference	322
6.87.1 Member Function Documentation	322
6.87.1.1 GetSolution()	322
6.88 corenc::test_case_elliptic_fem Class Reference	322
6.88.1 Constructor & Destructor Documentation	323
6.88.1.1 test_case_elliptic_fem()	323
6.88.1.2 ~test_case_elliptic_fem()	323

6.88.2 Member Function Documentation	323
6.88.2.1 conv_diff_fem_fixed_triangle()	323
6.88.2.2 elliptic_2layer_fem_2d_tria_h()	323
6.88.2.3 elliptic_fem_2d_rect_source()	323
6.88.2.4 elliptic_fem_2d_tria()	323
6.88.2.5 elliptic_fem_hp_fixed()	324
6.88.2.6 elliptic_fem_hp_fixed_triangle()	324
6.88.2.7 elliptic_fem_hp_lagrange_triangle()	324
6.88.2.8 elliptic_fem_hxhy_fixed_triangle()	324
6.88.2.9 elliptic_fem_solver()	324
6.88.2.10 elliptic_fem_square_lin_basis()	324
6.88.2.11 elliptic_gaussian_triangle()	324
6.88.2.12 global_matrix()	325
6.88.2.13 homotopy_conv_diff_fem()	325
6.88.2.14 mass_matrix_3rd_order()	325
6.88.2.15 mass_matrix_4th_order()	325
6.88.2.16 stress_matrix_3rd_order()	325
6.88.2.17 stress_matrix_4th_order()	325
6.89 corenc::tests::test_case_rectanglebasis Class Reference	325
6.89.1 Constructor & Destructor Documentation	326
6.89.1.1 test_case_rectanglebasis()	326
6.89.1.2 ~test_case_rectanglebasis()	326
6.89.2 Member Function Documentation	326
6.89.2.1 mass_matrix()	326
6.89.2.2 stress_matrix()	326
6.90 corenc::tests::test_case_regular_mesh Class Reference	326
6.90.1 Constructor & Destructor Documentation	327
6.90.1.1 test_case_regular_mesh()	327
6.90.1.2 ~test_case_regular_mesh()	327
6.90.2 Member Function Documentation	327
6.90.2.1 construct_mesh()	327
6.91 corenc::test_case_solver Class Reference	327
6.91.1 Constructor & Destructor Documentation	328
6.91.1.1 test_case_solver()	328
6.91.1.2 ~test_case_solver()	328
6.91.2 Member Function Documentation	328
6.91.2.1 gauss_solver()	328
6.92 corenc::tests::test_case_trianglebasis Class Reference	328
6.92.1 Constructor & Destructor Documentation	328
6.92.1.1 test_case_trianglebasis()	329
6.92.1.2 ~test_case_trianglebasis()	329
6.92.2 Member Function Documentation	329

6.92.2.1 mass_matrix()	329
6.92.2.2 stress_matrix()	329
6.93 corenc::test_cases Class Reference	329
6.93.1 Constructor & Destructor Documentation	329
6.93.1.1 test_cases()	330
6.93.1.2 ~test_cases()	330
6.93.2 Member Function Documentation	330
6.93.2.1 perform() [1/3]	330
6.93.2.2 perform() [2/3]	330
6.93.2.3 perform() [3/3]	330
6.94 corenc::test_conv_diff Class Reference	330
6.94.1 Constructor & Destructor Documentation	331
6.94.1.1 test_conv_diff()	331
6.94.1.2 ~test_conv_diff()	331
6.94.2 Member Function Documentation	331
6.94.2.1 conv_diff_eigen()	331
6.94.2.2 conv_diff_fem()	331
6.95 corenc::solvers::vector_solution Struct Reference	332
6.95.1 Constructor & Destructor Documentation	332
6.95.1.1 vector_solution() [1/2]	332
6.95.1.2 vector_solution() [2/2]	332
6.95.2 Member Data Documentation	332
6.95.2.1 S	332
7 File Documentation	333
7.1 colors.h File Reference	333
7.2 colors.h	334
7.3 CoreNCA/Matrix.cpp File Reference	334
7.4 CoreNCA/Matrix.h File Reference	334
7.5 Matrix.h	335
7.6 CoreNCA/MatrixDiag.cpp File Reference	335
7.7 CoreNCA/MatrixDiag.h File Reference	335
7.8 MatrixDiag.h	336
7.9 CoreNCA/MatrixSkyline.cpp File Reference	336
7.9.1 Macro Definition Documentation	336
7.9.1.1 _NOPE_	337
7.9.1.2 N_MIN	337
7.10 CoreNCA/MatrixSkyline.h File Reference	337
7.11 MatrixSkyline.h	338
7.12 CoreNCFEM/FESolution.h File Reference	341
7.13 FESolution.h	342
7.14 CoreNCFEM/FiniteElements/CRectangleBasis2x.cpp File Reference	343

7.15 CoreNCFEM/FiniteElements/Cube.cpp File Reference	343
7.16 CoreNCFEM/FiniteElements/Cube.h File Reference	343
7.16.1 Macro Definition Documentation	344
7.16.1.1 CORENC_MESH_CUBE_H_	344
7.17 Cube.h	344
7.18 CoreNCFEM/FiniteElements/CubeHBasis.cpp File Reference	346
7.19 CoreNCFEM/FiniteElements/Edge.cpp File Reference	346
7.20 CoreNCFEM/FiniteElements/Edge.h File Reference	346
7.21 Edge.h	347
7.22 CoreNCFEM/FiniteElements/FiniteElement.h File Reference	350
7.23 FiniteElement.h	351
7.24 CoreNCFEM/FiniteElements/FiniteElement2D.h File Reference	363
7.25 FiniteElement2D.h	364
7.26 CoreNCFEM/FiniteElements/Node.cpp File Reference	368
7.27 CoreNCFEM/FiniteElements/Node.h File Reference	368
7.28 Node.h	369
7.29 CoreNCFEM/FiniteElements/Rectangle.cpp File Reference	370
7.30 CoreNCFEM/FiniteElements/Rectangle.h File Reference	370
7.31 Rectangle.h	371
7.32 CoreNCFEM/FiniteElements/RectangleBasis2.cpp File Reference	376
7.33 CoreNCFEM/FiniteElements/RectangleBasis2y.cpp File Reference	376
7.34 CoreNCFEM/FiniteElements/RectangleHBasis.cpp File Reference	376
7.35 CoreNCFEM/FiniteElements/Shape.h File Reference	377
7.36 Shape.h	377
7.37 CoreNCFEM/FiniteElements/ShapeFunction.h File Reference	378
7.38 ShapeFunction.h	378
7.39 CoreNCFEM/FiniteElements/Triangle.cpp File Reference	379
7.39.1 Function Documentation	379
7.39.1.1 center_point()	379
7.39.1.2 mid_point()	379
7.39.1.3 s_point()	380
7.40 CoreNCFEM/FiniteElements/Triangle.h File Reference	380
7.41 Triangle.h	380
7.42 CoreNCFEM/FiniteElements/TriangleLagrange.cpp File Reference	383
7.43 CoreNCFEM/FiniteElements/TriangleLinear.cpp File Reference	383
7.44 CoreNCFEM/FiniteElements/TriangleLinear.h File Reference	383
7.45 TriangleLinear.h	384
7.46 CoreNCFEM/FiniteSolver.h File Reference	386
7.47 FiniteSolver.h	386
7.48 CoreNCFEM/GaussianField.h File Reference	387
7.49 GaussianField.h	387
7.50 CoreNCFEM/Grids/Mesh1D.cpp File Reference	388

7.51 CoreNCFEM/Grids/Mesh1D.h File Reference	389
7.52 Mesh1D.h	389
7.53 CoreNCFEM/Grids/RegularMesh.cpp File Reference	390
7.53.1 Function Documentation	390
7.53.1.1 sort_indexes()	391
7.54 CoreNCFEM/Grids/RegularMesh.h File Reference	391
7.55 RegularMesh.h	391
7.56 CoreNCFEM/Grids/RegularMesh3D.cpp File Reference	393
7.56.1 Function Documentation	393
7.56.1.1 sort_indexes()	393
7.57 CoreNCFEM/Grids/RegularMesh3D.h File Reference	393
7.58 RegularMesh3D.h	394
7.59 CoreNCFEM/Grids/TriangularMesh.cpp File Reference	395
7.59.1 Function Documentation	395
7.59.1.1 sort_indexes()	396
7.60 CoreNCFEM/Grids/TriangularMesh.h File Reference	396
7.61 TriangularMesh.h	396
7.62 CoreNCFEM/Grids/TriangularMeshLinear.cpp File Reference	398
7.63 CoreNCFEM/Grids/TriangularMeshLinear.h File Reference	398
7.64 TriangularMeshLinear.h	398
7.65 CoreNCFEM/Mesh.h File Reference	399
7.66 Mesh.h	400
7.67 CoreNCFEM/Methods/CSMethod.h File Reference	401
7.68 CSMethod.h	401
7.69 CoreNCFEM/Methods/dg_flux.h File Reference	401
7.70 dg_flux.h	402
7.71 CoreNCFEM/Methods/DGMethod.h File Reference	402
7.72 DGMethod.h	403
7.73 CoreNCFEM/Methods/DGMethodZero.h File Reference	421
7.74 DGMethodZero.h	422
7.75 CoreNCFEM/Methods/DGSolution.h File Reference	440
7.76 DGSolution.h	441
7.77 CoreNCFEM/Methods/FEAnalysis.h File Reference	442
7.77.1 Macro Definition Documentation	442
7.77.1.1 CORENC_METHODS_FEANALYSIS_H_	442
7.78 FEAnalysis.h	443
7.79 CoreNCFEM/Methods/FEMethod.h File Reference	443
7.80 FEMethod.h	444
7.81 CoreNCFEM/Methods/FEMethodZero.h File Reference	458
7.82 FEMethodZero.h	458
7.83 CoreNCFEM/Methods/FVMethod.cpp File Reference	475
7.84 CoreNCFEM/Methods/FVMethod.h File Reference	475

7.85 FVMethod.h	476
7.86 CoreNCFEM/Methods/RungeKutta.h File Reference	476
7.87 RungeKutta.h	476
7.88 CoreNCFEM/Methods/system_dg_method.h File Reference	478
7.88.1 Macro Definition Documentation	479
7.88.1.1 CORENC_METHODS_SYSTEM_DG_METHOD_H_	479
7.89 system_dg_method.h	479
7.90 CoreNCFEM/multi_vector.h File Reference	486
7.90.1 Macro Definition Documentation	486
7.90.1.1 CORENC_MULTI_VECTOR_H_	486
7.91 multi_vector.h	487
7.92 CoreNCFEM/Parameter.cpp File Reference	488
7.93 CoreNCFEM/Parameter.h File Reference	488
7.93.1 Macro Definition Documentation	489
7.93.1.1 CORENC_MESH_PARAMETER_H_	489
7.94 Parameter.h	489
7.95 CoreNCFEM/Point.cpp File Reference	490
7.96 CoreNCFEM/Point.h File Reference	490
7.96.1 Macro Definition Documentation	491
7.96.1.1 CORENC_MESH_Point_h	491
7.97 Point.h	491
7.98 main.cpp File Reference	493
7.98.1 Function Documentation	493
7.98.1.1 main()	494
7.99 Problems/BurgersScalar.cpp File Reference	494
7.100 Problems/BurgersScalar.h File Reference	494
7.101 BurgersScalar.h	494
7.102 Problems/DiffusionScalar.cpp File Reference	495
7.103 Problems/DiffusionScalar.h File Reference	495
7.104 DiffusionScalar.h	495
7.105 Problems/Problems.h File Reference	496
7.105.1 Macro Definition Documentation	497
7.105.1.1 CORENC_PROBLEMS_PROBLEMS_H_	497
7.106 Problems.h	497
7.107 Problems/ShallowWater.cpp File Reference	497
7.108 Problems/ShallowWater.h File Reference	497
7.109 ShallowWater.h	498
7.110 Solvers/dg_solver.h File Reference	498
7.111 dg_solver.h	499
7.112 Solvers/dg_solver_shallow_water.cpp File Reference	500
7.113 Solvers/dg_solver_shallow_water.h File Reference	500
7.114 dg_solver_shallow_water.h	501

7.115 Solvers/eigen_solver.h File Reference	508
7.116 eigen_solver.h	508
7.117 Solvers/fem_solver.h File Reference	509
7.118 fem_solver.h	509
7.119 Solvers/fem_solver_lib.h File Reference	512
7.120 fem_solver_lib.h	512
7.121 Tests/FiniteElements/test_case_rectanglebasis.cpp File Reference	515
7.122 Tests/FiniteElements/test_case_rectanglebasis.h File Reference	515
7.122.1 Macro Definition Documentation	516
7.122.1.1 CORENC_TEST_CASE_RECTANGLEBASIS_H_	516
7.123 test_case_rectanglebasis.h	516
7.124 Tests/FiniteElements/test_case_trianglebasis.cpp File Reference	516
7.125 Tests/FiniteElements/test_case_trianglebasis.h File Reference	516
7.125.1 Macro Definition Documentation	517
7.125.1.1 CORENC_TEST_CASE_TRIANGLEBASIS_H_	517
7.126 test_case_trianglebasis.h	517
7.127 Tests/test_case_elliptic_fem.cpp File Reference	517
7.127.1 Macro Definition Documentation	518
7.127.1.1 _USE_MATH_DEFINES	518
7.127.2 Function Documentation	518
7.127.2.1 kekus()	518
7.128 Tests/test_case_elliptic_fem.h File Reference	518
7.129 test_case_elliptic_fem.h	519
7.130 Tests/test_case_regular_mesh.cpp File Reference	519
7.131 Tests/test_case_regular_mesh.h File Reference	519
7.131.1 Macro Definition Documentation	520
7.131.1.1 CORENC_TEST_CASE_REGULAR_MESH_H_	520
7.132 test_case_regular_mesh.h	520
7.133 Tests/test_case_solver.cpp File Reference	520
7.133.1 Macro Definition Documentation	521
7.133.1.1 _USE_MATH_DEFINES	521
7.133.2 Function Documentation	521
7.133.2.1 solver()	521
7.134 Tests/test_case_solver.h File Reference	521
7.135 test_case_solver.h	521
7.136 Tests/test_cases.cpp File Reference	522
7.137 Tests/test_cases.h File Reference	522
7.137.1 Macro Definition Documentation	522
7.137.1.1 CORENC_TEST_CASES_H_	522
7.138 test_cases.h	523
7.139 Tests/test_conv_diff.cpp File Reference	523
7.139.1 Macro Definition Documentation	523

7.139.1.1 <code>_USE_MATH_DEFINES</code>	523
7.140 Tests/test_conv_diff.h File Reference	524
7.141 test_conv_diff.h	524
Index	525

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Algebra	13
corenc	14
corenc::color	16
corenc::Mesh	19
corenc::method	21
corenc::solvers	23
corenc::tests	23
Methods	23
wtf	23

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

corenc::method::CDGMethod< Type >	36
corenc::method::CDGMethodZero< Type >	37
corenc::Mesh::CElement< T >	64
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >	90
corenc::Mesh::CElement2D< T >	69
corenc::Mesh::CElement2D< bool >	74
corenc::Mesh::CElement2D<>	69
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >	98
corenc::Mesh::CElement< bool >	78
corenc::Mesh::CElement<>	64
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >	115
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >	107
corenc::method::CFEMethod< Type >	82
corenc::method::CFEMethodZero< Type >	83
corenc::CFEweights	89
corenc::CFiniteSolver< Method, Mesh, Solver >	123
corenc::Mesh::CMesh< T >	124
corenc::Mesh::CMesh< bool >	134
corenc::Mesh::CMesh< CFESolution >	124
corenc::Mesh::CMesh1D	129
corenc::Mesh::CMesh<>	124
corenc::Mesh::CTriangularMesh	221
corenc::Mesh::CTriangularMeshLinear	228
corenc::Mesh::CParameter	144
corenc::CProblem	147
corenc::CBurgersScalar	25
corenc::CDiffusionScalar	39
corenc::CShallowWater	187
corenc::Mesh::CRegularMesh	174
corenc::Mesh::CRegularMesh3D	180
corenc::Mesh::CShape	192
corenc::Mesh::CCube	27
corenc::Mesh::CEdge	45

corenc::Mesh::CNode	138
corenc::Mesh::CRectangle	148
corenc::Mesh::CTriangle	199
corenc::Mesh::CTriangleLinear	213
corenc::Mesh::CShapeFunction< Type >	195
corenc::Mesh::CShapeFunction< double >	195
corenc::Mesh::CCubeBasis	32
corenc::Mesh::CEdge2ndBasis	49
corenc::Mesh::CEdgeConstantBasis	52
corenc::Mesh::CEdgeHermiteBasis	55
corenc::Mesh::CEdgeLinearBasis	58
corenc::Mesh::CEdgeMultiBasis	61
corenc::Mesh::CNodeBasis	142
corenc::Mesh::CRectangleBasis	154
corenc::Mesh::CRectangleBasis2	157
corenc::Mesh::CRectangleBasis2x	160
corenc::Mesh::CRectangleBasis2y	164
corenc::Mesh::CRectangleConstantBasis	167
corenc::Mesh::CRectangleHBasis	170
corenc::Mesh::CTriangleBasis	205
corenc::Mesh::CTriangleBasis	205
corenc::Mesh::CTriangleLagrangeBasis	210
corenc::Mesh::CTriangleLinearBasis	218
Methods::CSMethod	198
corenc::CSolution	199
corenc::CFESolution	85
corenc::CVecSolution	233
corenc::solvers::dg_shallow_water< Mesh >	234
corenc::solvers::dg_solver< _Problem, _Mesh, _Result >	236
corenc::solvers::dg_solver_shallow_water	238
corenc::method::DGMethod< Problem, Grid, Matrix >	239
corenc::method::DGMethodZero< Problem, Grid, Matrix >	246
corenc::method::DGSolution< Grid >	253
corenc::solvers::eigen_solver< Matrix, Solver >	254
Algebra::ESolver	255
corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >	261
corenc::solvers::fem_solver< _Problem, _Mesh, _Result >	262
corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >	264
corenc::method::FEMethod< Problem, Grid, Matrix >	267
corenc::method::FEMethodZero< Problem, Grid, Matrix >	274
corenc::method::FVMethod1d	281
corenc::Mesh::Gauss1dim	282
corenc::Mesh::Gauss1dimN< N >	283
corenc::GaussianKernel	284
corenc::GaussianProcess	286
corenc::Mesh::GaussRectangular	288
corenc::Mesh::GaussRectangularCubic	290
corenc::Mesh::GaussTetrahedron	292
corenc::Mesh::GaussTriangle	293
Algebra::Matrix	295
Algebra::MatrixDiag	298
Algebra::MatrixSkyline	300
corenc::multi_vector< T >	304
corenc::Mesh::parameter< T >	306
corenc::Mesh::parameter< double >	306
corenc::Mesh::Point	309
corenc::Mesh::point_source< T >	313
corenc::method::RungeKutta< Problem, Type >	314

corenc::method::STSolution< Grid >	316
corenc::method::system_dg_method< Problem, Grid, Matrix >	318
corenc::method::system_dg_method< Grid, bool, bool >	322
corenc::test_case_elliptic_fem	322
corenc::tests::test_case_rectanglebasis	325
corenc::tests::test_case_regular_mesh	326
corenc::test_case_solver	327
corenc::tests::test_case_trianglebasis	328
corenc::test_cases	329
corenc::test_conv_diff	330
corenc::solvers::vector_solution	332

Chapter 3

Class Index

3.1 Class List

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

corenc::CBurgersScalar	25
corenc::Mesh::CCube	27
corenc::Mesh::CCubeBasis	32
corenc::method::CDGMethod< Type >	36
corenc::method::CDGMethodZero< Type >	37
corenc::CDiffusionScalar	39
corenc::Mesh::CEdge	45
corenc::Mesh::CEdge2ndBasis	49
corenc::Mesh::CEdgeConstantBasis	52
corenc::Mesh::CEdgeHermiteBasis	55
corenc::Mesh::CEdgeLinearBasis	58
corenc::Mesh::CEdgeMultiBasis	61
corenc::Mesh::CElement< T >	64
corenc::Mesh::CElement2D< T >	69
corenc::Mesh::CElement2D< bool >	74
corenc::Mesh::CElement< bool >	78
corenc::method::CFEMethod< Type >	82
corenc::method::CFEMethodZero< Type >	83
corenc::CFESolution	85
corenc::CFEweights	89
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >	90
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >	98
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >	107
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >	115
corenc::CFiniteSolver< Method, Mesh, Solver >	123
corenc::Mesh::CMesh< T >	124
corenc::Mesh::CMesh1D	129
corenc::Mesh::CMesh< bool >	134
corenc::Mesh::CNode	138
corenc::Mesh::CNodeBasis	142
corenc::Mesh::CParameter	144
corenc::CProblem	147
corenc::Mesh::CRectangle	148
corenc::Mesh::CRectangleBasis	154
corenc::Mesh::CRectangleBasis2	157

corenc::Mesh::CRectangleBasis2x	160
corenc::Mesh::CRectangleBasis2y	164
corenc::Mesh::CRectangleConstantBasis	167
corenc::Mesh::CRectangleHBasis	170
corenc::Mesh::CRegularMesh	174
corenc::Mesh::CRegularMesh3D	180
corenc::CShallowWater	187
corenc::Mesh::CShape	192
corenc::Mesh::CShapeFunction< Type >	195
Methods::CSMethod	198
corenc::CSolution	199
corenc::Mesh::CTriangle	199
corenc::Mesh::CTriangleBasis	205
corenc::Mesh::CTriangleLagrangeBasis	210
corenc::Mesh::CTriangleLinear	213
corenc::Mesh::CTriangleLinearBasis	218
corenc::Mesh::CTriangularMesh	221
corenc::Mesh::CTriangularMeshLinear	228
corenc::CVecSolution	233
corenc::solvers::dg_shallow_water< Mesh >	234
corenc::solvers::dg_solver< _Problem, _Mesh, _Result >	236
corenc::solvers::dg_solver_shallow_water	238
corenc::method::DGMethod< Problem, Grid, Matrix >	239
corenc::method::DGMethodZero< Problem, Grid, Matrix >	246
corenc::method::DGSolution< Grid >	253
corenc::solvers::eigen_solver< Matrix, Solver >	254
Algebra::ESolver	255
corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >	261
corenc::solvers::fem_solver< _Problem, _Mesh, _Result >	262
corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >	264
corenc::method::FEMethod< Problem, Grid, Matrix >	267
corenc::method::FEMethodZero< Problem, Grid, Matrix >	274
corenc::method::FVMethod1d	281
corenc::Mesh::Gauss1dim	282
corenc::Mesh::Gauss1dimN< N >	283
corenc::GaussianKernel	284
corenc::GaussianProcess	286
corenc::Mesh::GaussRectangular	288
corenc::Mesh::GaussRectangularCubic	290
corenc::Mesh::GaussTetrahedron	292
corenc::Mesh::GaussTriangle	293
Algebra::Matrix	295
Algebra::MatrixDiag	298
Algebra::MatrixSkyline	300
corenc::multi_vector< T >	304
corenc::Mesh::parameter< T >	306
corenc::Mesh::Point	309
corenc::Mesh::point_source< T >	313
corenc::method::RungeKutta< Problem, Type >	314
corenc::method::STSolution< Grid >	316
corenc::method::system_dg_method< Problem, Grid, Matrix >	318
corenc::method::system_dg_method< Grid, bool, bool >	322
corenc::test_case_elliptic_fem	322
corenc::tests::test_case_rectanglebasis	325
corenc::tests::test_case_regular_mesh	326
corenc::test_case_solver	327
corenc::tests::test_case_trianglebasis	328
corenc::test_cases	329

corenc::test_conv_diff	330
corenc::solvers::vector_solution	332

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

colors.h	333
main.cpp	493
CoreNCA/Matrix.cpp	334
CoreNCA/Matrix.h	334
CoreNCA/MatrixDiag.cpp	335
CoreNCA/MatrixDiag.h	335
CoreNCA/MatrixSkyline.cpp	336
CoreNCA/MatrixSkyline.h	337
CoreNCFEM/FESolution.h	341
CoreNCFEM/FiniteSolver.h	386
CoreNCFEM/GaussianField.h	387
CoreNCFEM/Mesh.h	399
CoreNCFEM/multi_vector.h	486
CoreNCFEM/Parameter.cpp	488
CoreNCFEM/Parameter.h	488
CoreNCFEM/Point.cpp	490
CoreNCFEM/Point.h	490
CoreNCFEM/FiniteElements/CRectangleBasis2x.cpp	343
CoreNCFEM/FiniteElements/Cube.cpp	343
CoreNCFEM/FiniteElements/Cube.h	343
CoreNCFEM/FiniteElements/CubeHBasis.cpp	346
CoreNCFEM/FiniteElements/Edge.cpp	346
CoreNCFEM/FiniteElements/Edge.h	346
CoreNCFEM/FiniteElements/FiniteElement.h	350
CoreNCFEM/FiniteElements/FiniteElement2D.h	363
CoreNCFEM/FiniteElements/Node.cpp	368
CoreNCFEM/FiniteElements/Node.h	368
CoreNCFEM/FiniteElements/Rectangle.cpp	370
CoreNCFEM/FiniteElements/Rectangle.h	370
CoreNCFEM/FiniteElements/RectangleBasis2.cpp	376
CoreNCFEM/FiniteElements/RectangleBasis2y.cpp	376
CoreNCFEM/FiniteElements/RectangleHBasis.cpp	376
CoreNCFEM/FiniteElements/Shape.h	377
CoreNCFEM/FiniteElements/ShapeFunction.h	378
CoreNCFEM/FiniteElements/Triangle.cpp	379

CoreNCFEM/FiniteElements/Triangle.h	380
CoreNCFEM/FiniteElements/TriangleLagrange.cpp	383
CoreNCFEM/FiniteElements/TriangleLinear.cpp	383
CoreNCFEM/FiniteElements/TriangleLinear.h	383
CoreNCFEM/Grids/Mesh1D.cpp	388
CoreNCFEM/Grids/Mesh1D.h	389
CoreNCFEM/Grids/RegularMesh.cpp	390
CoreNCFEM/Grids/RegularMesh.h	391
CoreNCFEM/Grids/RegularMesh3D.cpp	393
CoreNCFEM/Grids/RegularMesh3D.h	393
CoreNCFEM/Grids/TriangularMesh.cpp	395
CoreNCFEM/Grids/TriangularMesh.h	396
CoreNCFEM/Grids/TriangularMeshLinear.cpp	398
CoreNCFEM/Grids/TriangularMeshLinear.h	398
CoreNCFEM/Methods/CSMethod.h	401
CoreNCFEM/Methods/dg_flux.h	401
CoreNCFEM/Methods/DGMethod.h	402
CoreNCFEM/Methods/DGMethodZero.h	421
CoreNCFEM/Methods/DGSolution.h	440
CoreNCFEM/Methods/FEAnalysis.h	442
CoreNCFEM/Methods/FEMethod.h	443
CoreNCFEM/Methods/FEMethodZero.h	458
CoreNCFEM/Methods/FVMMethod.cpp	475
CoreNCFEM/Methods/FVMMethod.h	475
CoreNCFEM/Methods/RungeKutta.h	476
CoreNCFEM/Methods/system_dg_method.h	478
Problems/BurgersScalar.cpp	494
Problems/BurgersScalar.h	494
Problems/DiffusionScalar.cpp	495
Problems/DiffusionScalar.h	495
Problems/Problems.h	496
Problems/ShallowWater.cpp	497
Problems/ShallowWater.h	497
Solvers/dg_solver.h	498
Solvers/dg_solver_shallow_water.cpp	500
Solvers/dg_solver_shallow_water.h	500
Solvers/eigen_solver.h	508
Solvers/fem_solver.h	509
Solvers/fem_solver_lib.h	512
Tests/test_case_elliptic_fem.cpp	517
Tests/test_case_elliptic_fem.h	518
Tests/test_case_regular_mesh.cpp	519
Tests/test_case_regular_mesh.h	519
Tests/test_case_solver.cpp	520
Tests/test_case_solver.h	521
Tests/test_cases.cpp	522
Tests/test_cases.h	522
Tests/test_conv_diff.cpp	523
Tests/test_conv_diff.h	524
Tests/FiniteElements/test_case_rectanglebasis.cpp	515
Tests/FiniteElements/test_case_rectanglebasis.h	515
Tests/FiniteElements/test_case_trianglebasis.cpp	516
Tests/FiniteElements/test_case_trianglebasis.h	516

Chapter 5

Namespace Documentation

5.1 Algebra Namespace Reference

Classes

- class [ESolver](#)
- class [Matrix](#)
- class [MatrixDiag](#)
- class [MatrixSkyline](#)

Enumerations

- enum class [Solvers](#) {
 [BiCGStab](#) , [GMRES](#) , [GMRES_BiCGStab](#) , [Gauss](#) ,
 [PARDISO](#) }

5.1.1 Enumeration Type Documentation

5.1.1.1 Solvers

```
enum class Algebra::Solvers [strong]
```

Enumerator

BiCGStab	
GMRES	
GMRES_BiCGStab	
Gauss	
PARDISO	

5.2 corenc Namespace Reference

Namespaces

- namespace [color](#)
- namespace [Mesh](#)
- namespace [method](#)
- namespace [solvers](#)
- namespace [tests](#)

Classes

- class [CBurgersScalar](#)
- class [CDiffusionScalar](#)
- class [CFESolution](#)
- class [CFEweights](#)
- class [CFiniteSolver](#)
- class [CProblem](#)
- class [CShallowWater](#)
- class [CSolution](#)
- class [CVecSolution](#)
- struct [GaussianKernel](#)
- struct [GaussianProcess](#)
- class [multi_vector](#)
- class [test_case_elliptic_fem](#)
- class [test_case_solver](#)
- class [test_cases](#)
- class [test_conv_diff](#)

Typedefs

- using [scalar_func](#) = std::function< const double(const [Mesh::Point](#) &)>
- using [vector_func](#) = std::function< const [Mesh::Point](#)(const [Mesh::Point](#) &)>

Enumerations

- enum class [Terms](#) {
[IUV](#) , [IDUDV](#) , [IDUV](#) , [IUDV](#) ,
[EUV](#) , [EDUDV](#) , [EDUV](#) , [EUDV](#) ,
[EFV](#) , [RUV](#) , [SUPG](#) }
- enum class [Parameters](#) { [DIFFUSION](#) , [MASS](#) , [ADVECTION](#) }

5.2.1 Detailed Description

Usually it is a `vector<double>` but some methods required different types like vector of vectors or double/tripple values The interface for dealing with solutions; NOT IN USE

5.2.2 Typedef Documentation

5.2.2.1 scalar_func

```
using corenc::scalar_func = typedef std::function<const double(const Mesh::Point&)>
```

5.2.2.2 vector_func

```
using corenc::vector_func = typedef std::function<const Mesh::Point(const Mesh::Point&)>
```

5.2.3 Enumeration Type Documentation

5.2.3.1 Parameters

```
enum class corenc::Parameters [strong]
```

Enumerator

DIFFUSION	
MASS	
ADVECTION	

5.2.3.2 Terms

```
enum class corenc::Terms [strong]
```

Enumerator

IUV	
IDUDV	
IDUV	
IUDV	
EUV	
EDUDV	
EDUV	
EUDV	
EFV	
RUV	
SUPG	

5.3 corenc::color Namespace Reference

Variables

- const std::string **ESCAPE** = "\u001b[0m"
- const std::string **BLACK** = "\u001b[30m"
- const std::string **RED** = "\u001b[31m"
- const std::string **GREEN** = "\u001b[32m"
- const std::string **YELLOW** = "\u001b[33m"
- const std::string **BLUE** = "\u001b[34m"
- const std::string **MAGENTA** = "\u001b[35m"
- const std::string **CYAN** = "\u001b[36m"
- const std::string **WHITE** = "\u001b[37m"
- const std::string **PURPLE** = "\e[1;35m"
- const std::string **BBLACK** = "\u001b[30;1m"
- const std::string **BRED** = "\u001b[31;1m"
- const std::string **BGREEN** = "\u001b[32;1m"
- const std::string **BYELLOW** = "\u001b[33;1m"
- const std::string **BBLUE** = "\u001b[34;1m"
- const std::string **BMAGENTA** = "\u001b[35;1m"
- const std::string **BCYAN** = "\u001b[36;1m"
- const std::string **BWHITE** = "\u001b[37;1m"

5.3.1 Variable Documentation

5.3.1.1 BBLACK

```
const std::string corenc::color::BBLACK = "\u001b[30;1m"
```

5.3.1.2 BBLUE

```
const std::string corenc::color::BBLUE = "\u001b[34;1m"
```

5.3.1.3 BCYAN

```
const std::string corenc::color::BCYAN = "\u001b[36;1m"
```

5.3.1.4 BGREEN

```
const std::string corenc::color::BGREEN = "\u001b[32;1m"
```

5.3.1.5 BLACK

```
const std::string corenc::color::BLACK = "\u001b[30m"
```

5.3.1.6 BLUE

```
const std::string corenc::color::BLUE = "\u001b[34m"
```

5.3.1.7 BMAGENTA

```
const std::string corenc::color::BMAGENTA = "\u001b[35;1m"
```

5.3.1.8 BRED

```
const std::string corenc::color::BRED = "\u001b[31;1m"
```

5.3.1.9 BWHITE

```
const std::string corenc::color::BWHITE = "\u001b[37;1m"
```

5.3.1.10 BYELLOW

```
const std::string corenc::color::BYELLOW = "\u001b[33;1m"
```

5.3.1.11 CYAN

```
const std::string corenc::color::CYAN = "\u001b[36m"
```

5.3.1.12 ESCAPE

```
const std::string corenc::color::ESCAPE = "\u001b[0m"
```

5.3.1.13 GREEN

```
const std::string corenc::color::GREEN = "\u001b[32m"
```

5.3.1.14 MAGENTA

```
const std::string corenc::color::MAGENTA = "\u001b[35m"
```

5.3.1.15 PURPLE

```
const std::string corenc::color::PURPLE = "\e[1;35m"
```

5.3.1.16 RED

```
const std::string corenc::color::RED = "\u001b[31m"
```

5.3.1.17 WHITE

```
const std::string corenc::color::WHITE = "\u001b[37m"
```

5.3.1.18 YELLOW

```
const std::string corenc::color::YELLOW = "\u001b[33m"
```

5.4 corenc::Mesh Namespace Reference

Classes

- class [CCube](#)
- class [CCubeBasis](#)
- class [CEdge](#)
- class [CEdge2ndBasis](#)
- class [CEdgeConstantBasis](#)
- class [CEdgeHermiteBasis](#)
- class [CEdgeLinearBasis](#)
- class [CEdgeMultiBasis](#)
- class [CElement](#)
- class [CElement2D](#)
- class [CElement2D< bool >](#)
- class [CElement< bool >](#)
- class [CFiniteElement](#)
- class [CFiniteElement2D](#)
- class [CFiniteElement< Shape, ShapeFunction, bool, bool >](#)
- class [CFiniteElement< Shape, ShapeFunction, DoF, bool >](#)
- class [CMesh](#)
- class [CMesh1D](#)
- class [CMesh< bool >](#)
- class [CNode](#)
- class [CNodeBasis](#)
- class [CParameter](#)
- class [CRectangle](#)
- class [CRectangleBasis](#)
- class [CRectangleBasis2](#)
- class [CRectangleBasis2x](#)
- class [CRectangleBasis2y](#)
- class [CRectangleConstantBasis](#)
- class [CRectangleHBasis](#)
- class [CRegularMesh](#)
- class [CRegularMesh3D](#)
- class [CShape](#)
- class [CShapeFunction](#)
- class [CTriangle](#)
- class [CTriangleBasis](#)
- class [CTriangleLagrangeBasis](#)
- class [CTriangleLinear](#)
- class [CTriangleLinearBasis](#)
- class [CTriangularMesh](#)
- class [CTriangularMeshLinear](#)
- struct [Gauss1dim](#)
- struct [Gauss1dimN](#)
- struct [GaussRectangular](#)
- struct [GaussRectangularCubic](#)
- struct [GaussTetrahedron](#)
- struct [GaussTriangle](#)
- class [parameter](#)
- class [Point](#)
- class [point_source](#)

Typedefs

- using `function_dp` = `std::function< const double(const Point &)>`

Enumerations

- enum `Elements` {
`Interval` = 0 , `Triangle` = 1 , `Rectangle` = 2 , `Tetrahedron` = 3 ,
`Cube` = 4 }
- enum class `NODES` { `FIRST` , `LAST` }
- enum `Meshes` { `Mesh1D` = 0 , `TriangularMesh` = 1 , `TetrahedralMesh` = 2 }

5.4.1 Typedef Documentation

5.4.1.1 function_dp

```
typedef std::function< const double(const Point &)> corenc::Mesh::function_dp
```

5.4.2 Enumeration Type Documentation

5.4.2.1 Elements

```
enum corenc::Mesh::Elements
```

Enumerator

Interval	
Triangle	
Rectangle	
Tetrahedron	
Cube	

5.4.2.2 Meshes

```
enum corenc::Mesh::Meshes
```

Enumerator

Mesh1D	
TriangularMesh	
TetrahedralMesh	

5.4.2.3 NODES

```
enum class corenc::Mesh::NODES [strong]
```

Enumerator

FIRST	
LAST	

5.5 corenc::method Namespace Reference

Classes

- class [CDGMethod](#)
- class [CDGMethodZero](#)
- class [CFEMethod](#)
- class [CFEMethodZero](#)
- class [DGMethod](#)
- class [DGMethodZero](#)
- class [DGSolution](#)
- class [FEAnalysis](#)
- class [FEMethod](#)
- class [FEMethodZero](#)
- class [FVMethod1d](#)
- class [RungeKutta](#)
- class [STSolution](#)
- class [system_dg_method](#)
- class [system_dg_method](#)< Grid, bool, bool >

Enumerations

- enum class [DGFlux](#) {
[EIP](#) , [EBaumannOden](#) , [EBaumannOdenIP](#) , [ENIPG](#) ,
[EUpwind](#) , [ECentral](#) , [ELaxFriedrichs](#) , [IIP](#) ,
[IBaumannOden](#) , [IBaumannOdenIP](#) , [INIPG](#) , [IUpwind](#) ,
[ICentral](#) , [ILaxFriedrichs](#) , [CUSTOM](#) , [NOFLUX](#) }
- enum class [BoundaryType](#) { [MAIN](#) , [SECOND](#) , [THIRD](#) , [FREE](#) }
- enum class [FVFlux](#) { [LaxFriedrichs](#) , [Upwind](#) , [Central](#) , [NOFLUX](#) }

5.5.1 Enumeration Type Documentation

5.5.1.1 BoundaryType

```
enum class corenc::method::BoundaryType [strong]
```

Enumerator

MAIN	
SECOND	
THIRD	
FREE	

5.5.1.2 DGFlux

```
enum class corenc::method::DGFlux [strong]
```

Enumerator

EIP	
EBaumannOden	
EBaumannOdenIP	
ENIPG	
EUpwind	
ECentral	
ELaxFriedrichs	
IIP	
IBaumannOden	
IBaumannOdenIP	
INIPG	
IUpwind	
ICentral	
ILaxFriedrichs	
CUSTOM	
NOFLUX	

5.5.1.3 FVFlux

```
enum class corenc::method::FVFlux [strong]
```

Enumerator

LaxFriedrichs	
Upwind	
Central	
NOFLUX	

5.6 corenc::solvers Namespace Reference

Classes

- class [dg_shallow_water](#)
- class [dg_solver](#)
- class [dg_solver_shallow_water](#)
- class [eigen_solver](#)
- class [fem_solver](#)
- class [fem_solver_lib](#)
- struct [vector_solution](#)

5.7 corenc::tests Namespace Reference

Classes

- class [test_case_rectanglebasis](#)
- class [test_case_regular_mesh](#)
- class [test_case_trianglebasis](#)

5.8 Methods Namespace Reference

Classes

- class [CSMethod](#)

5.9 wtf Namespace Reference

Functions

- const [Point](#) [mid_point](#) (const [Point](#) &p1, const [Point](#) &p2)
- const [Point](#) [s_point](#) (const [Point](#) &p1, const [Point](#) &p2, const double s)
- const [Point](#) [center_point](#) (const [Point](#) &p1, const [Point](#) &p2, const [Point](#) &p3)

5.9.1 Function Documentation

5.9.1.1 center_point()

```
const Point wtf::center_point (  
    const Point & p1,  
    const Point & p2,  
    const Point & p3 )
```

5.9.1.2 mid_point()

```
const Point wtf::mid_point (
    const Point & p1,
    const Point & p2 )
```

5.9.1.3 s_point()

```
const Point wtf::s_point (
    const Point & p1,
    const Point & p2,
    const double s )
```

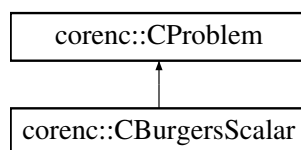
Chapter 6

Class Documentation

6.1 corenc::CBurgersScalar Class Reference

```
#include <BurgersScalar.h>
```

Inheritance diagram for corenc::CBurgersScalar:



Public Member Functions

- [CBurgersScalar](#) ()
- [~CBurgersScalar](#) ()
- [Terms](#) [getTerm](#) (const unsigned int) const
- const unsigned int [getNumberOfTerms](#) () const
- const int [setTerm](#) (const unsigned int, const [Terms](#) &)
- const int [addTerm](#) (const [Terms](#) &)
- const double [getFlux](#) (const double) const
- const int [removeTerm](#) (const [Terms](#) &)
- const int [load_parameters](#) (const std::string &file_name)

6.1.1 Constructor & Destructor Documentation

6.1.1.1 CBurgersScalar()

```
CBurgersScalar::CBurgersScalar ( )
```

6.1.1.2 ~CBurgersScalar()

```
CBurgersScalar::~~CBurgersScalar ( )
```

6.1.2 Member Function Documentation

6.1.2.1 addTerm()

```
const int CBurgersScalar::addTerm (
    const Terms & term ) [virtual]
```

Implements [corenc::CProblem](#).

6.1.2.2 getFlux()

```
const double CBurgersScalar::getFlux (
    const double p ) const
```

6.1.2.3 getNumberOfTerms()

```
const unsigned int CBurgersScalar::getNumberOfTerms ( ) const [virtual]
```

Implements [corenc::CProblem](#).

6.1.2.4 getTerm()

```
Terms CBurgersScalar::getTerm (
    const unsigned int i ) const [virtual]
```

Implements [corenc::CProblem](#).

6.1.2.5 load_parameters()

```
const int CBurgersScalar::load_parameters (
    const std::string & file_name ) [virtual]
```

Implements [corenc::CProblem](#).

6.1.2.6 removeTerm()

```
const int CBurgersScalar::removeTerm (
    const Terms & term )
```

6.1.2.7 setTerm()

```
const int CBurgersScalar::setTerm (
    const unsigned int i,
    const Terms & term ) [virtual]
```

Implements [corenc::CProblem](#).

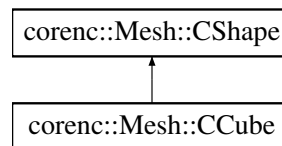
The documentation for this class was generated from the following files:

- Problems/[BurgersScalar.h](#)
- Problems/[BurgersScalar.cpp](#)

6.2 corenc::Mesh::CCube Class Reference

```
#include <Cube.h>
```

Inheritance diagram for corenc::Mesh::CCube:



Public Member Functions

- [CCube](#) ()
- [CCube](#) (const int n1, const int n2, const int n3, const int n4, const int order)
- [CCube](#) (const int n1, const int n2, const int n3, const int n4, const int e1, const int e2, const int e3, const int e4, const int order)
- [CCube](#) (const int *, const int order)
- [CCube](#) (const int *, const int *, const int order)
- [CCube](#) (const [CCube](#) &)
- [CCube](#) & [operator=](#) (const [CCube](#) &t)
- const bool [operator==](#) (const [CCube](#) &t)
- std::istream & [operator>>](#) (std::istream &is)
- [~CCube](#) ()
- const int [GetNode](#) (const int) const
- const int [GetNode](#) (const [NODES](#) &) const
- const int [GetEdge](#) (const int) const
- const int [GetFacet](#) (const int) const
- const int [GetNumberOfNodes](#) () const

- const int [GetNumberOfEdges](#) () const
- const int [GetNumberOfFacets](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- void [SetNode](#) (const int k, const int node)
- const int [IncreaseOrder](#) ()
- const int [SetOrder](#) (const int px, const int py)
- void [SetEdge](#) (const int k, const int edge)
- void [SetFacet](#) (const int k, const int facet)

6.2.1 Constructor & Destructor Documentation

6.2.1.1 CCube() [1/6]

```
CCube::CCube ( )
```

6.2.1.2 CCube() [2/6]

```
CCube::CCube (
    const int n1,
    const int n2,
    const int n3,
    const int n4,
    const int order )
```

6.2.1.3 CCube() [3/6]

```
CCube::CCube (
    const int n1,
    const int n2,
    const int n3,
    const int n4,
    const int e1,
    const int e2,
    const int e3,
    const int e4,
    const int order )
```

6.2.1.4 CCube() [4/6]

```
CCube::CCube (
    const int * nodes,
    const int order )
```

6.2.1.5 CCube() [5/6]

```
CCube::CCube (
    const int * nodes,
    const int * edges,
    const int order )
```

6.2.1.6 CCube() [6/6]

```
CCube::CCube (
    const CCube & t )
```

6.2.1.7 ~CCube()

```
corenc::Mesh::CCube::~~CCube ( ) [inline]
```

6.2.2 Member Function Documentation

6.2.2.1 GetEdge()

```
const int CCube::GetEdge (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.2 GetFacet()

```
const int CCube::GetFacet (
    const int ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.3 GetNode() [1/2]

```
const int CCube::GetNode (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.4 GetNode() [2/2]

```
const int CCube::GetNode (
    const NODES & node ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.5 GetNumberOfEdges()

```
const int CCube::GetNumberOfEdges ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.6 GetNumberOfFacets()

```
const int CCube::GetNumberOfFacets ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.7 GetNumberOfNodes()

```
const int CCube::GetNumberOfNodes ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.8 IncreaseOrder()

```
const int CCube::IncreaseOrder ( )
```


6.2.2.9 Integrate() [1/3]

```
const double CCube::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.2.2.10 Integrate() [2/3]

```
const Point CCube::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.2.2.11 Integrate() [3/3]

```
const vector< double > CCube::Integrate (
    const std::function< const std::vector< double >(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.2.2.12 operator=()

```
CCube & corenc::Mesh::CCube::operator= (
    const CCube & t ) [inline]
```

6.2.2.13 operator==()

```
const bool corenc::Mesh::CCube::operator== (
    const CCube & t ) [inline]
```

6.2.2.14 operator>>()

```
std::istream & corenc::Mesh::CCube::operator>> (
    std::istream & is ) [inline]
```

6.2.2.15 SetEdge()

```
void CCube::SetEdge (
    const int k,
    const int edge ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.16 SetFacet()

```
void CCube::SetFacet (
    const int k,
    const int facet ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.2.2.17 SetNode()

```
void CCube::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.2.2.18 SetOrder()

```
const int CCube::SetOrder (
    const int px,
    const int py )
```

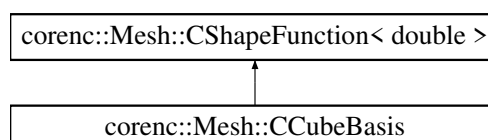
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Cube.h](#)
- [CoreNCFEM/FiniteElements/Cube.cpp](#)

6.3 corenc::Mesh::CCubeBasis Class Reference

```
#include <Cube.h>
```

Inheritance diagram for `corenc::Mesh::CCubeBasis`:



Public Member Functions

- [CCubeBasis](#) ()
- [CCubeBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CCubeBasis](#) (const [Point](#) *, const int order)
- [CCubeBasis](#) (const [CCubeBasis](#) &)
- [CCubeBasis](#) & [operator=](#) (const [CCubeBasis](#) &t)
- [~CCubeBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.3.1 Constructor & Destructor Documentation

6.3.1.1 CCubeBasis() [1/4]

```
CCubeBasis::CCubeBasis ( )
```

6.3.1.2 CCubeBasis() [2/4]

```
CCubeBasis::CCubeBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int order )
```

6.3.1.3 CCubeBasis() [3/4]

```
CCubeBasis::CCubeBasis (
    const Point * p,
    const int order )
```

6.3.1.4 CCubeBasis() [4/4]

```
CCubeBasis::CCubeBasis (
    const CCubeBasis & t )
```

6.3.1.5 ~CCubeBasis()

```
corenc::Mesh::CCubeBasis::~~CCubeBasis ( ) [inline]
```

6.3.2 Member Function Documentation

6.3.2.1 GetGradShapeFunction()

```
const Point CCubeBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.3.2.2 GetMeasure()

```
const double corenc::Mesh::CCubeBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.3.2.3 GetNormal()

```
const Point CCubeBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.3.2.4 GetNumberOfShapeFunctions()

```
const int CCubeBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.3.2.5 GetShapeFunction()

```
const double CCubeBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.3.2.6 GetValue()

```
const double CCubeBasis::GetValue (
    const Point & p ) const
```

6.3.2.7 GetWeight()

```
const double CCubeBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.3.2.8 IncreaseOrder()

```
const int CCubeBasis::IncreaseOrder ( )
```

6.3.2.9 operator=()

```
CCubeBasis & corenc::Mesh::CCubeBasis::operator= (
    const CCubeBasis & t ) [inline]
```

6.3.2.10 ReverseNormal()

```
void CCubeBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

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

- CoreNCFEM/FiniteElements/[Cube.h](#)
- CoreNCFEM/FiniteElements/[Cube.cpp](#)

6.4 corenc::method::CDGMethod< Type > Class Template Reference

```
#include <DGMethod.h>
```

Public Member Functions

- [CDGMethod\(\)](#)
- virtual [~CDGMethod\(\)](#)
- virtual const int [Assemble\(\)](#)=0
- virtual const Type [GetSolution](#) (const std::vector< double > &point) const =0
- virtual const std::vector< Type > [GetSolution](#) () const =0
- virtual const Type [GetMaxSolution](#) () const =0
- virtual const Type [GetMinSolution](#) () const =0

6.4.1 Constructor & Destructor Documentation

6.4.1.1 CDGMethod()

```
template<class Type >
corenc::method::CDGMethod< Type >::CDGMethod ( ) [inline]
```

6.4.1.2 ~CDGMethod()

```
template<class Type >
virtual corenc::method::CDGMethod< Type >::~~CDGMethod ( ) [inline], [virtual]
```

6.4.2 Member Function Documentation

6.4.2.1 Assemble()

```
template<class Type >
virtual const int corenc::method::CDGMethod< Type >::Assemble ( ) [pure virtual]
```

6.4.2.2 GetMaxSolution()

```
template<class Type >
virtual const Type corenc::method::CDGMethod< Type >::GetMaxSolution ( ) const [pure virtual]
```

6.4.2.3 GetMinSolution()

```
template<class Type >
virtual const Type corenc::method::CDGMethod< Type >::GetMinSolution ( ) const [pure virtual]
```

6.4.2.4 GetSolution() [1/2]

```
template<class Type >
virtual const std::vector< Type > corenc::method::CDGMethod< Type >::GetSolution ( ) const
[pure virtual]
```

6.4.2.5 GetSolution() [2/2]

```
template<class Type >
virtual const Type corenc::method::CDGMethod< Type >::GetSolution (
    const std::vector< double > & point ) const [pure virtual]
```

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

- CoreNCFEM/Methods/[DGMethod.h](#)

6.5 corenc::method::CDGMethodZero< Type > Class Template Reference

```
#include <DGMethodZero.h>
```

Public Member Functions

- [CDGMethodZero](#) ()
- virtual [~CDGMethodZero](#) ()
- virtual const int [Assemble](#) ()=0
- virtual const Type [GetSolution](#) (const std::vector< double > &point) const =0
- virtual const std::vector< Type > [GetSolution](#) () const =0
- virtual const Type [GetMaxSolution](#) () const =0
- virtual const Type [GetMinSolution](#) () const =0

6.5.1 Constructor & Destructor Documentation

6.5.1.1 CDGMethodZero()

```
template<class Type >
corenc::method::CDGMethodZero< Type >::CDGMethodZero ( ) [inline]
```

6.5.1.2 ~CDGMethodZero()

```
template<class Type >
virtual corenc::method::CDGMethodZero< Type >::~~CDGMethodZero ( ) [inline], [virtual]
```

6.5.2 Member Function Documentation

6.5.2.1 Assemble()

```
template<class Type >
virtual const int corenc::method::CDGMethodZero< Type >::Assemble ( ) [pure virtual]
```

6.5.2.2 GetMaxSolution()

```
template<class Type >
virtual const Type corenc::method::CDGMethodZero< Type >::GetMaxSolution ( ) const [pure virtual]
```

6.5.2.3 GetMinSolution()

```
template<class Type >
virtual const Type corenc::method::CDGMethodZero< Type >::GetMinSolution ( ) const [pure virtual]
```

6.5.2.4 GetSolution() [1/2]

```
template<class Type >
virtual const std::vector< Type > corenc::method::CDGMethodZero< Type >::GetSolution ( )
const [pure virtual]
```


6.5.2.5 GetSolution() [2/2]

```
template<class Type >
virtual const Type corenc::method::CDGMethodZero< Type >::GetSolution (
    const std::vector< double > & point ) const [pure virtual]
```

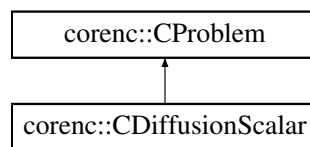
The documentation for this class was generated from the following file:

- CoreNCFEM/Methods/DGMethodZero.h

6.6 corenc::CDiffusionScalar Class Reference

```
#include <DiffusionScalar.h>
```

Inheritance diagram for corenc::CDiffusionScalar:



Public Member Functions

- [CDiffusionScalar](#) ()
- [~CDiffusionScalar](#) ()
- [Terms](#) [getTerm](#) (const unsigned int) const
- const unsigned int [getNumberOfTerms](#) () const
- const int [findTerm](#) (const [Terms](#) &) const
- const int [setTerm](#) (const unsigned int, const [Terms](#) &)
- const int [addTerm](#) (const [Terms](#) &)
- const int [removeTerm](#) (const [Terms](#) &)
- const int [load_parameters](#) (const std::string &file_name)
- const double [get_parameter](#) (const [Terms](#) &, const int element_type, const [Mesh::Point](#) &) const
- const double [get_parameter](#) (const [Terms](#) &, const int element_number, const int element_type, const [Mesh::Point](#) &) const
- const [Mesh::Point](#) [get_parameter](#) (const [Terms](#) &, const int element_number, const int element_type, const [Mesh::Point](#) &, const int) const
- const double [get_parameter](#) (const [Terms](#) &, const int element_type, const int element_number, const int node, const [Mesh::Point](#) &) const
- const [Mesh::Point](#) [get_parameter](#) (const [Terms](#) &, const int element_type, const int element_number, const int node, const [Mesh::Point](#) &, const int v) const
- const double [get_boundary_parameter](#) (const int type, const int element_type, const [Mesh::Point](#) &) const
- const double [get_boundary_parameter](#) (const int type, const int element_type, const int element_number, const [Mesh::Point](#) &) const
- const double [get_boundary_parameter](#) (const int type, const int element_type, const int element_number, const int node, const [Mesh::Point](#) &) const
- const int [get_number_of_boundaries](#) () const
- const int [get_boundary_type](#) (const int number) const
- const int [add_parameter](#) (const [Terms](#) &, const int element_type, const double &value)
- const int [add_parameter](#) (const [Terms](#) &, const int element_type, const [Mesh::parameter](#)< double > &value)

- `const int add_parameter (const Terms &, const int element_type, const Mesh::parameter< Mesh::Point > &value)`
- `const int set_parameter (const Terms &, const int element_type, const Mesh::parameter< double > &value)`
- `const int set_parameter (const Terms &, const int element_type, const Mesh::parameter< Mesh::Point > &value)`
- `const int set_boundary_parameter (const int type, const int element_type, const boundary &value)`
- `const int add_boundary_parameter (const int type, const int element_type, const Mesh::parameter< double > &value)`
- `const int add_boundary_parameter (const int element_type, const Mesh::parameter< double > &value, const Mesh::parameter< double > &value2)`
- `const Mesh::point_source< double > get_point_source (const int number) const`
- `void set_point_source (const int number, const Mesh::point_source< double > &)`
- `const int get_total_sources () const`

6.6.1 Constructor & Destructor Documentation

6.6.1.1 CDiffusionScalar()

```
CDiffusionScalar::CDiffusionScalar ( )
```

6.6.1.2 ~CDiffusionScalar()

```
CDiffusionScalar::~~CDiffusionScalar ( )
```

6.6.2 Member Function Documentation

6.6.2.1 add_boundary_parameter() [1/2]

```
const int CDiffusionScalar::add_boundary_parameter (
    const int element_type,
    const Mesh::parameter< double > & value,
    const Mesh::parameter< double > & value2 )
```

6.6.2.2 add_boundary_parameter() [2/2]

```
const int CDiffusionScalar::add_boundary_parameter (
    const int type,
    const int element_type,
    const Mesh::parameter< double > & value )
```

6.6.2.3 add_parameter() [1/3]

```
const int CDiffusionScalar::add_parameter (
    const Terms & term,
    const int element_type,
    const double & value )
```

6.6.2.4 add_parameter() [2/3]

```
const int CDiffusionScalar::add_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::parameter< double > & value )
```

6.6.2.5 add_parameter() [3/3]

```
const int CDiffusionScalar::add_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::parameter< Mesh::Point > & value )
```

6.6.2.6 addTerm()

```
const int CDiffusionScalar::addTerm (
    const Terms & term ) [virtual]
```

Implements [corenc::CProblem](#).

6.6.2.7 findTerm()

```
const int CDiffusionScalar::findTerm (
    const Terms & term ) const
```

6.6.2.8 get_boundary_parameter() [1/3]

```
const double CDiffusionScalar::get_boundary_parameter (
    const int type,
    const int element_type,
    const int element_number,
    const int node,
    const Mesh::Point & p ) const
```

6.6.2.9 get_boundary_parameter() [2/3]

```
const double CDiffusionScalar::get_boundary_parameter (
    const int type,
    const int element_type,
    const int element_number,
    const Mesh::Point & p ) const
```

6.6.2.10 get_boundary_parameter() [3/3]

```
const double CDiffusionScalar::get_boundary_parameter (
    const int type,
    const int element_type,
    const Mesh::Point & p ) const
```

6.6.2.11 get_boundary_type()

```
const int CDiffusionScalar::get_boundary_type (
    const int number ) const
```

6.6.2.12 get_number_of_boundaries()

```
const int CDiffusionScalar::get_number_of_boundaries ( ) const
```

6.6.2.13 get_parameter() [1/5]

```
const double CDiffusionScalar::get_parameter (
    const Terms & term,
    const int element_number,
    const int element_type,
    const Mesh::Point & p ) const
```

6.6.2.14 get_parameter() [2/5]

```
const Mesh::Point CDiffusionScalar::get_parameter (
    const Terms & term,
    const int element_number,
    const int element_type,
    const Mesh::Point & p,
    const int ) const
```

6.6.2.15 get_parameter() [3/5]

```
const double CDiffusionScalar::get_parameter (
    const Terms & term,
    const int element_type,
    const int element_number,
    const int node,
    const Mesh::Point & p ) const
```

6.6.2.16 get_parameter() [4/5]

```
const Mesh::Point CDiffusionScalar::get_parameter (
    const Terms & term,
    const int element_type,
    const int element_number,
    const int node,
    const Mesh::Point & p,
    const int v ) const
```

6.6.2.17 get_parameter() [5/5]

```
const double CDiffusionScalar::get_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::Point & p ) const
```

6.6.2.18 get_point_source()

```
const Mesh::point_source< double > CDiffusionScalar::get_point_source (
    const int number ) const
```

6.6.2.19 get_total_sources()

```
const int CDiffusionScalar::get_total_sources ( ) const
```

6.6.2.20 getNumberOfTerms()

```
const unsigned int CDiffusionScalar::getNumberOfTerms ( ) const [virtual]
```

Implements [corenc::CProblem](#).

6.6.2.21 `getTerm()`

```
Terms CDiffusionScalar::getTerm (
    const unsigned int i ) const [virtual]
```

Implements [corenc::CProblem](#).

6.6.2.22 `load_parameters()`

```
const int CDiffusionScalar::load_parameters (
    const std::string & file_name ) [virtual]
```

Implements [corenc::CProblem](#).

6.6.2.23 `removeTerm()`

```
const int CDiffusionScalar::removeTerm (
    const Terms & term )
```

6.6.2.24 `set_boundary_parameter()`

```
const int CDiffusionScalar::set_boundary_parameter (
    const int type,
    const int element_type,
    const boundary & value )
```

6.6.2.25 `set_parameter()` [1/2]

```
const int CDiffusionScalar::set_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::parameter< double > & value )
```

6.6.2.26 `set_parameter()` [2/2]

```
const int CDiffusionScalar::set_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::parameter< Mesh::Point > & value )
```

6.6.2.27 set_point_source()

```
void CDiffusionScalar::set_point_source (
    const int number,
    const Mesh::point_source< double > & srs )
```

6.6.2.28 setTerm()

```
const int CDiffusionScalar::setTerm (
    const unsigned int i,
    const Terms & term ) [virtual]
```

Implements [corenc::CProblem](#).

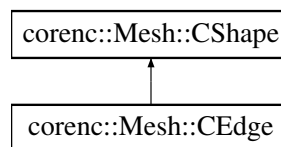
The documentation for this class was generated from the following files:

- Problems/[DiffusionScalar.h](#)
- Problems/[DiffusionScalar.cpp](#)

6.7 corenc::Mesh::CEdge Class Reference

```
#include <Edge.h>
```

Inheritance diagram for corenc::Mesh::CEdge:



Public Member Functions

- [CEdge](#) ()
- [CEdge](#) (const [CEdge](#) &)
- [CEdge](#) (const int n1, const int n2)
- [CEdge](#) (const int *)
- [CEdge](#) & [operator=](#) (const [CEdge](#) &e)
- [~CEdge](#) ()
- const int [GetNode](#) (const int) const
- const int [GetNode](#) (const [NODES](#) &) const
- const int [GetNumberOfNodes](#) () const
- void [SetNode](#) (const int k, const int node)
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const int [IncreaseOrder](#) ()
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const

Friends

- const bool `operator==` (const `CEdge` &e1, const `CEdge` &e2)
- std::istream & `operator>>` (std::istream &is, `CEdge` &e)

6.7.1 Constructor & Destructor Documentation

6.7.1.1 `CEdge()` [1/4]

```
CEdge::CEdge ( )
```

6.7.1.2 `CEdge()` [2/4]

```
CEdge::CEdge (
    const CEdge & e )
```

6.7.1.3 `CEdge()` [3/4]

```
CEdge::CEdge (
    const int n1,
    const int n2 )
```

6.7.1.4 `CEdge()` [4/4]

```
CEdge::CEdge (
    const int * n )
```

6.7.1.5 `~CEdge()`

```
corenc::Mesh::CEdge::~~CEdge ( ) [inline]
```

6.7.2 Member Function Documentation

6.7.2.1 GetNode() [1/2]

```
const int CEdge::GetNode (
    const int k ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.7.2.2 GetNode() [2/2]

```
const int CEdge::GetNode (
    const NODES & node ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.7.2.3 GetNumberOfNodes()

```
const int CEdge::GetNumberOfNodes ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.7.2.4 IncreaseOrder()

```
const int CEdge::IncreaseOrder ( )
```

6.7.2.5 Integrate() [1/3]

```
const double CEdge::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.7.2.6 Integrate() [2/3]

```
const Point CEdge::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.7.2.7 `Integrate()` [3/3]

```
const std::vector< double > corenc::Mesh::CEdge::Integrate (
    const std::function< const std::vector< double >(const Point &)> & ,
    const std::vector< Point > & ) const [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.7.2.8 `operator=()`

```
CEdge & corenc::Mesh::CEdge::operator= (
    const CEdge & e ) [inline]
```

6.7.2.9 `SetNode()`

```
void CEdge::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.7.3 Friends And Related Function Documentation

6.7.3.1 `operator==`

```
const bool operator== (
    const CEdge & e1,
    const CEdge & e2 ) [friend]
```

6.7.3.2 `operator>>`

```
std::istream & operator>> (
    std::istream & is,
    CEdge & e ) [friend]
```

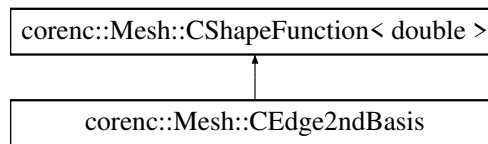
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Edge.h](#)
- [CoreNCFEM/FiniteElements/Edge.cpp](#)

6.8 corenc::Mesh::CEdge2ndBasis Class Reference

```
#include <Edge.h>
```

Inheritance diagram for corenc::Mesh::CEdge2ndBasis:



Public Member Functions

- [CEdge2ndBasis](#) ()
- [CEdge2ndBasis](#) (const [Point](#) &, const [Point](#) &)
- [CEdge2ndBasis](#) (const [Point](#) *)
- [CEdge2ndBasis](#) (const [CEdge2ndBasis](#) &)
- [CEdge2ndBasis](#) & [operator=](#) (const [CEdge2ndBasis](#) &e)
- [~CEdge2ndBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int node, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const

6.8.1 Constructor & Destructor Documentation

6.8.1.1 CEdge2ndBasis() [1/4]

```
CEdge2ndBasis::CEdge2ndBasis ( )
```

6.8.1.2 CEdge2ndBasis() [2/4]

```
CEdge2ndBasis::CEdge2ndBasis (
    const Point & p0,
    const Point & p1 )
```

6.8.1.3 CEdge2ndBasis() [3/4]

```
CEdge2ndBasis::CEdge2ndBasis (
    const Point * p )
```

6.8.1.4 CEdge2ndBasis() [4/4]

```
CEdge2ndBasis::CEdge2ndBasis (
    const CEdge2ndBasis & e )
```

6.8.1.5 ~CEdge2ndBasis()

```
corenc::Mesh::CEdge2ndBasis::~~CEdge2ndBasis ( ) [inline]
```

6.8.2 Member Function Documentation

6.8.2.1 GetGradShapeFunction()

```
const Point CEdge2ndBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.8.2.2 GetMeasure()

```
const double corenc::Mesh::CEdge2ndBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.8.2.3 GetNormal()

```
const Point CEdge2ndBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.8.2.4 GetNumberOfShapeFunctions()

```
const int CEdge2ndBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.8.2.5 GetShapeFunction()

```
const double CEdge2ndBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.8.2.6 GetWeight()

```
const double corenc::Mesh::CEdge2ndBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [inline]
```

6.8.2.7 IncreaseOrder()

```
const int corenc::Mesh::CEdge2ndBasis::IncreaseOrder ( ) [inline]
```

6.8.2.8 operator=()

```
CEdge2ndBasis & corenc::Mesh::CEdge2ndBasis::operator= (
    const CEdge2ndBasis & e ) [inline]
```

6.8.2.9 ReverseNormal()

```
void CEdge2ndBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

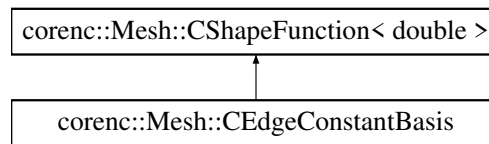
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Edge.h](#)
- [CoreNCFEM/FiniteElements/Edge.cpp](#)

6.9 corenc::Mesh::CEdgeConstantBasis Class Reference

```
#include <Edge.h>
```

Inheritance diagram for corenc::Mesh::CEdgeConstantBasis:



Public Member Functions

- [CEdgeConstantBasis](#) ()
- [CEdgeConstantBasis](#) (const [Point](#) &, const [Point](#) &)
- [CEdgeConstantBasis](#) (const [Point](#) *)
- [CEdgeConstantBasis](#) (const [CEdgeConstantBasis](#) &)
- [CEdgeConstantBasis](#) & operator= (const [CEdgeConstantBasis](#) &e)
- [~CEdgeConstantBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int node, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const

6.9.1 Constructor & Destructor Documentation

6.9.1.1 CEdgeConstantBasis() [1/4]

```
CEdgeConstantBasis::CEdgeConstantBasis ( )
```

6.9.1.2 CEdgeConstantBasis() [2/4]

```
CEdgeConstantBasis::CEdgeConstantBasis (
    const Point & p0,
    const Point & p1 )
```

6.9.1.3 CEdgeConstantBasis() [3/4]

```
CEdgeConstantBasis::CEdgeConstantBasis (
    const Point * p )
```

6.9.1.4 CEdgeConstantBasis() [4/4]

```
CEdgeConstantBasis::CEdgeConstantBasis (
    const CEdgeConstantBasis & e )
```

6.9.1.5 ~CEdgeConstantBasis()

```
corenc::Mesh::CEdgeConstantBasis::~~CEdgeConstantBasis ( ) [inline]
```

6.9.2 Member Function Documentation

6.9.2.1 GetGradShapeFunction()

```
const Point CEdgeConstantBasis::GetGradShapeFunction (
    const int ,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.9.2.2 GetMeasure()

```
const double corenc::Mesh::CEdgeConstantBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.9.2.3 GetNormal()

```
const Point CEdgeConstantBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.9.2.4 GetNumberOfShapeFunctions()

```
const int CEdgeConstantBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.9.2.5 GetShapeFunction()

```
const double CEdgeConstantBasis::GetShapeFunction (
    const int ,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.9.2.6 GetWeight()

```
const double corenc::Mesh::CEdgeConstantBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [inline]
```

6.9.2.7 IncreaseOrder()

```
const int corenc::Mesh::CEdgeConstantBasis::IncreaseOrder ( ) [inline]
```

6.9.2.8 operator=()

```
CEdgeConstantBasis & corenc::Mesh::CEdgeConstantBasis::operator= (
    const CEdgeConstantBasis & e ) [inline]
```

6.9.2.9 ReverseNormal()

```
void CEdgeConstantBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

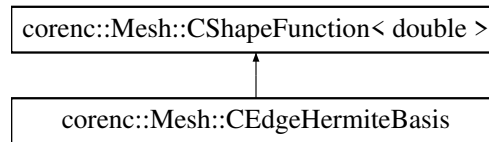
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Edge.h](#)
- [CoreNCFEM/FiniteElements/Edge.cpp](#)

6.10 corenc::Mesh::CEdgeHermiteBasis Class Reference

```
#include <Edge.h>
```

Inheritance diagram for corenc::Mesh::CEdgeHermiteBasis:



Public Member Functions

- [CEdgeHermiteBasis](#) ()
- [CEdgeHermiteBasis](#) (const [Point](#) &, const [Point](#) &)
- [CEdgeHermiteBasis](#) (const [Point](#) *)
- [CEdgeHermiteBasis](#) (const [CEdgeHermiteBasis](#) &)
- [CEdgeHermiteBasis](#) & operator= (const [CEdgeHermiteBasis](#) &e)
- [~CEdgeHermiteBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int node, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- const double [GetMeasure](#) () const

6.10.1 Constructor & Destructor Documentation

6.10.1.1 CEdgeHermiteBasis() [1/4]

```
CEdgeHermiteBasis::CEdgeHermiteBasis ( )
```

6.10.1.2 CEdgeHermiteBasis() [2/4]

```
CEdgeHermiteBasis::CEdgeHermiteBasis (
    const Point & p0,
    const Point & p1 )
```

6.10.1.3 CEdgeHermiteBasis() [3/4]

```
CEdgeHermiteBasis::CEdgeHermiteBasis (
    const Point * p )
```

6.10.1.4 CEdgeHermiteBasis() [4/4]

```
CEdgeHermiteBasis::CEdgeHermiteBasis (
    const CEdgeHermiteBasis & e )
```

6.10.1.5 ~CEdgeHermiteBasis()

```
corenc::Mesh::CEdgeHermiteBasis::~~CEdgeHermiteBasis ( ) [inline]
```

6.10.2 Member Function Documentation

6.10.2.1 GetGradShapeFunction()

```
const Point CEdgeHermiteBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.10.2.2 GetMeasure()

```
const double corenc::Mesh::CEdgeHermiteBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.10.2.3 GetNormal()

```
const Point CEdgeHermiteBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.10.2.4 GetNumberOfShapeFunctions()

```
const int CEdgeHermiteBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.10.2.5 GetShapeFunction()

```
const double CEdgeHermiteBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.10.2.6 GetWeight()

```
const double corenc::Mesh::CEdgeHermiteBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [inline]
```

6.10.2.7 IncreaseOrder()

```
const int corenc::Mesh::CEdgeHermiteBasis::IncreaseOrder ( ) [inline]
```

6.10.2.8 operator=()

```
CEdgeHermiteBasis & corenc::Mesh::CEdgeHermiteBasis::operator= (
    const CEdgeHermiteBasis & e ) [inline]
```

6.10.2.9 ReverseNormal()

```
void CEdgeHermiteBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

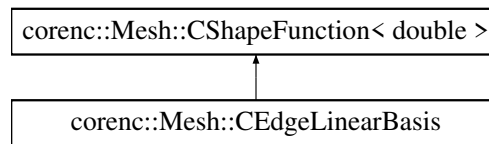
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Edge.h](#)
- [CoreNCFEM/FiniteElements/Edge.cpp](#)

6.11 corenc::Mesh::CEdgeLinearBasis Class Reference

```
#include <Edge.h>
```

Inheritance diagram for corenc::Mesh::CEdgeLinearBasis:



Public Member Functions

- [CEdgeLinearBasis](#) ()
- [CEdgeLinearBasis](#) (const [Point](#) &, const [Point](#) &)
- [CEdgeLinearBasis](#) (const [Point](#) *)
- [CEdgeLinearBasis](#) (const [CEdgeLinearBasis](#) &)
- [CEdgeLinearBasis](#) & operator= (const [CEdgeLinearBasis](#) &e)
- [~CEdgeLinearBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.11.1 Constructor & Destructor Documentation

6.11.1.1 CEdgeLinearBasis() [1/4]

```
CEdgeLinearBasis::CEdgeLinearBasis ( )
```

6.11.1.2 CEdgeLinearBasis() [2/4]

```
CEdgeLinearBasis::CEdgeLinearBasis (
    const Point & p0,
    const Point & p1 )
```

6.11.1.3 CEdgeLinearBasis() [3/4]

```
CEdgeLinearBasis::CEdgeLinearBasis (
    const Point * p )
```

6.11.1.4 CEdgeLinearBasis() [4/4]

```
CEdgeLinearBasis::CEdgeLinearBasis (
    const CEdgeLinearBasis & e )
```

6.11.1.5 ~CEdgeLinearBasis()

```
corenc::Mesh::CEdgeLinearBasis::~~CEdgeLinearBasis ( ) [inline]
```

6.11.2 Member Function Documentation**6.11.2.1 GetGradShapeFunction()**

```
const Point CEdgeLinearBasis::GetGradShapeFunction (
    const int k,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.11.2.2 GetMeasure()

```
const double corenc::Mesh::CEdgeLinearBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.11.2.3 GetNormal()

```
const Point CEdgeLinearBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.11.2.4 GetNumberOfShapeFunctions()

```
const int CEdgeLinearBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.11.2.5 GetShapeFunction()

```
const double CEdgeLinearBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.11.2.6 GetWeight()

```
const double CEdgeLinearBasis::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.11.2.7 IncreaseOrder()

```
const int CEdgeLinearBasis::IncreaseOrder ( )
```

6.11.2.8 operator=()

```
CEdgeLinearBasis & corenc::Mesh::CEdgeLinearBasis::operator= (
    const CEdgeLinearBasis & e ) [inline]
```

6.11.2.9 ReverseNormal()

```
void CEdgeLinearBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

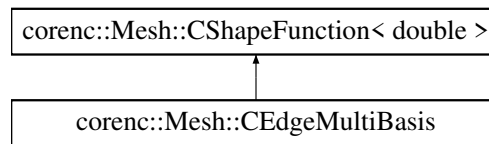
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Edge.h](#)
- [CoreNCFEM/FiniteElements/Edge.cpp](#)

6.12 corenc::Mesh::CEdgeMultiBasis Class Reference

```
#include <Edge.h>
```

Inheritance diagram for corenc::Mesh::CEdgeMultiBasis:



Public Member Functions

- [CEdgeMultiBasis](#) ()
- [CEdgeMultiBasis](#) (const [Point](#) &, const [Point](#) &)
- [CEdgeMultiBasis](#) (const [Point](#) *)
- [CEdgeMultiBasis](#) (const [CEdgeMultiBasis](#) &)
- [CEdgeMultiBasis](#) & [operator=](#) (const [CEdgeMultiBasis](#) &e)
- [~CEdgeMultiBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int node, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const

6.12.1 Constructor & Destructor Documentation

6.12.1.1 CEdgeMultiBasis() [1/4]

```
CEdgeMultiBasis::CEdgeMultiBasis ( )
```

6.12.1.2 CEdgeMultiBasis() [2/4]

```
CEdgeMultiBasis::CEdgeMultiBasis (
    const Point & p0,
    const Point & p1 )
```

6.12.1.3 CEdgeMultiBasis() [3/4]

```
CEdgeMultiBasis::CEdgeMultiBasis (
    const Point * p )
```

6.12.1.4 CEdgeMultiBasis() [4/4]

```
CEdgeMultiBasis::CEdgeMultiBasis (
    const CEdgeMultiBasis & e )
```

6.12.1.5 ~CEdgeMultiBasis()

```
corenc::Mesh::CEdgeMultiBasis::~~CEdgeMultiBasis ( ) [inline]
```

6.12.2 Member Function Documentation

6.12.2.1 GetGradShapeFunction()

```
const Point CEdgeMultiBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.12.2.2 GetMeasure()

```
const double corenc::Mesh::CEdgeMultiBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.12.2.3 GetNormal()

```
const Point CEdgeMultiBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.12.2.4 GetNumberOfShapeFunctions()

```
const int CEdgeMultiBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.12.2.5 GetShapeFunction()

```
const double CEdgeMultiBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.12.2.6 GetWeight()

```
const double corenc::Mesh::CEdgeMultiBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [inline]
```

6.12.2.7 IncreaseOrder()

```
const int corenc::Mesh::CEdgeMultiBasis::IncreaseOrder ( ) [inline]
```

6.12.2.8 operator=()

```
CEdgeMultiBasis & corenc::Mesh::CEdgeMultiBasis::operator= (
    const CEdgeMultiBasis & e ) [inline]
```

6.12.2.9 ReverseNormal()

```
void CEdgeMultiBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

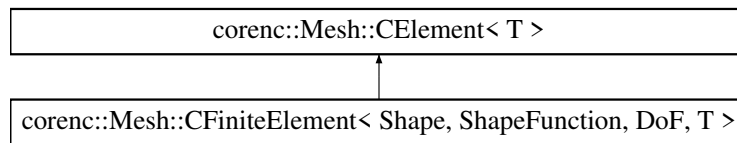
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Edge.h](#)
- [CoreNCFEM/FiniteElements/Edge.cpp](#)

6.13 corenc::Mesh::CElement< T > Class Template Reference

```
#include <FiniteElement.h>
```

Inheritance diagram for corenc::Mesh::CElement< T >:



Public Member Functions

- [CElement](#) ()
- virtual [~CElement](#) ()
- virtual const int [GetType](#) () const =0
- virtual [CElement](#) * [Clone](#) () const =0
- virtual const int [GetDoFs](#) () const =0
- virtual const int [GetNode](#) (const int) const =0
- virtual const int [GetNeighbour](#) (const int) const =0
- virtual void [SetNeighbour](#) (const int k, const int elem)=0
- virtual void [SetType](#) (const int)=0
- virtual void [SetNode](#) (const int, const int)=0
- virtual const int [GetNumberOfNodes](#) () const =0
- virtual const double [GetShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetNormal](#) () const =0
- virtual void [ReverseNormal](#) ()=0
- virtual const int [IncreaseOrder](#) ()=0
- virtual const double [Integrate](#) (const std::function< const double(const [Point](#) &> &, const std::vector< [Point](#) > &v) const =0
- virtual const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &> &, const std::vector< [Point](#) > &v) const =0
- virtual const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &> &, const std::vector< [Point](#) > &) const =0
- virtual const double [GetMeasure](#) () const =0
- virtual const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &> &f) const =0

6.13.1 Constructor & Destructor Documentation

6.13.1.1 CElement()

```
template<class T >
corenc::Mesh::CElement< T >::CElement ( ) [inline]
```

6.13.1.2 ~CElement()

```
template<class T >
virtual corenc::Mesh::CElement< T >::~~CElement ( ) [inline], [virtual]
```

6.13.2 Member Function Documentation

6.13.2.1 Clone()

```
template<class T >
virtual CElement * corenc::Mesh::CElement< T >::Clone ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.2 GetDoFs()

```
template<class T >
virtual const int corenc::Mesh::CElement< T >::GetDoFs ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.3 GetGradShapeFunction()

```
template<class T >
virtual const Point corenc::Mesh::CElement< T >::GetGradShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.4 GetMeasure()

```
template<class T >
virtual const double corenc::Mesh::CElement< T >::GetMeasure ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.5 GetNeighbour()

```
template<class T >
virtual const int corenc::Mesh::CElement< T >::GetNeighbour (
    const int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.6 GetNode()

```
template<class T >
virtual const int corenc::Mesh::CElement< T >::GetNode (
    const int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.7 GetNormal()

```
template<class T >
virtual const Point corenc::Mesh::CElement< T >::GetNormal ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.8 GetNumberOfNodes()

```
template<class T >
virtual const int corenc::Mesh::CElement< T >::GetNumberOfNodes ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.9 GetShapeFunction()

```
template<class T >
virtual const double corenc::Mesh::CElement< T >::GetShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.10 GetType()

```
template<class T >
virtual const int corenc::Mesh::CElement< T >::GetType ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.11 GetWeight()

```
template<class T >
virtual const double corenc::Mesh::CElement< T >::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.12 IncreaseOrder()

```
template<class T >
virtual const int corenc::Mesh::CElement< T >::IncreaseOrder ( ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.13 Integrate() [1/3]

```
template<class T >
virtual const double corenc::Mesh::CElement< T >::Integrate (
    const std::function< const double(const Point &)> & ,
    const std::vector< Point > & v ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.14 Integrate() [2/3]

```
template<class T >
virtual const Point corenc::Mesh::CElement< T >::Integrate (
    const std::function< const Point(const Point &)> & ,
    const std::vector< Point > & v ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.15 Integrate() [3/3]

```
template<class T >
virtual const std::vector< double > corenc::Mesh::CElement< T >::Integrate (
    const std::function< const std::vector< double >(const Point &)> & ,
    const std::vector< Point > & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.16 ReverseNormal()

```
template<class T >
virtual void corenc::Mesh::CElement< T >::ReverseNormal ( ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.17 SetNeighbour()

```
template<class T >
virtual void corenc::Mesh::CElement< T >::SetNeighbour (
    const int k,
    const int elem ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.18 SetNode()

```
template<class T >
virtual void corenc::Mesh::CElement< T >::SetNode (
    const int ,
    const int ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

6.13.2.19 SetType()

```
template<class T >
virtual void corenc::Mesh::CElement< T >::SetType (
    const int ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#), [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >](#) and [corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >](#).

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

- CoreNCFEM/FiniteElements/[FiniteElement.h](#)

6.14 corenc::Mesh::CElement2D< T > Class Template Reference

```
#include <FiniteElement2D.h>
```

Public Member Functions

- [CElement2D](#) ()
- virtual [~CElement2D](#) ()
- virtual const int [GetType](#) () const =0
- virtual [CElement2D](#) * [Clone](#) () const =0
- virtual const int [GetDoFs](#) () const =0
- virtual const int [GetNode](#) (const int) const =0
- virtual const int [GetNeighbour](#) (const int) const =0
- virtual void [SetNeighbour](#) (const int k, const int elem)=0
- virtual void [SetType](#) (const int)=0
- virtual void [SetNode](#) (const int, const int)=0
- virtual const int [GetNumberOfNodes](#) () const =0
- virtual const double [GetShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetNormal](#) () const =0
- virtual void [ReverseNormal](#) ()=0
- virtual const int [IncreaseOrder](#) ()=0
- virtual const int [SetOrder](#) (const int px, const int py)=0
- virtual const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const =0
- virtual const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const =0
- virtual const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const =0
- virtual const double [GetMeasure](#) () const =0
- virtual const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const =0

6.14.1 Constructor & Destructor Documentation

6.14.1.1 CElement2D()

```
template<class T >
corenc::Mesh::CElement2D< T >::CElement2D ( ) [inline]
```

6.14.1.2 ~CElement2D()

```
template<class T >
virtual corenc::Mesh::CElement2D< T >::~~CElement2D ( ) [inline], [virtual]
```

6.14.2 Member Function Documentation

6.14.2.1 Clone()

```
template<class T >
virtual CElement2D * corenc::Mesh::CElement2D< T >::Clone ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.2 GetDoFs()

```
template<class T >
virtual const int corenc::Mesh::CElement2D< T >::GetDoFs ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.3 GetGradShapeFunction()

```
template<class T >
virtual const Point corenc::Mesh::CElement2D< T >::GetGradShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.4 GetMeasure()

```
template<class T >
virtual const double corenc::Mesh::CElement2D< T >::GetMeasure ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.5 GetNeighbour()

```
template<class T >
virtual const int corenc::Mesh::CElement2D< T >::GetNeighbour (
    const int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.6 GetNode()

```
template<class T >
virtual const int corenc::Mesh::CElement2D< T >::GetNode (
    const int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.7 GetNormal()

```
template<class T >
virtual const Point corenc::Mesh::CElement2D< T >::GetNormal ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.8 GetNumberOfNodes()

```
template<class T >
virtual const int corenc::Mesh::CElement2D< T >::GetNumberOfNodes ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.9 GetShapeFunction()

```
template<class T >
virtual const double corenc::Mesh::CElement2D< T >::GetShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.10 GetType()

```
template<class T >
virtual const int corenc::Mesh::CElement2D< T >::GetType ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.11 GetWeight()

```
template<class T >
virtual const double corenc::Mesh::CElement2D< T >::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.12 IncreaseOrder()

```
template<class T >
virtual const int corenc::Mesh::CElement2D< T >::IncreaseOrder ( ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.13 Integrate() [1/3]

```
template<class T >
virtual const double corenc::Mesh::CElement2D< T >::Integrate (
    const std::function< const double(const Point &)> & ,
    const std::vector< Point > & v ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.14 Integrate() [2/3]

```
template<class T >
virtual const Point corenc::Mesh::CElement2D< T >::Integrate (
    const std::function< const Point (const Point &)> & ,
    const std::vector< Point > & v ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.15 Integrate() [3/3]

```
template<class T >
virtual const std::vector< double > corenc::Mesh::CElement2D< T >::Integrate (
    const std::function< const std::vector< double > (const Point &)> & ,
    const std::vector< Point > & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.16 ReverseNormal()

```
template<class T >
virtual void corenc::Mesh::CElement2D< T >::ReverseNormal ( ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.17 SetNeighbour()

```
template<class T >
virtual void corenc::Mesh::CElement2D< T >::SetNeighbour (
    const int k,
    const int elem ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.18 SetNode()

```
template<class T >
virtual void corenc::Mesh::CElement2D< T >::SetNode (
    const int ,
    const int ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.19 SetOrder()

```
template<class T >
virtual const int corenc::Mesh::CElement2D< T >::SetOrder (
    const int px,
    const int py ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

6.14.2.20 SetType()

```
template<class T >
virtual void corenc::Mesh::CElement2D< T >::SetType (
    const int ) [pure virtual]
```

Implemented in [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#).

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

- [CoreNCFEM/FiniteElements/FiniteElement2D.h](#)

6.15 [corenc::Mesh::CElement2D](#)< bool > Class Reference

```
#include <FiniteElement2D.h>
```

Public Member Functions

- [CElement2D](#) ()
- virtual [~CElement2D](#) ()
- virtual const int [GetType](#) () const =0
- virtual [CElement2D](#) * [Clone](#) () const =0
- virtual const int [GetDoFs](#) () const =0
- virtual const int [GetNode](#) (const int) const =0
- virtual const int [GetNeighbour](#) (const int) const =0
- virtual void [SetNeighbour](#) (const int k, const int elem)=0
- virtual void [SetType](#) (const int)=0
- virtual void [SetNode](#) (const int, const int)=0
- virtual const int [GetNumberOfNodes](#) () const =0
- virtual const double [GetShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetNormal](#) () const =0
- virtual void [ReverseNormal](#) ()=0
- virtual const int [SetOrder](#) (const int px, const int py)=0
- virtual const double [Integrate](#) (const [function_dp](#) &, const std::vector< [Point](#) > &v) const =0
- virtual const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const =0
- virtual const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const =0
- virtual const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const [function_dp](#) &f) const =0
- virtual const int [IncreaseOrder](#) ()=0
- virtual const double [GetMeasure](#) () const =0

6.15.1 Constructor & Destructor Documentation

6.15.1.1 CElement2D()

```
corenc::Mesh::CElement2D< bool >::CElement2D ( ) [inline]
```

6.15.1.2 ~CElement2D()

```
virtual corenc::Mesh::CElement2D< bool >::~~CElement2D ( ) [inline], [virtual]
```

6.15.2 Member Function Documentation

6.15.2.1 Clone()

```
virtual CElement2D * corenc::Mesh::CElement2D< bool >::Clone ( ) const [pure virtual]
```

6.15.2.2 GetDoFs()

```
virtual const int corenc::Mesh::CElement2D< bool >::GetDoFs ( ) const [pure virtual]
```

6.15.2.3 GetGradShapeFunction()

```
virtual const Point corenc::Mesh::CElement2D< bool >::GetGradShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

6.15.2.4 GetMeasure()

```
virtual const double corenc::Mesh::CElement2D< bool >::GetMeasure ( ) const [pure virtual]
```

6.15.2.5 GetNeighbour()

```
virtual const int corenc::Mesh::CElement2D< bool >::GetNeighbour (
    const int ) const [pure virtual]
```

6.15.2.6 GetNode()

```
virtual const int corenc::Mesh::CElement2D< bool >::GetNode (
    const int ) const [pure virtual]
```

6.15.2.7 GetNormal()

```
virtual const Point corenc::Mesh::CElement2D< bool >::GetNormal ( ) const [pure virtual]
```

6.15.2.8 GetNumberOfNodes()

```
virtual const int corenc::Mesh::CElement2D< bool >::GetNumberOfNodes ( ) const [pure virtual]
```

6.15.2.9 GetShapeFunction()

```
virtual const double corenc::Mesh::CElement2D< bool >::GetShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

6.15.2.10 GetType()

```
virtual const int corenc::Mesh::CElement2D< bool >::GetType ( ) const [pure virtual]
```

6.15.2.11 GetWeight()

```
virtual const double corenc::Mesh::CElement2D< bool >::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const function_dp & f ) const [pure virtual]
```

6.15.2.12 IncreaseOrder()

```
virtual const int corenc::Mesh::CElement2D< bool >::IncreaseOrder ( ) [pure virtual]
```

6.15.2.13 Integrate() [1/3]

```
virtual const double corenc::Mesh::CElement2D< bool >::Integrate (
    const function_dp & ,
    const std::vector< Point > & v ) const [pure virtual]
```

6.15.2.14 Integrate() [2/3]

```
virtual const Point corenc::Mesh::CElement2D< bool >::Integrate (
    const std::function< const Point (const Point &)> & ,
    const std::vector< Point > & v ) const [pure virtual]
```

6.15.2.15 Integrate() [3/3]

```
virtual const std::vector< double > corenc::Mesh::CElement2D< bool >::Integrate (
    const std::function< const std::vector< double > (const Point &)> & ,
    const std::vector< Point > & ) const [pure virtual]
```

6.15.2.16 ReverseNormal()

```
virtual void corenc::Mesh::CElement2D< bool >::ReverseNormal ( ) [pure virtual]
```

6.15.2.17 SetNeighbour()

```
virtual void corenc::Mesh::CElement2D< bool >::SetNeighbour (
    const int k,
    const int elem ) [pure virtual]
```

6.15.2.18 SetNode()

```
virtual void corenc::Mesh::CElement2D< bool >::SetNode (
    const int ,
    const int ) [pure virtual]
```

6.15.2.19 SetOrder()

```
virtual const int corenc::Mesh::CElement2D< bool >::SetOrder (
    const int px,
    const int py ) [pure virtual]
```

6.15.2.20 SetType()

```
virtual void corenc::Mesh::CElement2D< bool >::SetType (
    const int ) [pure virtual]
```

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

- CoreNCFEM/FiniteElements/[FiniteElement2D.h](#)

6.16 corenc::Mesh::CElement< bool > Class Reference

```
#include <FiniteElement.h>
```

Public Member Functions

- [CElement](#) ()
- virtual [~CElement](#) ()
- virtual const int [GetType](#) () const =0
- virtual [CElement](#) * [Clone](#) () const =0
- virtual const int [GetDoFs](#) () const =0
- virtual const int [GetNode](#) (const int) const =0
- virtual const int [GetNeighbour](#) (const int) const =0
- virtual void [SetNeighbour](#) (const int k, const int elem)=0
- virtual void [SetType](#) (const int)=0
- virtual void [SetNode](#) (const int, const int)=0
- virtual const int [GetNumberOfNodes](#) () const =0
- virtual const double [GetShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetNormal](#) () const =0
- virtual void [ReverseNormal](#) ()=0
- virtual const double [Integrate](#) (const [function_dp](#) &, const std::vector< [Point](#) > &v) const =0
- virtual const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const =0
- virtual const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const =0
- virtual const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const [function_dp](#) &f) const =0
- virtual const int [IncreaseOrder](#) ()=0
- virtual const double [GetMeasure](#) () const =0

6.16.1 Constructor & Destructor Documentation

6.16.1.1 CElement()

```
corenc::Mesh::CElement< bool >::CElement ( ) [inline]
```

6.16.1.2 ~CElement()

```
virtual corenc::Mesh::CElement< bool >::~~CElement ( ) [inline], [virtual]
```

6.16.2 Member Function Documentation

6.16.2.1 Clone()

```
virtual CElement * corenc::Mesh::CElement< bool >::Clone ( ) const [pure virtual]
```

6.16.2.2 GetDoFs()

```
virtual const int corenc::Mesh::CElement< bool >::GetDoFs ( ) const [pure virtual]
```

6.16.2.3 GetGradShapeFunction()

```
virtual const Point corenc::Mesh::CElement< bool >::GetGradShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

6.16.2.4 GetMeasure()

```
virtual const double corenc::Mesh::CElement< bool >::GetMeasure ( ) const [pure virtual]
```

6.16.2.5 GetNeighbour()

```
virtual const int corenc::Mesh::CElement< bool >::GetNeighbour (
    const int ) const [pure virtual]
```

6.16.2.6 GetNode()

```
virtual const int corenc::Mesh::CElement< bool >::GetNode (
    const int ) const [pure virtual]
```

6.16.2.7 GetNormal()

```
virtual const Point corenc::Mesh::CElement< bool >::GetNormal ( ) const [pure virtual]
```

6.16.2.8 GetNumberOfNodes()

```
virtual const int corenc::Mesh::CElement< bool >::GetNumberOfNodes ( ) const [pure virtual]
```

6.16.2.9 GetShapeFunction()

```
virtual const double corenc::Mesh::CElement< bool >::GetShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

6.16.2.10 GetType()

```
virtual const int corenc::Mesh::CElement< bool >::GetType ( ) const [pure virtual]
```

6.16.2.11 GetWeight()

```
virtual const double corenc::Mesh::CElement< bool >::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const function_dp & f ) const [pure virtual]
```

6.16.2.12 IncreaseOrder()

```
virtual const int corenc::Mesh::CElement< bool >::IncreaseOrder ( ) [pure virtual]
```

6.16.2.13 Integrate() [1/3]

```
virtual const double corenc::Mesh::CElement< bool >::Integrate (
    const function_dp & ,
    const std::vector< Point > & v ) const [pure virtual]
```

6.16.2.14 Integrate() [2/3]

```
virtual const Point corenc::Mesh::CElement< bool >::Integrate (
    const std::function< const Point (const Point &)> & ,
    const std::vector< Point > & v ) const [pure virtual]
```

6.16.2.15 Integrate() [3/3]

```
virtual const std::vector< double > corenc::Mesh::CElement< bool >::Integrate (
    const std::function< const std::vector< double > (const Point &)> & ,
    const std::vector< Point > & ) const [pure virtual]
```

6.16.2.16 ReverseNormal()

```
virtual void corenc::Mesh::CElement< bool >::ReverseNormal ( ) [pure virtual]
```

6.16.2.17 SetNeighbour()

```
virtual void corenc::Mesh::CElement< bool >::SetNeighbour (
    const int k,
    const int elem ) [pure virtual]
```

6.16.2.18 SetNode()

```
virtual void corenc::Mesh::CElement< bool >::SetNode (
    const int ,
    const int ) [pure virtual]
```

6.16.2.19 SetType()

```
virtual void corenc::Mesh::CElement< bool >::SetType (
    const int ) [pure virtual]
```

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

- CoreNCFEM/FiniteElements/[FiniteElement.h](#)

6.17 corenc::method::CFEMethod< Type > Class Template Reference

```
#include <FEMethod.h>
```

Public Member Functions

- [CFEMethod](#) ()
- virtual [~CFEMethod](#) ()
- virtual const int [Assemble](#) ()=0
- virtual const Type [GetSolution](#) (const std::vector< double > &point) const =0
- virtual const std::vector< Type > [GetSolution](#) () const =0
- virtual const Type [GetMaxSolution](#) () const =0
- virtual const Type [GetMinSolution](#) () const =0

6.17.1 Constructor & Destructor Documentation

6.17.1.1 CFEMethod()

```
template<class Type >
corenc::method::CFEMethod< Type >::CFEMethod ( ) [inline]
```

6.17.1.2 ~CFEMethod()

```
template<class Type >
virtual corenc::method::CFEMethod< Type >::~~CFEMethod ( ) [inline], [virtual]
```

6.17.2 Member Function Documentation

6.17.2.1 Assemble()

```
template<class Type >
virtual const int corenc::method::CFEMethod< Type >::Assemble ( ) [pure virtual]
```

6.17.2.2 GetMaxSolution()

```
template<class Type >
virtual const Type corenc::method::CFEMethod< Type >::GetMaxSolution ( ) const [pure virtual]
```

6.17.2.3 GetMinSolution()

```
template<class Type >
virtual const Type corenc::method::CFEMethod< Type >::GetMinSolution ( ) const [pure virtual]
```

6.17.2.4 GetSolution() [1/2]

```
template<class Type >
virtual const std::vector< Type > corenc::method::CFEMethod< Type >::GetSolution ( ) const
[pure virtual]
```

6.17.2.5 GetSolution() [2/2]

```
template<class Type >
virtual const Type corenc::method::CFEMethod< Type >::GetSolution (
    const std::vector< double > & point ) const [pure virtual]
```

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

- CoreNCFEM/Methods/[FEMethod.h](#)

6.18 corenc::method::CFEMethodZero< Type > Class Template Reference

```
#include <FEMethodZero.h>
```

Public Member Functions

- [CFEMethodZero](#) ()
- virtual [~CFEMethodZero](#) ()
- virtual const int [Assemble](#) ()=0
- virtual const Type [GetSolution](#) (const std::vector< double > &point) const =0
- virtual const std::vector< Type > [GetSolution](#) () const =0
- virtual const Type [GetMaxSolution](#) () const =0
- virtual const Type [GetMinSolution](#) () const =0

6.18.1 Constructor & Destructor Documentation

6.18.1.1 CFEMethodZero()

```
template<class Type >
corenc::method::CFEMethodZero< Type >::CFEMethodZero ( ) [inline]
```

6.18.1.2 ~CFEMethodZero()

```
template<class Type >
virtual corenc::method::CFEMethodZero< Type >::~~CFEMethodZero ( ) [inline], [virtual]
```

6.18.2 Member Function Documentation

6.18.2.1 Assemble()

```
template<class Type >
virtual const int corenc::method::CFEMethodZero< Type >::Assemble ( ) [pure virtual]
```

6.18.2.2 GetMaxSolution()

```
template<class Type >
virtual const Type corenc::method::CFEMethodZero< Type >::GetMaxSolution ( ) const [pure virtual]
```

6.18.2.3 GetMinSolution()

```
template<class Type >
virtual const Type corenc::method::CFEMethodZero< Type >::GetMinSolution ( ) const [pure virtual]
```

6.18.2.4 GetSolution() [1/2]

```
template<class Type >
virtual const std::vector< Type > corenc::method::CFEMethodZero< Type >::GetSolution ( )
const [pure virtual]
```

6.18.2.5 GetSolution() [2/2]

```
template<class Type >
virtual const Type corenc::method::CFEMethodZero< Type >::GetSolution (
    const std::vector< double > & point ) const [pure virtual]
```

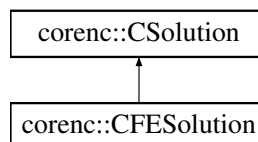
The documentation for this class was generated from the following file:

- CoreNCFEM/Methods/[FEMethodZero.h](#)

6.19 corenc::CFESolution Class Reference

```
#include <FESolution.h>
```

Inheritance diagram for corenc::CFESolution:



Public Member Functions

- [CFESolution](#) ()
- [~CFESolution](#) ()
- [CFESolution](#) & [operator=](#) (const [CFESolution](#) &fe)
- [CFESolution](#) & [operator=](#) (const double fe)
- [CFESolution](#) (const [CFESolution](#) &fe)
- [CFESolution](#) (const double &fe)
- [operator double](#) () const
- const bool [operator==](#) (const [CFESolution](#) &fe)
- const bool [operator!=](#) (const [CFESolution](#) &fe)
- [CFESolution](#) & [operator+=](#) (const [CFESolution](#) &fe)
- [CFESolution](#) & [operator-=](#) (const [CFESolution](#) &fe)
- [CFESolution](#) & [operator*=](#) (const [CFESolution](#) &fe)
- [CFESolution](#) & [operator/=](#) (const [CFESolution](#) &fe)

Friends

- const double [operator*](#) (const [CFESolution](#) &lhs, const [CFESolution](#) &rhs)
- const double [operator*](#) (const [CFESolution](#) &lhs, const double rhs)
- const double [operator*](#) (const double lhs, const [CFESolution](#) &rhs)
- const double [operator-](#) (const [CFESolution](#) &lhs, const [CFESolution](#) &rhs)
- const double [operator+](#) (const [CFESolution](#) &lhs, const [CFESolution](#) &rhs)
- const double [operator/](#) (const [CFESolution](#) &lhs, const [CFESolution](#) &rhs)

6.19.1 Constructor & Destructor Documentation

6.19.1.1 CFESolution() [1/3]

```
corenc::CFESolution::CFESolution ( ) [inline]
```

6.19.1.2 ~CFESolution()

```
corenc::CFESolution::~~CFESolution ( ) [inline]
```

6.19.1.3 CFESolution() [2/3]

```
corenc::CFESolution::CFESolution (
    const CFESolution & fe ) [inline]
```

6.19.1.4 CFESolution() [3/3]

```
corenc::CFESolution::CFESolution (
    const double & fe ) [inline]
```

6.19.2 Member Function Documentation

6.19.2.1 operator double()

```
corenc::CFESolution::operator double ( ) const [inline]
```


6.19.2.2 operator!=(())

```
const bool corenc::CFESolution::operator!= (
    const CFESolution & fe ) [inline]
```

6.19.2.3 operator*=(())

```
CFESolution & corenc::CFESolution::operator*= (
    const CFESolution & fe ) [inline]
```

6.19.2.4 operator+=(())

```
CFESolution & corenc::CFESolution::operator+= (
    const CFESolution & fe ) [inline]
```

6.19.2.5 operator-=(())

```
CFESolution & corenc::CFESolution::operator-= (
    const CFESolution & fe ) [inline]
```

6.19.2.6 operator/=(())

```
CFESolution & corenc::CFESolution::operator/= (
    const CFESolution & fe ) [inline]
```

6.19.2.7 operator=() [1/2]

```
CFESolution & corenc::CFESolution::operator= (
    const CFESolution & fe ) [inline]
```

6.19.2.8 operator=() [2/2]

```
CFESolution & corenc::CFESolution::operator= (
    const double fe ) [inline]
```

6.19.2.9 operator==()

```
const bool corenc::CFESolution::operator== (
    const CFESolution & fe ) [inline]
```

6.19.3 Friends And Related Function Documentation

6.19.3.1 operator* [1/3]

```
const double operator* (
    const CFESolution & lhs,
    const CFESolution & rhs ) [friend]
```

6.19.3.2 operator* [2/3]

```
const double operator* (
    const CFESolution & lhs,
    const double rhs ) [friend]
```

6.19.3.3 operator* [3/3]

```
const double operator* (
    const double lhs,
    const CFESolution & rhs ) [friend]
```

6.19.3.4 operator+

```
const double operator+ (
    const CFESolution & lhs,
    const CFESolution & rhs ) [friend]
```

6.19.3.5 operator-

```
const double operator- (
    const CFESolution & lhs,
    const CFESolution & rhs ) [friend]
```

6.19.3.6 operator/

```
const double operator/ (
    const CFESolution & lhs,
    const CFESolution & rhs ) [friend]
```

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

- CoreNCFEM/[FESolution.h](#)

6.20 corenc::CFEweights Class Reference

```
#include <FESolution.h>
```

Public Member Functions

- [CFEweights](#) ()
- [~CFEweights](#) ()
- const [CFESolution](#) [getWeight](#) (const unsigned int i) const
- const int [updateWeight](#) (const unsigned int i, const [CFESolution](#) &cfe)

6.20.1 Constructor & Destructor Documentation

6.20.1.1 CFEweights()

```
corenc::CFEweights::CFEweights ( ) [inline]
```

6.20.1.2 ~CFEweights()

```
corenc::CFEweights::~~CFEweights ( ) [inline]
```

6.20.2 Member Function Documentation

6.20.2.1 getWeight()

```
const CFESolution corenc::CFEweights::getWeight (
    const unsigned int i ) const [inline]
```

6.20.2.2 updateWeight()

```
const int corenc::CFEweights::updateWeight (
    const unsigned int i,
    const CFESolution & cfe ) [inline]
```

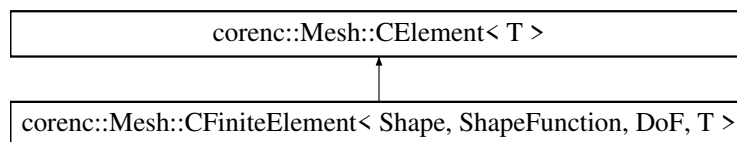
The documentation for this class was generated from the following file:

- [CoreNCFEM/FESolution.h](#)

6.21 corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T > Class Template Reference

```
#include <FiniteElement.h>
```

Inheritance diagram for corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >:



Public Member Functions

- [CFiniteElement](#) ()
- [CFiniteElement](#) (const int *nodes, const [Point](#) *points, const int dofs)
- [CFiniteElement](#) (const int *nodes, const [Point](#) *points)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &f, const DoF &d)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &shfunc, const DoF &dofs, const int type)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &shfunc, const DoF &dofs, const int type, const int *neigs)
- [CFiniteElement](#) (const [CFiniteElement](#)< Shape, ShapeFunction, DoF > &e)
- [CElement](#)< T > * [Clone](#) () const
- [~CFiniteElement](#) ()
- const int [GetType](#) () const
- const int [GetNode](#) (const int) const
- const int [GetNeighbour](#) (const int) const
- const Shape [GetShape](#) () const
- const ShapeFunction [GetShapeFunctions](#) () const
- const DoF [GetDoF](#) () const
- const int [GetDoFs](#) () const
- void [SetNeighbour](#) (const int k, const int elem)
- void [SetType](#) (const int)
- void [SetShapeFunction](#) (const int, const ShapeFunction &)
- void [SetDoF](#) (const DoF &)
- void [SetShape](#) (const Shape &)
- const int [IncreaseOrder](#) ()
- void [SetNode](#) (const int, const int)
- const int [GetNumberOfNodes](#) () const

- const double [GetMeasure](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- [CFiniteElement](#) & [operator=](#) (const [CFiniteElement](#) &e)

Friends

- const bool [operator==](#) (const [CFiniteElement](#) &e1, const [CFiniteElement](#) &e2)
- std::istream & [operator>>](#) (std::istream &is, [CFiniteElement](#) &k)

6.21.1 Constructor & Destructor Documentation

6.21.1.1 CFiniteElement() [1/7]

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::CFiniteElement ( ) [inline]
```

6.21.1.2 CFiniteElement() [2/7]

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::CFiniteElement (
    const int * nodes,
    const Point * points,
    const int dofs ) [inline]
```

6.21.1.3 CFiniteElement() [3/7]

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::CFiniteElement (
    const int * nodes,
    const Point * points ) [inline]
```

6.21.1.4 CFiniteElement() [4/7]

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & f,
    const DoF & d ) [inline]
```

6.21.1.5 CFiniteElement() [5/7]

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const DoF & dofs,
    const int type ) [inline]
```

6.21.1.6 CFiniteElement() [6/7]

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const DoF & dofs,
    const int type,
    const int * neigs ) [inline]
```

6.21.1.7 CFiniteElement() [7/7]

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::CFiniteElement (
    const CFiniteElement< Shape, ShapeFunction, DoF > & e ) [inline]
```

6.21.1.8 ~CFiniteElement()

```
template<class Shape , class ShapeFunction , class DoF , class T >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::~~CFiniteElement ( ) [inline]
```

6.21.2 Member Function Documentation

6.21.2.1 Clone()

```
template<class Shape , class ShapeFunction , class DoF , class T >
CElement< T > * corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::Clone ( ) const
[inline], [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.2 GetDoF()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const DoF corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetDoF
```

6.21.2.3 GetDoFs()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetDoFs [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.4 GetGradShapeFunction()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetGradShapeFunction
(
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.5 GetMeasure()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetMeasure [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.6 GetNeighbour()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetNeighbour (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.7 GetNode()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetNode (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.8 GetNormal()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetNormal [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.9 GetNumberOfNodes()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetNumberOfNodes
[virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.10 GetShape()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const Shape corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetShape
```


6.21.2.11 GetShapeFunction()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.12 GetShapeFunctions()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const ShapeFunction corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetShapeFunctions
```

6.21.2.13 GetType()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetType [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.14 GetWeight()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.15 IncreaseOrder()

```
template<class Shape , class ShapeFunction , class DoF , class T >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::IncreaseOrder [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.16 Integrate() [1/3]

```
template<class Shape , class ShapeFunction , class DoF , class T >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.17 Integrate() [2/3]

```
template<class Shape , class ShapeFunction , class DoF , class T >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.18 Integrate() [3/3]

```
template<class Shape , class ShapeFunction , class DoF , class T >
const std::vector< double > corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::↵
Integrate (
    const std::function< const std::vector< double >(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.19 operator=()

```
template<class Shape , class ShapeFunction , class DoF , class T >
CFiniteElement & corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::operator= (
    const CFiniteElement< Shape, ShapeFunction, DoF, T > & e ) [inline]
```

6.21.2.20 ReverseNormal()

```
template<class Shape , class ShapeFunction , class DoF , class T >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::ReverseNormal [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.21 SetDoF()

```
template<class Shape , class ShapeFunction , class DoF , class T >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::SetDoF (
    const DoF & dof )
```

6.21.2.22 SetNeighbour()

```
template<class Shape , class ShapeFunction , class DoF , class T >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::SetNeighbour (
    const int k,
    const int elem ) [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.23 SetNode()

```
template<class Shape , class ShapeFunction , class DoF , class T >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.2.24 SetShape()

```
template<class Shape , class ShapeFunction , class DoF , class T >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::SetShape (
    const Shape & shape )
```

6.21.2.25 SetShapeFunction()

```
template<class Shape , class ShapeFunction , class DoF , class T >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::SetShapeFunction (
    const int k,
    const ShapeFunction & func )
```

6.21.2.26 SetType()

```
template<class Shape , class ShapeFunction , class DoF , class T >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >::SetType (
    const int k ) [virtual]
```

Implements [corenc::Mesh::CElement< T >](#).

6.21.3 Friends And Related Function Documentation

6.21.3.1 operator==

```
template<class Shape , class ShapeFunction , class DoF , class T >
const bool operator== (
    const CFiniteElement< Shape, ShapeFunction, DoF, T > & e1,
    const CFiniteElement< Shape, ShapeFunction, DoF, T > & e2 ) [friend]
```

6.21.3.2 operator>>

```
template<class Shape , class ShapeFunction , class DoF , class T >
std::istream & operator>> (
    std::istream & is,
    CFiniteElement< Shape, ShapeFunction, DoF, T > & k ) [friend]
```

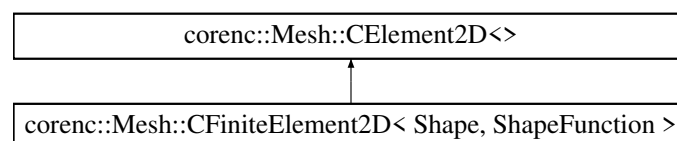
The documentation for this class was generated from the following file:

- CoreNCFEM/FiniteElements/[FiniteElement.h](#)

6.22 corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction > Class Template Reference

```
#include <FiniteElement2D.h>
```

Inheritance diagram for `corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >`:



Public Member Functions

- [CFiniteElement2D](#) ()
- [CFiniteElement2D](#) (const int *nodes, const [Point](#) *points, const int dofs)
- [CFiniteElement2D](#) (const int *nodes, const [Point](#) *points, const int dofs, const int type)
- [CFiniteElement2D](#) (const int *nodes, const [Point](#) *points)
- [CFiniteElement2D](#) (const Shape &shape, const ShapeFunction &f)
- [CFiniteElement2D](#) (const Shape &shape, const ShapeFunction &shfunc, const int type)
- [CFiniteElement2D](#) (const Shape &shape, const ShapeFunction &shfunc, const int type, const int *neigs)
- [CFiniteElement2D](#) (const [CFiniteElement2D](#) &e)
- [CElement2D](#) * [Clone](#) () const
- [~CFiniteElement2D](#) ()
- const int [GetType](#) () const
- const int [GetNode](#) (const int) const
- const int [GetNeighbour](#) (const int) const
- const Shape [GetShape](#) () const
- const ShapeFunction [GetShapeFunctions](#) () const
- const int [GetDoFs](#) () const
- void [SetNeighbour](#) (const int k, const int elem)
- void [SetType](#) (const int)
- void [SetShapeFunction](#) (const int, const ShapeFunction &)
- void [SetShape](#) (const Shape &)
- const int [SetOrder](#) (const int px, const int py)
- void [SetNode](#) (const int, const int)
- const int [GetNumberOfNodes](#) () const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- [CFiniteElement2D](#) & [operator=](#) (const [CFiniteElement2D](#) &e)

Friends

- const bool [operator==](#) (const [CFiniteElement2D](#) &e1, const [CFiniteElement2D](#) &e2)
- std::istream & [operator>>](#) (std::istream &is, [CFiniteElement2D](#) &k)

6.22.1 Constructor & Destructor Documentation

6.22.1.1 CFiniteElement2D() [1/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D ( ) [inline]
```

6.22.1.2 CFiniteElement2D() [2/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D (
    const int * nodes,
    const Point * points,
    const int dofs ) [inline]
```

6.22.1.3 CFiniteElement2D() [3/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D (
    const int * nodes,
    const Point * points,
    const int dofs,
    const int type ) [inline]
```

6.22.1.4 CFiniteElement2D() [4/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D (
    const int * nodes,
    const Point * points ) [inline]
```

6.22.1.5 CFiniteElement2D() [5/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D (
    const Shape & shape,
    const ShapeFunction & f ) [inline]
```

6.22.1.6 CFiniteElement2D() [6/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const int type ) [inline]
```

6.22.1.7 CFiniteElement2D() [7/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const int type,
    const int * neigs ) [inline]
```

6.22.1.8 CFiniteElement2D() [8/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::CFiniteElement2D (
    const CFiniteElement2D< Shape, ShapeFunction > & e ) [inline]
```

6.22.1.9 ~CFiniteElement2D()

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::~~CFiniteElement2D ( ) [inline]
```

6.22.2 Member Function Documentation**6.22.2.1 Clone()**

```
template<class Shape , class ShapeFunction >
CElement2D * corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::Clone ( ) const [inline],
[virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.2 GetDoFs()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetDoFs [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.3 GetGradShapeFunction()

```
template<class Shape , class ShapeFunction >
const Point corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.4 GetMeasure()

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetMeasure [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.5 GetNeighbour()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetNeighbour (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.6 GetNode()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetNode (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.7 GetNormal()

```
template<class Shape , class ShapeFunction >
const Point corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetNormal [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.8 GetNumberOfNodes()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetNumberOfNodes [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.9 GetShape()

```
template<class Shape , class ShapeFunction >
const Shape corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetShape
```

6.22.2.10 GetShapeFunction()

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.11 GetShapeFunctions()

```
template<class Shape , class ShapeFunction >
const ShapeFunction corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetShapeFunctions
```

6.22.2.12 GetType()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetType [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.13 GetWeight()

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.14 IncreaseOrder()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::IncreaseOrder [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.15 Integrate() [1/3]

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.16 Integrate() [2/3]

```
template<class Shape , class ShapeFunction >
const Point corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.17 Integrate() [3/3]

```
template<class Shape , class ShapeFunction >
const std::vector< double > corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::Integrate
(
    const std::function< const std::vector< double >(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.18 operator=()

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::operator= (
    const CFiniteElement2D< Shape, ShapeFunction > & e ) [inline]
```

6.22.2.19 ReverseNormal()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::ReverseNormal [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.20 SetNeighbour()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::SetNeighbour (
    const int k,
    const int elem ) [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.21 SetNode()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.22 SetOrder()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::SetOrder (
    const int px,
    const int py ) [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.2.23 SetShape()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::SetShape (
    const Shape & shape )
```

6.22.2.24 SetShapeFunction()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::SetShapeFunction (
    const int k,
    const ShapeFunction & func )
```

6.22.2.25 SetType()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >::SetType (
    const int k ) [virtual]
```

Implements [corenc::Mesh::CElement2D<>](#).

6.22.3 Friends And Related Function Documentation

6.22.3.1 operator==

```
template<class Shape , class ShapeFunction >
const bool operator== (
    const CFiniteElement2D< Shape, ShapeFunction > & e1,
    const CFiniteElement2D< Shape, ShapeFunction > & e2 ) [friend]
```

6.22.3.2 operator>>

```
template<class Shape , class ShapeFunction >
std::istream & operator>> (
    std::istream & is,
    CFiniteElement2D< Shape, ShapeFunction > & k ) [friend]
```

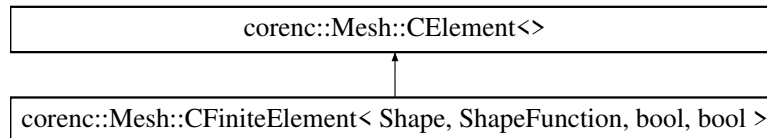
The documentation for this class was generated from the following file:

- [CoreNCFEM/FiniteElements/FiniteElement2D.h](#)

6.23 corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool > Class Template Reference

```
#include <FiniteElement.h>
```

Inheritance diagram for corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >:



Public Member Functions

- [CFiniteElement](#) ()
- [CFiniteElement](#) (const int *nodes, const [Point](#) *points, const int dofs)
- [CFiniteElement](#) (const int *nodes, const [Point](#) *points, const int dofs, const int type)
- [CFiniteElement](#) (const int *nodes, const [Point](#) *points)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &f)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &shfunc, const int type)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &shfunc, const int type, const int *neigs)
- [CFiniteElement](#) (const [CFiniteElement](#)< Shape, ShapeFunction > &e)
- [CElement](#) * [Clone](#) () const
- [~CFiniteElement](#) ()
- const int [GetType](#) () const
- const int [GetNode](#) (const int) const
- const int [GetNeighbour](#) (const int) const
- const Shape [GetShape](#) () const
- const ShapeFunction [GetShapeFunctions](#) () const
- const int [GetDoFs](#) () const
- void [SetNeighbour](#) (const int k, const int elem)
- void [SetType](#) (const int)
- void [SetShapeFunction](#) (const int, const ShapeFunction &)
- void [SetShape](#) (const Shape &)
- void [SetNode](#) (const int, const int)
- const int [GetNumberOfNodes](#) () const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- [CFiniteElement](#) & [operator=](#) (const [CFiniteElement](#) &e)

Friends

- const bool `operator==` (const `CFiniteElement` &e1, const `CFiniteElement` &e2)
- std::istream & `operator>>` (std::istream &is, `CFiniteElement` &k)

6.23.1 Constructor & Destructor Documentation

6.23.1.1 `CFiniteElement()` [1/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement ( ) [inline]
```

6.23.1.2 `CFiniteElement()` [2/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement (
    const int * nodes,
    const Point * points,
    const int dofs ) [inline]
```

6.23.1.3 `CFiniteElement()` [3/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement (
    const int * nodes,
    const Point * points,
    const int dofs,
    const int type ) [inline]
```

6.23.1.4 `CFiniteElement()` [4/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement (
    const int * nodes,
    const Point * points ) [inline]
```

6.23.1.5 CFiniteElement() [5/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & f ) [inline]
```

6.23.1.6 CFiniteElement() [6/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const int type ) [inline]
```

6.23.1.7 CFiniteElement() [7/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const int type,
    const int * neigs ) [inline]
```

6.23.1.8 CFiniteElement() [8/8]

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::CFiniteElement (
    const CFiniteElement< Shape, ShapeFunction > & e ) [inline]
```

6.23.1.9 ~CFiniteElement()

```
template<class Shape , class ShapeFunction >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::~~CFiniteElement ( ) [inline]
```

6.23.2 Member Function Documentation

6.23.2.1 Clone()

```
template<class Shape , class ShapeFunction >
CElement * corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::Clone ( ) const
[inline], [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.2 GetDoFs()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetDoFs [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.3 GetGradShapeFunction()

```
template<class Shape , class ShapeFunction >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.4 GetMeasure()

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetMeasure
[virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.5 GetNeighbour()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetNeighbour (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.6 GetNode()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetNode (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.7 GetNormal()

```
template<class Shape , class ShapeFunction >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetNormal [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.8 GetNumberOfNodes()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetNumberOfNodes
[virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.9 GetShape()

```
template<class Shape , class ShapeFunction >
const Shape corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetShape
```

6.23.2.10 GetShapeFunction()

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetShape↵
Function (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.11 GetShapeFunctions()

```
template<class Shape , class ShapeFunction >
const ShapeFunction corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::Get↵
ShapeFunctions
```

6.23.2.12 GetType()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetType [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.13 GetWeight()

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.14 IncreaseOrder()

```
template<class Shape , class ShapeFunction >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::IncreaseOrder
[virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.15 Integrate() [1/3]

```
template<class Shape , class ShapeFunction >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.16 Integrate() [2/3]

```
template<class Shape , class ShapeFunction >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::Integrate (
    const std::function< const Point (const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.17 Integrate() [3/3]

```
template<class Shape , class ShapeFunction >
const std::vector< double > corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::Integrate (
    const std::function< const std::vector< double > (const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.18 operator=()

```
template<class Shape , class ShapeFunction >
CFiniteElement & corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::operator= (
    const CFiniteElement< Shape, ShapeFunction, bool, bool > & e ) [inline]
```

6.23.2.19 ReverseNormal()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::ReverseNormal [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.20 SetNeighbour()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::SetNeighbour (
    const int k,
    const int elem ) [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.21 SetNode()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.2.22 SetShape()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::SetShape (
    const Shape & shape )
```

6.23.2.23 SetShapeFunction()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::SetShapeFunction (
    const int k,
    const ShapeFunction & func )
```

6.23.2.24 SetType()

```
template<class Shape , class ShapeFunction >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >::SetType (
    const int k ) [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.23.3 Friends And Related Function Documentation

6.23.3.1 operator==

```
template<class Shape , class ShapeFunction >
const bool operator== (
    const CFiniteElement< Shape, ShapeFunction, bool, bool > & e1,
    const CFiniteElement< Shape, ShapeFunction, bool, bool > & e2 ) [friend]
```

6.23.3.2 operator>>

```
template<class Shape , class ShapeFunction >
std::istream & operator>> (
    std::istream & is,
    CFiniteElement< Shape, ShapeFunction, bool, bool > & k ) [friend]
```

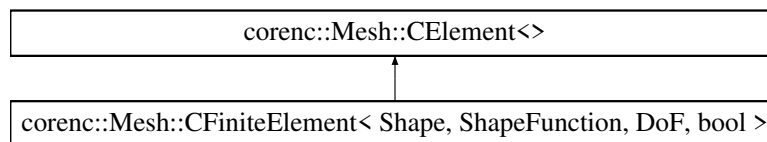
The documentation for this class was generated from the following file:

- CoreNCFEM/FiniteElements/[FiniteElement.h](#)

6.24 corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool > Class Template Reference

```
#include <FiniteElement.h>
```

Inheritance diagram for corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >:



Public Member Functions

- [CFiniteElement](#) ()
- [CFiniteElement](#) (const int *nodes, const [Point](#) *points, const int dofs)
- [CFiniteElement](#) (const int *nodes, const [Point](#) *points)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &f, const DoF &d)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &shfunc, const DoF &dofs, const int type)
- [CFiniteElement](#) (const Shape &shape, const ShapeFunction &shfunc, const DoF &dofs, const int type, const int *neigh)
- [CFiniteElement](#) (const [CFiniteElement](#)< Shape, ShapeFunction, DoF > &e)
- [CElement](#) * [Clone](#) () const
- [~CFiniteElement](#) ()
- const int [GetType](#) () const
- const int [GetNode](#) (const int) const
- const int [GetNeighbour](#) (const int) const
- const Shape [GetShape](#) () const
- const ShapeFunction [GetShapeFunctions](#) () const
- const DoF [GetDoF](#) () const
- const int [GetDoFs](#) () const
- void [SetNeighbour](#) (const int k, const int elem)
- void [SetType](#) (const int)
- void [SetShapeFunction](#) (const int, const ShapeFunction &)
- void [SetDoF](#) (const DoF &)
- void [SetShape](#) (const Shape &)
- void [SetNode](#) (const int, const int)
- const int [GetNumberOfNodes](#) () const

- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- [CFiniteElement](#) & [operator=](#) (const [CFiniteElement](#) &e)

Friends

- const bool [operator==](#) (const [CFiniteElement](#) &e1, const [CFiniteElement](#) &e2)
- std::istream & [operator>>](#) (std::istream &is, [CFiniteElement](#) &k)

6.24.1 Constructor & Destructor Documentation

6.24.1.1 CFiniteElement() [1/7]

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::CFiniteElement ( ) [inline]
```

6.24.1.2 CFiniteElement() [2/7]

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::CFiniteElement (
    const int * nodes,
    const Point * points,
    const int dofs ) [inline]
```

6.24.1.3 CFiniteElement() [3/7]

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::CFiniteElement (
    const int * nodes,
    const Point * points ) [inline]
```

6.24.1.4 CFiniteElement() [4/7]

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & f,
    const DoF & d ) [inline]
```

6.24.1.5 CFiniteElement() [5/7]

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const DoF & dofs,
    const int type ) [inline]
```

6.24.1.6 CFiniteElement() [6/7]

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::CFiniteElement (
    const Shape & shape,
    const ShapeFunction & shfunc,
    const DoF & dofs,
    const int type,
    const int * neigh ) [inline]
```

6.24.1.7 CFiniteElement() [7/7]

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::CFiniteElement (
    const CFiniteElement< Shape, ShapeFunction, DoF > & e ) [inline]
```

6.24.1.8 ~CFiniteElement()

```
template<class Shape , class ShapeFunction , class DoF >
corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::~~CFiniteElement ( ) [inline]
```

6.24.2 Member Function Documentation

6.24.2.1 Clone()

```
template<class Shape , class ShapeFunction , class DoF >
CElement * corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::Clone ( ) const
[inline], [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.2 GetDoF()

```
template<class Shape , class ShapeFunction , class DoF >
const DoF corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetDoF
```

6.24.2.3 GetDoFs()

```
template<class Shape , class ShapeFunction , class DoF >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetDoFs [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.4 GetGradShapeFunction()

```
template<class Shape , class ShapeFunction , class DoF >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.5 GetMeasure()

```
template<class Shape , class ShapeFunction , class DoF >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetMeasure
[virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.6 GetNeighbour()

```
template<class Shape , class ShapeFunction , class DoF >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetNeighbour (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.7 GetNode()

```
template<class Shape , class ShapeFunction , class DoF >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetNode (
    const int k ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.8 GetNormal()

```
template<class Shape , class ShapeFunction , class DoF >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetNormal [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.9 GetNumberOfNodes()

```
template<class Shape , class ShapeFunction , class DoF >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetNumberOfNodes
[virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.10 GetShape()

```
template<class Shape , class ShapeFunction , class DoF >
const Shape corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetShape
```

6.24.2.11 GetShapeFunction()

```
template<class Shape , class ShapeFunction , class DoF >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetShapeFunction
(
    const int ,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.12 GetShapeFunctions()

```
template<class Shape , class ShapeFunction , class DoF >
const ShapeFunction corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::Get←
ShapeFunctions
```

6.24.2.13 GetType()

```
template<class Shape , class ShapeFunction , class DoF >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetType [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.14 GetWeight()

```
template<class Shape , class ShapeFunction , class DoF >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.15 IncreaseOrder()

```
template<class Shape , class ShapeFunction , class DoF >
const int corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::IncreaseOrder
[virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.16 Integrate() [1/3]

```
template<class Shape , class ShapeFunction , class DoF >
const double corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.17 Integrate() [2/3]

```
template<class Shape , class ShapeFunction , class DoF >
const Point corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.18 Integrate() [3/3]

```
template<class Shape , class ShapeFunction , class DoF >
const std::vector< double > corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >↵
::Integrate (
    const std::function< const std::vector< double >(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.19 operator=()

```
template<class Shape , class ShapeFunction , class DoF >
CFiniteElement & corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::operator= (
    const CFiniteElement< Shape, ShapeFunction, DoF, bool > & e ) [inline]
```

6.24.2.20 ReverseNormal()

```
template<class Shape , class ShapeFunction , class DoF >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::ReverseNormal [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.21 SetDoF()

```
template<class Shape , class ShapeFunction , class DoF >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::SetDoF (
    const DoF & dof )
```

6.24.2.22 SetNeighbour()

```
template<class Shape , class ShapeFunction , class DoF >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::SetNeighbour (
    const int k,
    const int elem ) [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.23 SetNode()

```
template<class Shape , class ShapeFunction , class DoF >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.2.24 SetShape()

```
template<class Shape , class ShapeFunction , class DoF >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::SetShape (
    const Shape & shape )
```

6.24.2.25 SetShapeFunction()

```
template<class Shape , class ShapeFunction , class DoF >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::SetShapeFunction (
    const int k,
    const ShapeFunction & func )
```

6.24.2.26 SetType()

```
template<class Shape , class ShapeFunction , class DoF >
void corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >::SetType (
    const int k ) [virtual]
```

Implements [corenc::Mesh::CElement<>](#).

6.24.3 Friends And Related Function Documentation

6.24.3.1 operator==

```
template<class Shape , class ShapeFunction , class DoF >
const bool operator== (
    const CFiniteElement< Shape, ShapeFunction, DoF, bool > & e1,
    const CFiniteElement< Shape, ShapeFunction, DoF, bool > & e2 ) [friend]
```

6.24.3.2 operator>>

```
template<class Shape , class ShapeFunction , class DoF >
std::istream & operator>> (
    std::istream & is,
    CFiniteElement< Shape, ShapeFunction, DoF, bool > & k ) [friend]
```

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

- CoreNCFEM/FiniteElements/[FiniteElement.h](#)

6.25 corenc::CFiniteSolver< Method, Mesh, Solver > Class Template Reference

```
#include <FiniteSolver.h>
```

Public Member Functions

- [CFiniteSolver](#) ()
- [~CFiniteSolver](#) ()
- void [Solve](#) ()

6.25.1 Constructor & Destructor Documentation

6.25.1.1 CFiniteSolver()

```
template<class Method , class Mesh , class Solver >
corenc::CFiniteSolver< Method, Mesh, Solver >::CFiniteSolver ( ) [inline]
```

6.25.1.2 ~CFiniteSolver()

```
template<class Method , class Mesh , class Solver >
corenc::CFiniteSolver< Method, Mesh, Solver >::~~CFiniteSolver ( ) [inline]
```

6.25.2 Member Function Documentation

6.25.2.1 Solve()

```
template<class Method , class Mesh , class Solver >
void corenc::CFiniteSolver< Method, Mesh, Solver >::Solve
```

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

- CoreNCFEM/[FiniteSolver.h](#)

6.26 corenc::Mesh::CMesh< T > Class Template Reference

```
#include <Mesh.h>
```

Public Member Functions

- [CMesh](#) ()
- virtual [~CMesh](#) ()
- virtual const unsigned int [GetNumberOfNodes](#) () const =0
- virtual const unsigned int [GetNumberOfElements](#) () const =0
- virtual const int [FindElement](#) (const [Point](#) &) const =0
- virtual const unsigned int [GetNumberOfBoundaries](#) () const =0
- virtual const [CElement](#)< T > * [GetElement](#) (const unsigned int) const =0
- virtual const [CElement](#)< T > * [GetBoundary](#) (const unsigned int) const =0
- virtual const [Point](#) [GetNode](#) (const unsigned int) const =0
- virtual const double [getSolution](#) (const unsigned int element, const unsigned int node) const =0
- virtual const int [updateSolution](#) (const unsigned int element, const unsigned int node, const double value)=0
- virtual const std::vector< double > [getSolution](#) () const =0
- virtual const int [updateSolution](#) (const std::vector< double > &)=0
- virtual const int [updateSolution](#) (const unsigned int element, const unsigned int node, [CSolution](#) *value)=0
- virtual const double [getParameter](#) ([Parameters](#), const unsigned int, const [Point](#) &) const =0
- virtual const double [getParameter](#) ([Parameters](#), const unsigned int, const int) const =0
- virtual const int [setParameter](#) ([Parameters](#), const double, const unsigned int)=0
- virtual const double [getMinSize](#) () const =0
- virtual const int [updateSolution](#) (const unsigned int node, const double value)=0

6.26.1 Constructor & Destructor Documentation

6.26.1.1 CMesh()

```
template<class T >
corenc::Mesh::CMesh< T >::CMesh ( ) [inline]
```

6.26.1.2 ~CMesh()

```
template<class T >
virtual corenc::Mesh::CMesh< T >::~~CMesh ( ) [inline], [virtual]
```

6.26.2 Member Function Documentation

6.26.2.1 FindElement()

```
template<class T >
virtual const int corenc::Mesh::CMesh< T >::FindElement (
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.2 GetBoundary()

```
template<class T >
virtual const CElement< T > * corenc::Mesh::CMesh< T >::GetBoundary (
    const unsigned int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.3 GetElement()

```
template<class T >
virtual const CElement< T > * corenc::Mesh::CMesh< T >::GetElement (
    const unsigned int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.4 getMinSize()

```
template<class T >
virtual const double corenc::Mesh::CMesh< T >::getMinSize ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.5 GetNode()

```
template<class T >
virtual const Point corenc::Mesh::CMesh< T >::GetNode (
    const unsigned int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.6 GetNumberOfBoundaries()

```
template<class T >
virtual const unsigned int corenc::Mesh::CMesh< T >::GetNumberOfBoundaries ( ) const [pure
virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.7 GetNumberOfElements()

```
template<class T >
virtual const unsigned int corenc::Mesh::CMesh< T >::GetNumberOfElements ( ) const [pure
virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.8 GetNumberOfNodes()

```
template<class T >
virtual const unsigned int corenc::Mesh::CMesh< T >::GetNumberOfNodes ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.9 `getParameter()` [1/2]

```
template<class T >
virtual const double corenc::Mesh::CMesh< T >::getParameter (
    Parameters ,
    const unsigned int ,
    const int ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.10 `getParameter()` [2/2]

```
template<class T >
virtual const double corenc::Mesh::CMesh< T >::getParameter (
    Parameters ,
    const unsigned int ,
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CTriangularMesh](#), [corenc::Mesh::CTriangularMeshLinear](#), and [corenc::Mesh::CMesh1D](#).

6.26.2.11 `getSolution()` [1/2]

```
template<class T >
virtual const std::vector< double > corenc::Mesh::CMesh< T >::getSolution ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.12 `getSolution()` [2/2]

```
template<class T >
virtual const double corenc::Mesh::CMesh< T >::getSolution (
    const unsigned int element,
    const unsigned int node ) const [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.13 `setParameter()`

```
template<class T >
virtual const int corenc::Mesh::CMesh< T >::setParameter (
    Parameters ,
    const double ,
    const unsigned int ) [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.14 updateSolution() [1/4]

```
template<class T >
virtual const int corenc::Mesh::CMesh< T >::updateSolution (
    const std::vector< double > & ) [pure virtual]
```

Implemented in [corenc::Mesh::CTriangularMesh](#), [corenc::Mesh::CTriangularMeshLinear](#), and [corenc::Mesh::CMesh1D](#).

6.26.2.15 updateSolution() [2/4]

```
template<class T >
virtual const int corenc::Mesh::CMesh< T >::updateSolution (
    const unsigned int element,
    const unsigned int node,
    const double value ) [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.16 updateSolution() [3/4]

```
template<class T >
virtual const int corenc::Mesh::CMesh< T >::updateSolution (
    const unsigned int element,
    const unsigned int node,
    CSolution * value ) [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

6.26.2.17 updateSolution() [4/4]

```
template<class T >
virtual const int corenc::Mesh::CMesh< T >::updateSolution (
    const unsigned int node,
    const double value ) [pure virtual]
```

Implemented in [corenc::Mesh::CMesh1D](#), [corenc::Mesh::CTriangularMesh](#), and [corenc::Mesh::CTriangularMeshLinear](#).

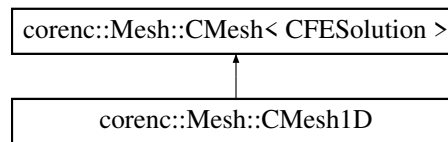
The documentation for this class was generated from the following file:

- [CoreNCFEM/Mesh.h](#)

6.27 corenc::Mesh::CMesh1D Class Reference

```
#include <Mesh1D.h>
```

Inheritance diagram for corenc::Mesh::CMesh1D:



Public Member Functions

- [CMesh1D](#) ()
- [CMesh1D](#) (const std::string &domain_name)
- [CMesh1D](#) (const std::string &domain_file, const std::string &init_file)
- [CMesh1D](#) (const double x0, const double x1, const unsigned n, const int order, const std::function< const double(const [Point](#) &)> &init_func)
- [CMesh1D](#) (const double x0, const double x1, const unsigned n, const int order, const std::function< const double(const [Point](#) &)> &init_func, const std::function< const double(const [Point](#) &)> &init_derivative)
- [CMesh1D](#) (const [CMesh1D](#) &)
- [CMesh1D](#) & operator= (const [CMesh1D](#) &m)
- const unsigned int [GetNumberOfElements](#) () const
- const unsigned int [GetNumberOfNodes](#) () const
- const unsigned int [GetNumberOfBoundaries](#) () const
- const int [FindElement](#) (const [Point](#) &) const
- const [Point](#) [GetNode](#) (const unsigned int) const
- const [CElement](#)< [CFESolution](#) > * [GetElement](#) (const unsigned int) const
- const [CElement](#)< [CFESolution](#) > * [GetBoundary](#) (const unsigned int) const
- const double [getSolution](#) (const unsigned int element, const unsigned int node) const
- const double [getParameter](#) ([Parameters](#), const unsigned int, const [Point](#) &p) const
- const double [getParameter](#) ([Parameters](#), const unsigned int, const int) const
- const std::vector< double > [getSolution](#) () const
- const int [updateSolution](#) (const std::vector< double > &new_solution)
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, const double value)
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, [CSolution](#) *value)
- const int [updateSolution](#) (const unsigned int node, const double value)
- const int [setParameter](#) ([Parameters](#), const double, const unsigned int)
- const double [getMinSize](#) () const
- [~CMesh1D](#) ()
- auto [GetElements](#) () -> decltype(m_elems)
- auto [GetBoundary](#) () -> decltype(m_bnds)

6.27.1 Constructor & Destructor Documentation

6.27.1.1 CMesh1D() [1/6]

```
CMesh1D::CMesh1D ( )
```

6.27.1.2 CMesh1D() [2/6]

```
CMesh1D::CMesh1D (
    const std::string & domain_name )
```

6.27.1.3 CMesh1D() [3/6]

```
CMesh1D::CMesh1D (
    const std::string & domain_file,
    const std::string & init_file )
```

6.27.1.4 CMesh1D() [4/6]

```
CMesh1D::CMesh1D (
    const double x0,
    const double x1,
    const unsigned n,
    const int order,
    const std::function< const double(const Point &)> & init_func )
```

6.27.1.5 CMesh1D() [5/6]

```
CMesh1D::CMesh1D (
    const double x0,
    const double x1,
    const unsigned n,
    const int order,
    const std::function< const double(const Point &)> & init_func,
    const std::function< const double(const Point &)> & init_derivative )
```

6.27.1.6 CMesh1D() [6/6]

```
CMesh1D::CMesh1D (
    const CMesh1D & m )
```

6.27.1.7 ~CMesh1D()

```
CMesh1D::~~CMesh1D ( )
```

6.27.2 Member Function Documentation

6.27.2.1 FindElement()

```
const int CMesh1D::FindElement (
    const Point & test ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.2 GetBoundary() [1/2]

```
auto corenc::Mesh::CMesh1D::GetBoundary ( ) -> decltype(m_bnds) [inline]
```

6.27.2.3 GetBoundary() [2/2]

```
const CElement< CFESolution > * CMesh1D::GetBoundary (
    const unsigned int n ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.4 GetElement()

```
const CElement< CFESolution > * CMesh1D::GetElement (
    const unsigned int n ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.5 GetElements()

```
auto corenc::Mesh::CMesh1D::GetElements ( ) -> decltype(m_elems) [inline]
```

6.27.2.6 getMinSize()

```
const double corenc::Mesh::CMesh1D::getMinSize ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.7 GetNode()

```
const Point CMesh1D::GetNode (
    const unsigned int n ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.8 GetNumberOfBoundaries()

```
const unsigned int CMesh1D::GetNumberOfBoundaries ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.9 GetNumberOfElements()

```
const unsigned int CMesh1D::GetNumberOfElements ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.10 GetNumberOfNodes()

```
const unsigned int CMesh1D::GetNumberOfNodes ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.11 getParameter() [1/2]

```
const double CMesh1D::getParameter (
    Parameters param,
    const unsigned int ,
    const int ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.12 `getParameter()` [2/2]

```
const double CMesh1D::getParameter (
    Parameters param,
    const unsigned int ,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.13 `getSolution()` [1/2]

```
const std::vector< double > corenc::Mesh::CMesh1D::getSolution ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.14 `getSolution()` [2/2]

```
const double CMesh1D::getSolution (
    const unsigned int element,
    const unsigned int node ) const [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.15 `operator=()`

```
CMesh1D & corenc::Mesh::CMesh1D::operator= (
    const CMesh1D & m ) [inline]
```

6.27.2.16 `setParameter()`

```
const int CMesh1D::setParameter (
    Parameters param,
    const double value,
    const unsigned int ) [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.17 updateSolution() [1/4]

```
const int corenc::Mesh::CMesh1D::updateSolution (
    const std::vector< double > & new_solution ) [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.18 updateSolution() [2/4]

```
const int CMesh1D::updateSolution (
    const unsigned int element,
    const unsigned int node,
    const double value ) [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.19 updateSolution() [3/4]

```
const int CMesh1D::updateSolution (
    const unsigned int element,
    const unsigned int node,
    CSolution * value ) [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

6.27.2.20 updateSolution() [4/4]

```
const int CMesh1D::updateSolution (
    const unsigned int node,
    const double value ) [virtual]
```

Implements [corenc::Mesh::CMesh< CFESolution >](#).

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

- CoreNCFEM/Grids/[Mesh1D.h](#)
- CoreNCFEM/Grids/[Mesh1D.cpp](#)

6.28 corenc::Mesh::CMesh< bool > Class Reference

```
#include <Mesh.h>
```


Public Member Functions

- [CMesh](#) ()
- virtual [~CMesh](#) ()
- virtual const unsigned int [GetNumberOfNodes](#) () const =0
- virtual const unsigned int [GetNumberOfElements](#) () const =0
- virtual const int [FindElement](#) (const [Point](#) &) const =0
- virtual const unsigned int [GetNumberOfBoundaries](#) () const =0
- virtual const [CElement](#) * [GetElement](#) (const unsigned int) const =0
- virtual const [CElement](#) * [GetBoundary](#) (const unsigned int) const =0
- virtual const [Point](#) [GetNode](#) (const unsigned int) const =0
- virtual const double [getSolution](#) (const unsigned int element, const unsigned int node) const =0
- virtual const int [updateSolution](#) (const unsigned int element, const unsigned int node, const double value)=0
- virtual const std::vector< double > [getSolution](#) () const =0
- virtual const int [updateSolution](#) (const std::vector< double > &)=0
- virtual const int [updateSolution](#) (const unsigned int element, const unsigned int node, [CSolution](#) *value)=0
- virtual const double [getParameter](#) ([Parameters](#), const unsigned int, const [Point](#) &p) const =0
- virtual const double [getParameter](#) ([Parameters](#), const unsigned int, const int) const =0
- virtual const int [setParameter](#) ([Parameters](#), const double, const unsigned int)=0
- virtual const double [getMinSize](#) () const =0

6.28.1 Constructor & Destructor Documentation

6.28.1.1 CMesh()

```
corenc::Mesh::CMesh< bool >::CMesh ( ) [inline]
```

6.28.1.2 ~CMesh()

```
virtual corenc::Mesh::CMesh< bool >::~~CMesh ( ) [inline], [virtual]
```

6.28.2 Member Function Documentation

6.28.2.1 FindElement()

```
virtual const int corenc::Mesh::CMesh< bool >::FindElement (
    const Point & ) const [pure virtual]
```

6.28.2.2 GetBoundary()

```
virtual const CElement * corenc::Mesh::CMesh< bool >::GetBoundary (
    const unsigned int ) const [pure virtual]
```

6.28.2.3 GetElement()

```
virtual const CElement * corenc::Mesh::CMesh< bool >::GetElement (
    const unsigned int ) const [pure virtual]
```

6.28.2.4 getMinSize()

```
virtual const double corenc::Mesh::CMesh< bool >::getMinSize ( ) const [pure virtual]
```

6.28.2.5 GetNode()

```
virtual const Point corenc::Mesh::CMesh< bool >::GetNode (
    const unsigned int ) const [pure virtual]
```

6.28.2.6 GetNumberOfBoundaries()

```
virtual const unsigned int corenc::Mesh::CMesh< bool >::GetNumberOfBoundaries ( ) const [pure
virtual]
```

6.28.2.7 GetNumberOfElements()

```
virtual const unsigned int corenc::Mesh::CMesh< bool >::GetNumberOfElements ( ) const [pure
virtual]
```

6.28.2.8 GetNumberOfNodes()

```
virtual const unsigned int corenc::Mesh::CMesh< bool >::GetNumberOfNodes ( ) const [pure
virtual]
```

6.28.2.9 getParameter() [1/2]

```
virtual const double corenc::Mesh::CMesh< bool >::getParameter (
    Parameters ,
    const unsigned int ,
    const int ) const [pure virtual]
```

6.28.2.10 getParameter() [2/2]

```
virtual const double corenc::Mesh::CMesh< bool >::getParameter (
    Parameters ,
    const unsigned int ,
    const Point & p ) const [pure virtual]
```

6.28.2.11 getSolution() [1/2]

```
virtual const std::vector< double > corenc::Mesh::CMesh< bool >::getSolution ( ) const [pure
virtual]
```

6.28.2.12 getSolution() [2/2]

```
virtual const double corenc::Mesh::CMesh< bool >::getSolution (
    const unsigned int element,
    const unsigned int node ) const [pure virtual]
```

6.28.2.13 setParameter()

```
virtual const int corenc::Mesh::CMesh< bool >::setParameter (
    Parameters ,
    const double ,
    const unsigned int ) [pure virtual]
```

6.28.2.14 updateSolution() [1/3]

```
virtual const int corenc::Mesh::CMesh< bool >::updateSolution (
    const std::vector< double > & ) [pure virtual]
```

6.28.2.15 updateSolution() [2/3]

```
virtual const int corenc::Mesh::CMesh< bool >::updateSolution (
    const unsigned int element,
    const unsigned int node,
    const double value ) [pure virtual]
```

6.28.2.16 updateSolution() [3/3]

```
virtual const int corenc::Mesh::CMesh< bool >::updateSolution (
    const unsigned int element,
    const unsigned int node,
    CSolution * value ) [pure virtual]
```

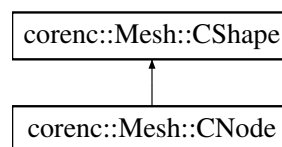
The documentation for this class was generated from the following file:

- CoreNCFEM/[Mesh.h](#)

6.29 corenc::Mesh::CNode Class Reference

```
#include <Node.h>
```

Inheritance diagram for corenc::Mesh::CNode:



Public Member Functions

- [CNode](#) ()
- [CNode](#) (const [CNode](#) &)
- [CNode](#) (const int n)
- [CNode](#) (const int *n)
- [CNode](#) & [operator=](#) (const [CNode](#) &e)
- [~CNode](#) ()
- const int [GetNode](#) (const int) const
- const int [GetNode](#) (const [NODES](#) &) const
- const int [IncreaseOrder](#) ()
- const int [GetNumberOfNodes](#) () const
- void [SetNode](#) (const int k, const int node)
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const

Friends

- const bool `operator==` (const `CNode` &e1, const `CNode` &e2)
- std::istream & `operator>>` (std::istream &is, `CNode` &e)

6.29.1 Constructor & Destructor Documentation

6.29.1.1 CNode() [1/4]

```
CNode::CNode ( )
```

6.29.1.2 CNode() [2/4]

```
CNode::CNode (
    const CNode & n )
```

6.29.1.3 CNode() [3/4]

```
corenc::Mesh::CNode::CNode (
    const int n )
```

6.29.1.4 CNode() [4/4]

```
CNode::CNode (
    const int * n )
```

6.29.1.5 ~CNode()

```
corenc::Mesh::CNode::~~CNode ( ) [inline]
```

6.29.2 Member Function Documentation

6.29.2.1 GetNode() [1/2]

```
const int CNode::GetNode (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.29.2.2 GetNode() [2/2]

```
const int CNode::GetNode (
    const NODES & node ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.29.2.3 GetNumberOfNodes()

```
const int CNode::GetNumberOfNodes ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.29.2.4 IncreaseOrder()

```
const int corenc::Mesh::CNode::IncreaseOrder ( ) [inline]
```

6.29.2.5 Integrate() [1/3]

```
const double CNode::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.29.2.6 Integrate() [2/3]

```
const Point CNode::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.29.2.7 Integrate() [3/3]

```
const std::vector< double > corenc::Mesh::CNode::Integrate (
    const std::function< const std::vector< double > (const Point &) > & ,
    const std::vector< Point > & ) const [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.29.2.8 operator=()

```
CNode & corenc::Mesh::CNode::operator= (
    const CNode & e ) [inline]
```

6.29.2.9 SetNode()

```
void CNode::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.29.3 Friends And Related Function Documentation

6.29.3.1 operator==

```
const bool operator== (
    const CNode & e1,
    const CNode & e2 ) [friend]
```

6.29.3.2 operator>>

```
std::istream & operator>> (
    std::istream & is,
    CNode & e ) [friend]
```

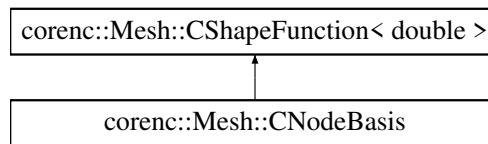
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Node.h](#)
- [CoreNCFEM/FiniteElements/Node.cpp](#)

6.30 corenc::Mesh::CNodeBasis Class Reference

```
#include <Node.h>
```

Inheritance diagram for corenc::Mesh::CNodeBasis:



Public Member Functions

- [CNodeBasis](#) ()
- [CNodeBasis](#) (const [Point](#) *)
- [CNodeBasis](#) (const [CNodeBasis](#) &e)
- [CNodeBasis](#) & [operator=](#) (const [CNodeBasis](#) &e)
- [~CNodeBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetWeight](#) (const int node, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const

6.30.1 Constructor & Destructor Documentation

6.30.1.1 CNodeBasis() [1/3]

```
CNodeBasis::CNodeBasis ( )
```

6.30.1.2 CNodeBasis() [2/3]

```
CNodeBasis::CNodeBasis (
    const Point * p )
```


6.30.1.3 CNodeBasis() [3/3]

```
corenc::Mesh::CNodeBasis::CNodeBasis (
    const CNodeBasis & e ) [inline]
```

6.30.1.4 ~CNodeBasis()

```
corenc::Mesh::CNodeBasis::~~CNodeBasis ( ) [inline]
```

6.30.2 Member Function Documentation

6.30.2.1 GetGradShapeFunction()

```
const Point CNodeBasis::GetGradShapeFunction (
    const int ,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.30.2.2 GetMeasure()

```
const double corenc::Mesh::CNodeBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.30.2.3 GetNormal()

```
const Point CNodeBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.30.2.4 GetNumberOfShapeFunctions()

```
const int CNodeBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.30.2.5 GetShapeFunction()

```
const double CNodeBasis::GetShapeFunction (
    const int ,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.30.2.6 GetWeight()

```
const double corenc::Mesh::CNodeBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const [inline]
```

6.30.2.7 IncreaseOrder()

```
const int corenc::Mesh::CNodeBasis::IncreaseOrder ( ) [inline]
```

6.30.2.8 operator=()

```
CNodeBasis & corenc::Mesh::CNodeBasis::operator= (
    const CNodeBasis & e ) [inline]
```

6.30.2.9 ReverseNormal()

```
void CNodeBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

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

- [CoreNCFEM/FiniteElements/Node.h](#)
- [CoreNCFEM/FiniteElements/Node.cpp](#)

6.31 corenc::Mesh::CParameter Class Reference

```
#include <Parameter.h>
```

Public Member Functions

- [CParameter](#) ()
- [CParameter](#) (const [parameter](#)< double > &_diff, const [parameter](#)< double > &_adv, const [parameter](#)< double > &_mass)
- [CParameter](#) (const [Parameters](#) &, const [parameter](#)< double > &)
- [~CParameter](#) ()
- const double [GetDiffusion](#) () const
- const double [GetAdvection](#) () const
- const double [GetMass](#) () const
- const double [GetDiffusion](#) (const [Point](#) &) const
- const double [GetAdvection](#) (const [Point](#) &) const
- const double [GetMass](#) (const [Point](#) &) const

6.31.1 Constructor & Destructor Documentation

6.31.1.1 CParameter() [1/3]

```
CParameter::CParameter ( )
```

6.31.1.2 CParameter() [2/3]

```
CParameter::CParameter (
    const parameter< double > & _diff,
    const parameter< double > & _adv,
    const parameter< double > & _mass )
```

6.31.1.3 CParameter() [3/3]

```
CParameter::CParameter (
    const Parameters & type,
    const parameter< double > & p )
```

6.31.1.4 ~CParameter()

```
CParameter::~~CParameter ( )
```

6.31.2 Member Function Documentation

6.31.2.1 GetAdvection() [1/2]

```
const double CParameter::GetAdvection ( ) const
```

6.31.2.2 GetAdvection() [2/2]

```
const double CParameter::GetAdvection (
    const Point & p ) const
```

6.31.2.3 GetDiffusion() [1/2]

```
const double CParameter::GetDiffusion ( ) const
```

6.31.2.4 GetDiffusion() [2/2]

```
const double CParameter::GetDiffusion (
    const Point & p ) const
```

6.31.2.5 GetMass() [1/2]

```
const double CParameter::GetMass ( ) const
```

6.31.2.6 GetMass() [2/2]

```
const double CParameter::GetMass (
    const Point & p ) const
```

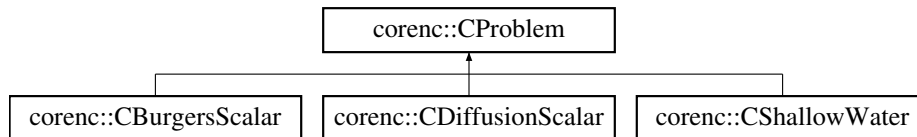
The documentation for this class was generated from the following files:

- [CoreNCFEM/Parameter.h](#)
- [CoreNCFEM/Parameter.cpp](#)

6.32 corenc::CProblem Class Reference

```
#include <Problems.h>
```

Inheritance diagram for corenc::CProblem:



Public Member Functions

- [CProblem](#) ()
- virtual [~CProblem](#) ()
- virtual [Terms](#) [getTerm](#) (const unsigned int) const =0
- virtual const unsigned int [getNumberOfTerms](#) () const =0
- virtual const int [setTerm](#) (const unsigned int, const [Terms](#) &)=0
- virtual const int [addTerm](#) (const [Terms](#) &)=0
- virtual const int [load_parameters](#) (const std::string &file_name)=0

6.32.1 Constructor & Destructor Documentation

6.32.1.1 CProblem()

```
corenc::CProblem::CProblem ( ) [inline]
```

6.32.1.2 ~CProblem()

```
virtual corenc::CProblem::~~CProblem ( ) [inline], [virtual]
```

6.32.2 Member Function Documentation

6.32.2.1 addTerm()

```
virtual const int corenc::CProblem::addTerm (
    const Terms & ) [pure virtual]
```

Implemented in [corenc::CBurgersScalar](#), [corenc::CDiffusionScalar](#), and [corenc::CShallowWater](#).

6.32.2.2 getNumberOfTerms()

```
virtual const unsigned int corenc::CProblem::getNumberOfTerms ( ) const [pure virtual]
```

Implemented in [corenc::CBurgersScalar](#), [corenc::CDiffusionScalar](#), and [corenc::CShallowWater](#).

6.32.2.3 getTerm()

```
virtual Terms corenc::CProblem::getTerm (
    const unsigned int ) const [pure virtual]
```

Implemented in [corenc::CBurgersScalar](#), [corenc::CDiffusionScalar](#), and [corenc::CShallowWater](#).

6.32.2.4 load_parameters()

```
virtual const int corenc::CProblem::load_parameters (
    const std::string & file_name ) [pure virtual]
```

Implemented in [corenc::CBurgersScalar](#), [corenc::CDiffusionScalar](#), and [corenc::CShallowWater](#).

6.32.2.5 setTerm()

```
virtual const int corenc::CProblem::setTerm (
    const unsigned int ,
    const Terms & ) [pure virtual]
```

Implemented in [corenc::CBurgersScalar](#), [corenc::CDiffusionScalar](#), and [corenc::CShallowWater](#).

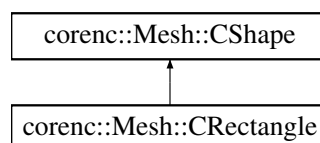
The documentation for this class was generated from the following file:

- Problems/[Problems.h](#)

6.33 corenc::Mesh::CRectangle Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for corenc::Mesh::CRectangle:



Public Member Functions

- [CRectangle](#) ()
- [CRectangle](#) (const int n1, const int n2, const int n3, const int n4, const int order)
- [CRectangle](#) (const int n1, const int n2, const int n3, const int n4, const int e1, const int e2, const int e3, const int e4, const int order)
- [CRectangle](#) (const int *, const int order)
- [CRectangle](#) (const int *, const int *, const int order)
- [CRectangle](#) (const [CRectangle](#) &)
- [CRectangle](#) & [operator=](#) (const [CRectangle](#) &t)
- const bool [operator==](#) (const [CRectangle](#) &t)
- std::istream & [operator>>](#) (std::istream &is)
- [~CRectangle](#) ()
- const int [GetNode](#) (const int) const
- const int [GetNode](#) (const [NODES](#) &) const
- const int [GetEdge](#) (const int) const
- const int [GetFacet](#) (const int) const
- const int [GetNumberOfNodes](#) () const
- const int [GetNumberOfEdges](#) () const
- const int [GetNumberOfFacets](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- void [SetNode](#) (const int k, const int node)
- const int [IncreaseOrder](#) ()
- const int [SetOrder](#) (const int px, const int py)
- void [SetEdge](#) (const int k, const int edge)
- void [SetFacet](#) (const int k, const int facet)

6.33.1 Constructor & Destructor Documentation

6.33.1.1 CRectangle() [1/6]

```
CRectangle::CRectangle ( )
```

6.33.1.2 CRectangle() [2/6]

```
CRectangle::CRectangle (
    const int n1,
    const int n2,
    const int n3,
    const int n4,
    const int order )
```

6.33.1.3 CRectangle() [3/6]

```
CRectangle::CRectangle (
    const int n1,
    const int n2,
    const int n3,
    const int n4,
    const int e1,
    const int e2,
    const int e3,
    const int e4,
    const int order )
```

6.33.1.4 CRectangle() [4/6]

```
CRectangle::CRectangle (
    const int * nodes,
    const int order )
```

6.33.1.5 CRectangle() [5/6]

```
CRectangle::CRectangle (
    const int * nodes,
    const int * edges,
    const int order )
```

6.33.1.6 CRectangle() [6/6]

```
CRectangle::CRectangle (
    const CRectangle & t )
```

6.33.1.7 ~CRectangle()

```
corenc::Mesh::CRectangle::~~CRectangle ( ) [inline]
```

6.33.2 Member Function Documentation

6.33.2.1 GetEdge()

```
const int CRectangle::GetEdge (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.2 GetFacet()

```
const int CRectangle::GetFacet (
    const int ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.3 GetNode() [1/2]

```
const int CRectangle::GetNode (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.4 GetNode() [2/2]

```
const int CRectangle::GetNode (
    const NODES & node ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.5 GetNumberOfEdges()

```
const int CRectangle::GetNumberOfEdges ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.6 GetNumberOfFacets()

```
const int CRectangle::GetNumberOfFacets ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.7 GetNumberOfNodes()

```
const int CRectangle::GetNumberOfNodes ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.8 IncreaseOrder()

```
const int CRectangle::IncreaseOrder ( )
```

6.33.2.9 Integrate() [1/3]

```
const double CRectangle::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.33.2.10 Integrate() [2/3]

```
const Point CRectangle::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.33.2.11 Integrate() [3/3]

```
const vector< double > CRectangle::Integrate (
    const std::function< const std::vector< double >(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.33.2.12 operator=()

```
CRectangle & corenc::Mesh::CRectangle::operator= (
    const CRectangle & t ) [inline]
```

6.33.2.13 operator==()

```
const bool corenc::Mesh::CRectangle::operator== (
    const CRectangle & t ) [inline]
```

6.33.2.14 operator>>()

```
std::istream & corenc::Mesh::CRectangle::operator>> (
    std::istream & is ) [inline]
```

6.33.2.15 SetEdge()

```
void CRectangle::SetEdge (
    const int k,
    const int edge ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.16 SetFacet()

```
void CRectangle::SetFacet (
    const int k,
    const int facet ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.33.2.17 SetNode()

```
void CRectangle::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.33.2.18 SetOrder()

```
const int CRectangle::SetOrder (
    const int px,
    const int py )
```

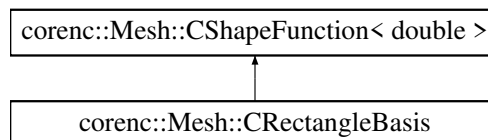
The documentation for this class was generated from the following files:

- CoreNCFEM/FiniteElements/[Rectangle.h](#)
- CoreNCFEM/FiniteElements/[Rectangle.cpp](#)

6.34 corenc::Mesh::CRectangleBasis Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for corenc::Mesh::CRectangleBasis:



Public Member Functions

- [CRectangleBasis](#) ()
- [CRectangleBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CRectangleBasis](#) (const [Point](#) *, const int order)
- [CRectangleBasis](#) (const [CRectangleBasis](#) &)
- [CRectangleBasis](#) & [operator=](#) (const [CRectangleBasis](#) &t)
- [~CRectangleBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.34.1 Constructor & Destructor Documentation

6.34.1.1 CRectangleBasis() [1/4]

```
CRectangleBasis::CRectangleBasis ( )
```

6.34.1.2 CRectangleBasis() [2/4]

```
CRectangleBasis::CRectangleBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int order )
```

6.34.1.3 CRectangleBasis() [3/4]

```
CRectangleBasis::CRectangleBasis (
    const Point * p,
    const int order )
```

6.34.1.4 CRectangleBasis() [4/4]

```
CRectangleBasis::CRectangleBasis (
    const CRectangleBasis & t )
```

6.34.1.5 ~CRectangleBasis()

```
corenc::Mesh::CRectangleBasis::~~CRectangleBasis ( ) [inline]
```

6.34.2 Member Function Documentation**6.34.2.1 GetGradShapeFunction()**

```
const Point CRectangleBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.34.2.2 GetMeasure()

```
const double corenc::Mesh::CRectangleBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.34.2.3 GetNormal()

```
const Point CRectangleBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.34.2.4 GetNumberOfShapeFunctions()

```
const int CRectangleBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.34.2.5 GetShapeFunction()

```
const double CRectangleBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.34.2.6 GetValue()

```
const double CRectangleBasis::GetValue (
    const Point & p ) const
```

6.34.2.7 GetWeight()

```
const double CRectangleBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.34.2.8 IncreaseOrder()

```
const int CRectangleBasis::IncreaseOrder ( )
```

6.34.2.9 operator=()

```
CRectangleBasis & corenc::Mesh::CRectangleBasis::operator= (
    const CRectangleBasis & t ) [inline]
```

6.34.2.10 ReverseNormal()

```
void CRectangleBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

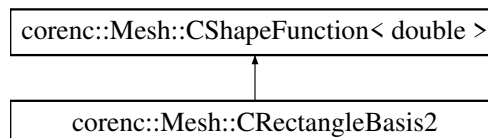
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Rectangle.h](#)
- [CoreNCFEM/FiniteElements/Rectangle.cpp](#)

6.35 corenc::Mesh::CRectangleBasis2 Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for corenc::Mesh::CRectangleBasis2:



Public Member Functions

- [CRectangleBasis2](#) ()
- [CRectangleBasis2](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CRectangleBasis2](#) (const [Point](#) *, const int order)
- [CRectangleBasis2](#) (const [CRectangleBasis2](#) &)
- [CRectangleBasis2](#) & [operator=](#) (const [CRectangleBasis2](#) &t)
- [~CRectangleBasis2](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.35.1 Constructor & Destructor Documentation

6.35.1.1 CRectangleBasis2() [1/4]

```
CRectangleBasis2::CRectangleBasis2 ( )
```

6.35.1.2 CRectangleBasis2() [2/4]

```
CRectangleBasis2::CRectangleBasis2 (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int order )
```

6.35.1.3 CRectangleBasis2() [3/4]

```
CRectangleBasis2::CRectangleBasis2 (
    const Point * p,
    const int order )
```

6.35.1.4 CRectangleBasis2() [4/4]

```
CRectangleBasis2::CRectangleBasis2 (
    const CRectangleBasis2 & t )
```

6.35.1.5 ~CRectangleBasis2()

```
corenc::Mesh::CRectangleBasis2::~~CRectangleBasis2 ( ) [inline]
```

6.35.2 Member Function Documentation

6.35.2.1 GetGradShapeFunction()

```
const Point CRectangleBasis2::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.35.2.2 GetMeasure()

```
const double corenc::Mesh::CRectangleBasis2::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.35.2.3 GetNormal()

```
const Point CRectangleBasis2::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.35.2.4 GetNumberOfShapeFunctions()

```
const int CRectangleBasis2::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.35.2.5 GetShapeFunction()

```
const double CRectangleBasis2::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.35.2.6 GetValue()

```
const double CRectangleBasis2::GetValue (
    const Point & p ) const
```

6.35.2.7 GetWeight()

```
const double CRectangleBasis2::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.35.2.8 IncreaseOrder()

```
const int CRectangleBasis2::IncreaseOrder ( )
```

6.35.2.9 operator=()

```
CRectangleBasis2 & corenc::Mesh::CRectangleBasis2::operator= (
    const CRectangleBasis2 & t ) [inline]
```

6.35.2.10 ReverseNormal()

```
void CRectangleBasis2::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

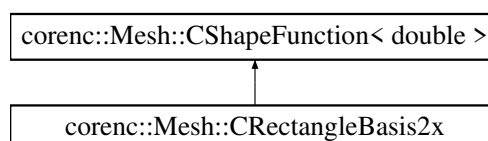
The documentation for this class was generated from the following files:

- CoreNCFEM/FiniteElements/[Rectangle.h](#)
- CoreNCFEM/FiniteElements/[RectangleBasis2.cpp](#)

6.36 corenc::Mesh::CRectangleBasis2x Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for corenc::Mesh::CRectangleBasis2x:



Public Member Functions

- [CRectangleBasis2x](#) ()
- [CRectangleBasis2x](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CRectangleBasis2x](#) (const [Point](#) *, const int order)
- [CRectangleBasis2x](#) (const [CRectangleBasis2x](#) &)
- [CRectangleBasis2x](#) & [operator=](#) (const [CRectangleBasis2x](#) &t)
- [~CRectangleBasis2x](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.36.1 Constructor & Destructor Documentation

6.36.1.1 CRectangleBasis2x() [1/4]

```
CRectangleBasis2x::CRectangleBasis2x ( )
```

6.36.1.2 CRectangleBasis2x() [2/4]

```
CRectangleBasis2x::CRectangleBasis2x (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int order )
```

6.36.1.3 CRectangleBasis2x() [3/4]

```
CRectangleBasis2x::CRectangleBasis2x (
    const Point * p,
    const int order )
```

6.36.1.4 CRectangleBasis2x() [4/4]

```
CRectangleBasis2x::CRectangleBasis2x (
    const CRectangleBasis2x & t )
```

6.36.1.5 ~CRectangleBasis2x()

```
corenc::Mesh::CRectangleBasis2x::~CRectangleBasis2x ( ) [inline]
```

6.36.2 Member Function Documentation

6.36.2.1 GetGradShapeFunction()

```
const Point CRectangleBasis2x::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.36.2.2 GetMeasure()

```
const double corenc::Mesh::CRectangleBasis2x::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.36.2.3 GetNormal()

```
const Point CRectangleBasis2x::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.36.2.4 GetNumberOfShapeFunctions()

```
const int CRectangleBasis2x::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.36.2.5 GetShapeFunction()

```
const double CRectangleBasis2x::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.36.2.6 GetValue()

```
const double CRectangleBasis2x::GetValue (
    const Point & p ) const
```

6.36.2.7 GetWeight()

```
const double CRectangleBasis2x::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.36.2.8 IncreaseOrder()

```
const int CRectangleBasis2x::IncreaseOrder ( )
```

6.36.2.9 operator=()

```
CRectangleBasis2x & corenc::Mesh::CRectangleBasis2x::operator= (
    const CRectangleBasis2x & t ) [inline]
```

6.36.2.10 ReverseNormal()

```
void CRectangleBasis2x::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

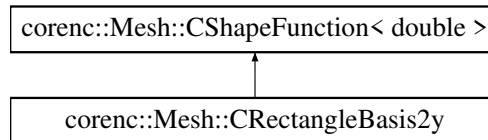
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Rectangle.h](#)
- [CoreNCFEM/FiniteElements/CRectangleBasis2x.cpp](#)

6.37 corenc::Mesh::CRectangleBasis2y Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for corenc::Mesh::CRectangleBasis2y:



Public Member Functions

- [CRectangleBasis2y](#) ()
- [CRectangleBasis2y](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CRectangleBasis2y](#) (const [Point](#) *, const int order)
- [CRectangleBasis2y](#) (const [CRectangleBasis2y](#) &)
- [CRectangleBasis2y](#) & [operator=](#) (const [CRectangleBasis2y](#) &t)
- [~CRectangleBasis2y](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.37.1 Constructor & Destructor Documentation

6.37.1.1 CRectangleBasis2y() [1/4]

```
CRectangleBasis2y::CRectangleBasis2y ( )
```

6.37.1.2 CRectangleBasis2y() [2/4]

```
CRectangleBasis2y::CRectangleBasis2y (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int order )
```

6.37.1.3 CRectangleBasis2y() [3/4]

```
CRectangleBasis2y::CRectangleBasis2y (
    const Point * p,
    const int order )
```

6.37.1.4 CRectangleBasis2y() [4/4]

```
CRectangleBasis2y::CRectangleBasis2y (
    const CRectangleBasis2y & t )
```

6.37.1.5 ~CRectangleBasis2y()

```
corenc::Mesh::CRectangleBasis2y::~~CRectangleBasis2y ( ) [inline]
```

6.37.2 Member Function Documentation

6.37.2.1 GetGradShapeFunction()

```
const Point CRectangleBasis2y::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.37.2.2 GetMeasure()

```
const double corenc::Mesh::CRectangleBasis2y::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.37.2.3 GetNormal()

```
const Point CRectangleBasis2y::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.37.2.4 GetNumberOfShapeFunctions()

```
const int CRectangleBasis2y::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.37.2.5 GetShapeFunction()

```
const double CRectangleBasis2y::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.37.2.6 GetValue()

```
const double CRectangleBasis2y::GetValue (
    const Point & p ) const
```

6.37.2.7 GetWeight()

```
const double CRectangleBasis2y::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.37.2.8 IncreaseOrder()

```
const int CRectangleBasis2y::IncreaseOrder ( )
```

6.37.2.9 operator=()

```
CRectangleBasis2y & corenc::Mesh::CRectangleBasis2y::operator= (
    const CRectangleBasis2y & t ) [inline]
```


6.37.2.10 ReverseNormal()

```
void CRectangleBasis2y::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

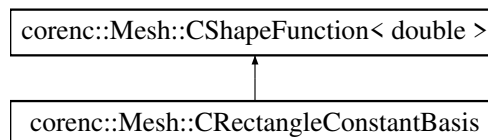
The documentation for this class was generated from the following files:

- CoreNCFEM/FiniteElements/[Rectangle.h](#)
- CoreNCFEM/FiniteElements/[RectangleBasis2y.cpp](#)

6.38 corenc::Mesh::CRectangleConstantBasis Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for corenc::Mesh::CRectangleConstantBasis:



Public Member Functions

- [CRectangleConstantBasis](#) ()
- [CRectangleConstantBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CRectangleConstantBasis](#) (const [Point](#) *, const int order)
- [CRectangleConstantBasis](#) (const [CRectangleConstantBasis](#) &)
- [CRectangleConstantBasis](#) & operator= (const [CRectangleConstantBasis](#) &t)
- [~CRectangleConstantBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const

6.38.1 Constructor & Destructor Documentation

6.38.1.1 CRectangleConstantBasis() [1/4]

```
CRectangleConstantBasis::CRectangleConstantBasis ( )
```

6.38.1.2 CRectangleConstantBasis() [2/4]

```
CRectangleConstantBasis::CRectangleConstantBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int order )
```

6.38.1.3 CRectangleConstantBasis() [3/4]

```
CRectangleConstantBasis::CRectangleConstantBasis (
    const Point * p,
    const int order )
```

6.38.1.4 CRectangleConstantBasis() [4/4]

```
CRectangleConstantBasis::CRectangleConstantBasis (
    const CRectangleConstantBasis & t )
```

6.38.1.5 ~CRectangleConstantBasis()

```
corenc::Mesh::CRectangleConstantBasis::~~CRectangleConstantBasis ( ) [inline]
```

6.38.2 Member Function Documentation**6.38.2.1 GetGradShapeFunction()**

```
const Point CRectangleConstantBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.38.2.2 GetMeasure()

```
const double corenc::Mesh::CRectangleConstantBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.38.2.3 GetNormal()

```
const Point CRectangleConstantBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.38.2.4 GetNumberOfShapeFunctions()

```
const int CRectangleConstantBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.38.2.5 GetShapeFunction()

```
const double CRectangleConstantBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.38.2.6 GetValue()

```
const double CRectangleConstantBasis::GetValue (
    const Point & p ) const
```

6.38.2.7 IncreaseOrder()

```
const int CRectangleConstantBasis::IncreaseOrder ( )
```

6.38.2.8 operator=()

```
CRectangleConstantBasis & corenc::Mesh::CRectangleConstantBasis::operator= (
    const CRectangleConstantBasis & t ) [inline]
```

6.38.2.9 ReverseNormal()

```
void CRectangleConstantBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

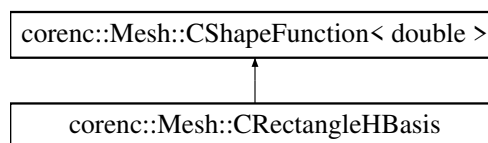
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Rectangle.h](#)
- [CoreNCFEM/FiniteElements/Rectangle.cpp](#)

6.39 corenc::Mesh::CRectangleHBasis Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for `corenc::Mesh::CRectangleHBasis`:



Public Member Functions

- [CRectangleHBasis](#) ()
- [CRectangleHBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CRectangleHBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const [Point](#) &, const int px, const int py)
- [CRectangleHBasis](#) (const [Point](#) *, const int order)
- [CRectangleHBasis](#) (const [Point](#) *, const int px, const int py)
- [CRectangleHBasis](#) (const [CRectangleHBasis](#) &)
- [CRectangleHBasis](#) & [operator=](#) (const [CRectangleHBasis](#) &t)
- [~CRectangleHBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const int [SetOrder](#) (const int px, const int py)
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.39.1 Constructor & Destructor Documentation

6.39.1.1 CRectangleHBasis() [1/6]

```
CRectangleHBasis::CRectangleHBasis ( )
```

6.39.1.2 CRectangleHBasis() [2/6]

```
CRectangleHBasis::CRectangleHBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int order )
```

6.39.1.3 CRectangleHBasis() [3/6]

```
CRectangleHBasis::CRectangleHBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const Point & p4,
    const int px,
    const int py )
```

6.39.1.4 CRectangleHBasis() [4/6]

```
CRectangleHBasis::CRectangleHBasis (
    const Point * p,
    const int order )
```

6.39.1.5 CRectangleHBasis() [5/6]

```
CRectangleHBasis::CRectangleHBasis (
    const Point * p,
    const int px,
    const int py )
```

6.39.1.6 CRectangleHBasis() [6/6]

```
CRectangleHBasis::CRectangleHBasis (
    const CRectangleHBasis & t )
```

6.39.1.7 ~CRectangleHBasis()

```
corenc::Mesh::CRectangleHBasis::~~CRectangleHBasis ( ) [inline]
```

6.39.2 Member Function Documentation

6.39.2.1 GetGradShapeFunction()

```
const Point CRectangleHBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.39.2.2 GetMeasure()

```
const double corenc::Mesh::CRectangleHBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.39.2.3 GetNormal()

```
const Point CRectangleHBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.39.2.4 GetNumberOfShapeFunctions()

```
const int CRectangleHBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.39.2.5 GetShapeFunction()

```
const double CRectangleHBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.39.2.6 GetValue()

```
const double CRectangleHBasis::GetValue (
    const Point & p ) const
```

6.39.2.7 GetWeight()

```
const double CRectangleHBasis::GetWeight (
    const int node,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.39.2.8 IncreaseOrder()

```
const int CRectangleHBasis::IncreaseOrder ( )
```

6.39.2.9 operator=()

```
CRectangleHBasis & corenc::Mesh::CRectangleHBasis::operator= (
    const CRectangleHBasis & t ) [inline]
```

6.39.2.10 ReverseNormal()

```
void CRectangleHBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.39.2.11 SetOrder()

```
const int CRectangleHBasis::SetOrder (
    const int px,
    const int py )
```

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

- CoreNCFEM/FiniteElements/[Rectangle.h](#)
- CoreNCFEM/FiniteElements/[RectangleHBasis.cpp](#)

6.40 corenc::Mesh::CRegularMesh Class Reference

```
#include <RegularMesh.h>
```

Public Member Functions

- [CRegularMesh](#) ()
- [CRegularMesh](#) (const std::string &file_name)
- [CRegularMesh](#) (const [CRegularMesh](#) &)
- [CRegularMesh](#) (const [Point](#) &p1, const [Point](#) &p2, const int nx, const int ny)
- [CRegularMesh](#) (const [Point](#) &p1, const [Point](#) &p2, const int nx, const int ny, const int px, const int py)
- [CRegularMesh](#) (const double x1, const double y1, const double x2, const double y2, const int nx, const int ny)
- [CRegularMesh](#) & operator= (const [CRegularMesh](#) &tr)
- [CRegularMesh](#) * [Clone](#) () const
- const unsigned int [GetNumberOfElements](#) () const
- const unsigned int [GetNumberOfNodes](#) () const
- const int [GetNumberOfNodes](#) () const
- const unsigned int [GetNumberOfBoundaries](#) () const
- const int [FindElement](#) (const [Point](#) &) const
- const [Point](#) [GetNode](#) (const unsigned int) const
- const [CElement2D](#) * [GetElement](#) (const unsigned int) const
- const [CElement](#) * [GetBoundary](#) (const unsigned int) const
- const double [getMinSize](#) () const
- const double [getSolution](#) (const unsigned int element, const unsigned int node) const
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, const double value)
- const std::vector< double > [getSolution](#) () const
- const int [updateSolution](#) (const std::vector< double > &)
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, [CSolution](#) *value)
- const double [getParameter](#) ([Parameters](#), const unsigned int, const [Point](#) &) const
- const double [getParameter](#) ([Parameters](#), const unsigned int, const int) const
- const int [setParameter](#) ([Parameters](#), const double, const unsigned int)
- const int [setParameter](#) (const [CParameter](#) &, const unsigned int type)
- const int [updateSolution](#) (const unsigned int node, const double value)
- const int [refine_hx](#) ()
- const int [refine_hy](#) ()
- const int [refine_h](#) ()
- const int [refine_p](#) ()
- const int [refine_hp](#) ()
- const int [interpolate](#) (const int node) const
- [~CRegularMesh](#) ()
- auto [GetElements](#) () -> decltype(m_elems)
- auto [GetBoundary](#) () -> decltype(m_edges)

6.40.1 Constructor & Destructor Documentation

6.40.1.1 CRegularMesh() [1/6]

```
CRegularMesh::CRegularMesh ( )
```

6.40.1.2 CRegularMesh() [2/6]

```
corenc::Mesh::CRegularMesh::CRegularMesh (
    const std::string & file_name )
```

6.40.1.3 CRegularMesh() [3/6]

```
CRegularMesh::CRegularMesh (
    const CRegularMesh & tr )
```

6.40.1.4 CRegularMesh() [4/6]

```
CRegularMesh::CRegularMesh (
    const Point & p1,
    const Point & p2,
    const int nx,
    const int ny )
```

6.40.1.5 CRegularMesh() [5/6]

```
CRegularMesh::CRegularMesh (
    const Point & p1,
    const Point & p2,
    const int nx,
    const int ny,
    const int px,
    const int py )
```

6.40.1.6 CRegularMesh() [6/6]

```
CRegularMesh::CRegularMesh (
    const double x1,
    const double y1,
    const double x2,
    const double y2,
    const int nx,
    const int ny )
```

6.40.1.7 ~CRegularMesh()

```
CRegularMesh::~~CRegularMesh ( )
```

6.40.2 Member Function Documentation

6.40.2.1 Clone()

```
CRegularMesh * corenc::Mesh::CRegularMesh::Clone ( ) const [inline]
```

6.40.2.2 FindElement()

```
const int CRegularMesh::FindElement (
    const Point & test ) const
```

6.40.2.3 GetBoundary() [1/2]

```
auto corenc::Mesh::CRegularMesh::GetBoundary ( ) -> decltype(m_edges) [inline]
```

6.40.2.4 GetBoundary() [2/2]

```
const CElement * CRegularMesh::GetBoundary (
    const unsigned int n ) const
```

6.40.2.5 GetElement()

```
const CElement2D * CRegularMesh::GetElement (
    const unsigned int n ) const
```

6.40.2.6 GetElements()

```
auto corenc::Mesh::CRegularMesh::GetElements ( ) -> decltype(m_elems)    [inline]
```

6.40.2.7 getMinSize()

```
const double corenc::Mesh::CRegularMesh::getMinSize ( ) const    [inline]
```

6.40.2.8 GetNode()

```
const Mesh::Point CRegularMesh::GetNode (
    const unsigned int n ) const
```

6.40.2.9 GetNumberOfBoundaries()

```
const unsigned int CRegularMesh::GetNumberOfBoundaries ( ) const
```

6.40.2.10 GetNumberOfElements()

```
const unsigned int CRegularMesh::GetNumberOfElements ( ) const
```

6.40.2.11 GetNumberOfINodes()

```
const int CRegularMesh::GetNumberOfINodes ( ) const
```

6.40.2.12 GetNumberOfNodes()

```
const unsigned int CRegularMesh::GetNumberOfNodes ( ) const
```

6.40.2.13 getParameter() [1/2]

```
const double CRegularMesh::getParameter (
    Parameters param,
    const unsigned int l,
    const int i ) const
```

6.40.2.14 getParameter() [2/2]

```
const double CRegularMesh::getParameter (
    Parameters param,
    const unsigned int l,
    const Point & p ) const
```

6.40.2.15 getSolution() [1/2]

```
const std::vector< double > CRegularMesh::getSolution ( ) const
```

6.40.2.16 getSolution() [2/2]

```
const double CRegularMesh::getSolution (
    const unsigned int element,
    const unsigned int node ) const
```

6.40.2.17 interpolate()

```
const int CRegularMesh::interpolate (
    const int node ) const
```

6.40.2.18 operator=()

```
CRegularMesh & corenc::Mesh::CRegularMesh::operator= (
    const CRegularMesh & tr ) [inline]
```

6.40.2.19 refine_h()

```
const int CRegularMesh::refine_h ( )
```

6.40.2.20 refine_hp()

```
const int corenc::Mesh::CRegularMesh::refine_hp ( )
```

6.40.2.21 refine_hx()

```
const int corenc::Mesh::CRegularMesh::refine_hx ( )
```

6.40.2.22 refine_hy()

```
const int corenc::Mesh::CRegularMesh::refine_hy ( )
```

6.40.2.23 refine_p()

```
const int CRegularMesh::refine_p ( )
```

6.40.2.24 setParameter() [1/2]

```
const int CRegularMesh::setParameter (
    const CParameter & p,
    const unsigned int type )
```

6.40.2.25 setParameter() [2/2]

```
const int CRegularMesh::setParameter (
    Parameters param,
    const double ,
    const unsigned int )
```

6.40.2.26 updateSolution() [1/4]

```
const int CRegularMesh::updateSolution (
    const std::vector< double > & )
```

6.40.2.27 updateSolution() [2/4]

```
const int CRegularMesh::updateSolution (
    const unsigned int element,
    const unsigned int node,
    const double value )
```

6.40.2.28 updateSolution() [3/4]

```
const int CRegularMesh::updateSolution (
    const unsigned int element,
    const unsigned int node,
    CSolution * value )
```

6.40.2.29 updateSolution() [4/4]

```
const int CRegularMesh::updateSolution (
    const unsigned int node,
    const double value )
```

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

- CoreNCFEM/Grids/[RegularMesh.h](#)
- CoreNCFEM/Grids/[RegularMesh.cpp](#)

6.41 corenc::Mesh::CRegularMesh3D Class Reference

```
#include <RegularMesh3D.h>
```

Public Member Functions

- [CRegularMesh3D](#) ()
- [CRegularMesh3D](#) (const std::string &file_name)
- [CRegularMesh3D](#) (const [CRegularMesh3D](#) &)
- [CRegularMesh3D](#) (const [Point](#) &p1, const [Point](#) &p2, const int nx, const int ny)
- [CRegularMesh3D](#) (const [Point](#) &p1, const [Point](#) &p2, const int nx, const int ny, const int px, const int py)
- [CRegularMesh3D](#) (const double x1, const double y1, const double x2, const double y2, const int nx, const int ny)
- [CRegularMesh3D](#) & [operator=](#) (const [CRegularMesh3D](#) &tr)
- [CRegularMesh3D](#) * [Clone](#) () const
- const unsigned int [GetNumberOfElements](#) () const
- const unsigned int [GetNumberOfNodes](#) () const
- const int [GetNumberOfNodes](#) () const
- const unsigned int [GetNumberOfBoundaries](#) () const
- const int [FindElement](#) (const [Point](#) &) const
- const [Point](#) [GetNode](#) (const unsigned int) const
- const [CElement](#) * [GetElement](#) (const unsigned int) const
- const [CElement](#) * [GetBoundary](#) (const unsigned int) const
- const double [getMinSize](#) () const
- const double [getSolution](#) (const unsigned int element, const unsigned int node) const
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, const double value)
- const std::vector< double > [getSolution](#) () const
- const int [updateSolution](#) (const std::vector< double > &)
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, [CSolution](#) *value)
- const double [getParameter](#) ([Parameters](#), const unsigned int, const [Point](#) &) const
- const double [getParameter](#) ([Parameters](#), const unsigned int, const int) const
- const int [setParameter](#) ([Parameters](#), const double, const unsigned int)
- const int [setParameter](#) (const [CParameter](#) &, const unsigned int type)
- const int [updateSolution](#) (const unsigned int node, const double value)
- const int [refine_hx](#) ()
- const int [refine_hy](#) ()
- const int [refine_h](#) ()
- const int [refine_p](#) ()
- const int [refine_hp](#) ()
- const int [interpolate](#) (const int node) const
- [~CRegularMesh3D](#) ()
- auto [GetElements](#) () -> decltype(m_elems)
- auto [GetBoundary](#) () -> decltype(m_edges)

6.41.1 Constructor & Destructor Documentation

6.41.1.1 CRegularMesh3D() [1/6]

`CRegularMesh3D::CRegularMesh3D ()`

6.41.1.2 CRegularMesh3D() [2/6]

```
corenc::Mesh::CRegularMesh3D::CRegularMesh3D (
    const std::string & file_name )
```

6.41.1.3 CRegularMesh3D() [3/6]

```
CRegularMesh3D::CRegularMesh3D (
    const CRegularMesh3D & tr )
```

6.41.1.4 CRegularMesh3D() [4/6]

```
CRegularMesh3D::CRegularMesh3D (
    const Point & p1,
    const Point & p2,
    const int nx,
    const int ny )
```

6.41.1.5 CRegularMesh3D() [5/6]

```
CRegularMesh3D::CRegularMesh3D (
    const Point & p1,
    const Point & p2,
    const int nx,
    const int ny,
    const int px,
    const int py )
```

6.41.1.6 CRegularMesh3D() [6/6]

```
CRegularMesh3D::CRegularMesh3D (
    const double x1,
    const double y1,
    const double x2,
    const double y2,
    const int nx,
    const int ny )
```


6.41.1.7 ~CRegularMesh3D()

```
CRegularMesh3D::~~CRegularMesh3D ( )
```

6.41.2 Member Function Documentation

6.41.2.1 Clone()

```
CRegularMesh3D * corenc::Mesh::CRegularMesh3D::Clone ( ) const [inline]
```

6.41.2.2 FindElement()

```
const int CRegularMesh3D::FindElement (
    const Point & test ) const
```

6.41.2.3 GetBoundary() [1/2]

```
auto corenc::Mesh::CRegularMesh3D::GetBoundary ( ) -> decltype(m_edges) [inline]
```

6.41.2.4 GetBoundary() [2/2]

```
const CElement * CRegularMesh3D::GetBoundary (
    const unsigned int n ) const
```

6.41.2.5 GetElement()

```
const CElement * CRegularMesh3D::GetElement (
    const unsigned int n ) const
```

6.41.2.6 GetElements()

```
auto corenc::Mesh::CRegularMesh3D::GetElements ( ) -> decltype(m_elems) [inline]
```

6.41.2.7 getMinSize()

```
const double corenc::Mesh::CRegularMesh3D::getMinSize ( ) const [inline]
```

6.41.2.8 GetNode()

```
const Mesh::Point CRegularMesh3D::GetNode (
    const unsigned int n ) const
```

6.41.2.9 GetNumberOfBoundaries()

```
const unsigned int CRegularMesh3D::GetNumberOfBoundaries ( ) const
```

6.41.2.10 GetNumberOfElements()

```
const unsigned int CRegularMesh3D::GetNumberOfElements ( ) const
```

6.41.2.11 GetNumberOfINodes()

```
const int CRegularMesh3D::GetNumberOfINodes ( ) const
```

6.41.2.12 GetNumberOfNodes()

```
const unsigned int CRegularMesh3D::GetNumberOfNodes ( ) const
```

6.41.2.13 getParameter() [1/2]

```
const double CRegularMesh3D::getParameter (
    Parameters param,
    const unsigned int l,
    const int i ) const
```

6.41.2.14 `getParameter()` [2/2]

```
const double CRegularMesh3D::getParameter (
    Parameters param,
    const unsigned int l,
    const Point & p ) const
```

6.41.2.15 `getSolution()` [1/2]

```
const std::vector< double > CRegularMesh3D::getSolution ( ) const
```

6.41.2.16 `getSolution()` [2/2]

```
const double CRegularMesh3D::getSolution (
    const unsigned int element,
    const unsigned int node ) const
```

6.41.2.17 `interpolate()`

```
const int CRegularMesh3D::interpolate (
    const int node ) const
```

6.41.2.18 `operator=()`

```
CRegularMesh3D & corenc::Mesh::CRegularMesh3D::operator= (
    const CRegularMesh3D & tr ) [inline]
```

6.41.2.19 `refine_h()`

```
const int CRegularMesh3D::refine_h ( )
```

6.41.2.20 `refine_hp()`

```
const int corenc::Mesh::CRegularMesh3D::refine_hp ( )
```

6.41.2.21 refine_hx()

```
const int corenc::Mesh::CRegularMesh3D::refine_hx ( )
```

6.41.2.22 refine_hy()

```
const int corenc::Mesh::CRegularMesh3D::refine_hy ( )
```

6.41.2.23 refine_p()

```
const int CRegularMesh3D::refine_p ( )
```

6.41.2.24 setParameter() [1/2]

```
const int CRegularMesh3D::setParameter (
    const CParameter & p,
    const unsigned int type )
```

6.41.2.25 setParameter() [2/2]

```
const int CRegularMesh3D::setParameter (
    Parameters param,
    const double ,
    const unsigned int )
```

6.41.2.26 updateSolution() [1/4]

```
const int CRegularMesh3D::updateSolution (
    const std::vector< double > & )
```

6.41.2.27 updateSolution() [2/4]

```
const int CRegularMesh3D::updateSolution (
    const unsigned int element,
    const unsigned int node,
    const double value )
```

6.41.2.28 updateSolution() [3/4]

```
const int CRegularMesh3D::updateSolution (
    const unsigned int element,
    const unsigned int node,
    CSolution * value )
```

6.41.2.29 updateSolution() [4/4]

```
const int CRegularMesh3D::updateSolution (
    const unsigned int node,
    const double value )
```

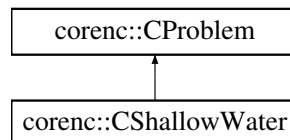
The documentation for this class was generated from the following files:

- CoreNCFEM/Grids/[RegularMesh3D.h](#)
- CoreNCFEM/Grids/[RegularMesh3D.cpp](#)

6.42 corenc::CShallowWater Class Reference

```
#include <ShallowWater.h>
```

Inheritance diagram for corenc::CShallowWater:

**Public Member Functions**

- [CShallowWater](#) ()
- [~CShallowWater](#) ()
- [Terms getTerm](#) (const unsigned int) const
- const unsigned int [getNumberOfTerms](#) () const
- const int [setTerm](#) (const unsigned int, const [Terms](#) &)
- const int [addTerm](#) (const [Terms](#) &)
- const int [removeTerm](#) (const [Terms](#) &)
- const int [load_parameters](#) (const std::string &file_name)
- const double [get_parameter](#) (const [Terms](#) &, const int element_type, const [Mesh::Point](#) &) const
- const double [get_parameter](#) (const [Terms](#) &, const int element_number, const int element_type, const [Mesh::Point](#) &) const
- const double [get_boundary_parameter](#) (const int type, const int element_type, const [Mesh::Point](#) &) const
- const double [get_boundary_parameter](#) (const int type, const int element_number, const int element_type, const [Mesh::Point](#) &) const
- const int [get_number_of_boundaries](#) () const

- const double [get_solution](#) (const int sys_number, const int element_type, const int element_number, const [Mesh::Point](#) &) const
- const int [get_boundary_type](#) (const int number) const
- const int [add_parameter](#) (const [Terms](#) &, const int element_type, const [Mesh::parameter](#)< double > &value)
- const int [set_parameter](#) (const [Terms](#) &, const int element_type, const [Mesh::parameter](#)< double > &value)
- const int [set_boundary_parameter](#) (const int type, const int element_type, const boundary &value)
- const int [add_boundary_parameter](#) (const int type, const int element_type, const [Mesh::parameter](#)< double > &value)
- const int [add_boundary_parameter](#) (const int element_type, const [Mesh::parameter](#)< double > &value, const [Mesh::parameter](#)< double > &value2)

6.42.1 Constructor & Destructor Documentation

6.42.1.1 CShallowWater()

```
CShallowWater::CShallowWater ( )
```

6.42.1.2 ~CShallowWater()

```
CShallowWater::~~CShallowWater ( )
```

6.42.2 Member Function Documentation

6.42.2.1 add_boundary_parameter() [1/2]

```
const int CShallowWater::add_boundary_parameter (
    const int element_type,
    const Mesh::parameter< double > & value,
    const Mesh::parameter< double > & value2 )
```

6.42.2.2 add_boundary_parameter() [2/2]

```
const int CShallowWater::add_boundary_parameter (
    const int type,
    const int element_type,
    const Mesh::parameter< double > & value )
```

6.42.2.3 add_parameter()

```
const int CShallowWater::add_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::parameter< double > & value )
```

6.42.2.4 addTerm()

```
const int CShallowWater::addTerm (
    const Terms & term ) [virtual]
```

Implements [corenc::CProblem](#).

6.42.2.5 get_boundary_parameter() [1/2]

```
const double CShallowWater::get_boundary_parameter (
    const int type,
    const int element_number,
    const int element_type,
    const Mesh::Point & p ) const
```

6.42.2.6 get_boundary_parameter() [2/2]

```
const double CShallowWater::get_boundary_parameter (
    const int type,
    const int element_type,
    const Mesh::Point & p ) const
```

6.42.2.7 get_boundary_type()

```
const int CShallowWater::get_boundary_type (
    const int number ) const
```

6.42.2.8 get_number_of_boundaries()

```
const int CShallowWater::get_number_of_boundaries ( ) const
```

6.42.2.9 `get_parameter()` [1/2]

```
const double CShallowWater::get_parameter (
    const Terms & term,
    const int element_number,
    const int element_type,
    const Mesh::Point & p ) const
```

6.42.2.10 `get_parameter()` [2/2]

```
const double CShallowWater::get_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::Point & p ) const
```

6.42.2.11 `get_solution()`

```
const double CShallowWater::get_solution (
    const int sys_number,
    const int element_type,
    const int element_number,
    const Mesh::Point & ) const
```

6.42.2.12 `getNumberOfTerms()`

```
const unsigned int CShallowWater::getNumberOfTerms ( ) const [virtual]
```

Implements [corenc::CProblem](#).

6.42.2.13 `getTerm()`

```
Terms CShallowWater::getTerm (
    const unsigned int i ) const [virtual]
```

Implements [corenc::CProblem](#).

6.42.2.14 load_parameters()

```
const int CShallowWater::load_parameters (
    const std::string & file_name ) [virtual]
```

Implements [corenc::CProblem](#).

6.42.2.15 removeTerm()

```
const int CShallowWater::removeTerm (
    const Terms & term )
```

6.42.2.16 set_boundary_parameter()

```
const int CShallowWater::set_boundary_parameter (
    const int type,
    const int element_type,
    const boundary & value )
```

6.42.2.17 set_parameter()

```
const int CShallowWater::set_parameter (
    const Terms & term,
    const int element_type,
    const Mesh::parameter< double > & value )
```

6.42.2.18 setTerm()

```
const int CShallowWater::setTerm (
    const unsigned int i,
    const Terms & term ) [virtual]
```

Implements [corenc::CProblem](#).

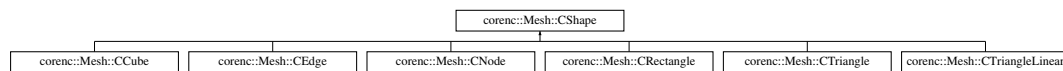
The documentation for this class was generated from the following files:

- Problems/[ShallowWater.h](#)
- Problems/[ShallowWater.cpp](#)

6.43 corenc::Mesh::CShape Class Reference

```
#include <Shape.h>
```

Inheritance diagram for corenc::Mesh::CShape:



Public Member Functions

- [CShape](#) ()
- [CShape](#) (const int *)
- virtual [~CShape](#) ()
- virtual const int [GetNumberOfNodes](#) () const
- virtual const int [GetNumberOfEdges](#) () const
- virtual const int [GetNumberOfFacets](#) () const
- virtual const int [GetNode](#) (const int) const
- virtual const int [GetNode](#) (const [NODES](#) &) const
- virtual const int [GetEdge](#) (const int) const
- virtual const int [GetFacet](#) (const int) const
- virtual const double [Integrate](#) (const [scalar_func](#) &, const std::vector< [Point](#) > &) const =0
- virtual const [Point](#) [Integrate](#) (const [vector_func](#) &, const std::vector< [Point](#) > &) const =0
- virtual const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const =0
- virtual void [SetNode](#) (const int, const int)=0
- virtual void [SetEdge](#) (const int, const int)
- virtual void [SetFacet](#) (const int, const int)

6.43.1 Constructor & Destructor Documentation

6.43.1.1 CShape() [1/2]

```
corenc::Mesh::CShape::CShape ( ) [inline]
```

6.43.1.2 CShape() [2/2]

```
corenc::Mesh::CShape::CShape (
    const int * ) [inline]
```

6.43.1.3 ~CShape()

```
virtual corenc::Mesh::CShape::~CShape ( ) [inline], [virtual]
```

6.43.2 Member Function Documentation

6.43.2.1 GetEdge()

```
virtual const int corenc::Mesh::CShape::GetEdge (
    const int ) const [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.2 GetFacet()

```
virtual const int corenc::Mesh::CShape::GetFacet (
    const int ) const [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.3 GetNode() [1/2]

```
virtual const int corenc::Mesh::CShape::GetNode (
    const int ) const [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CEdge](#), [corenc::Mesh::CNode](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.4 GetNode() [2/2]

```
virtual const int corenc::Mesh::CShape::GetNode (
    const NODES & ) const [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CEdge](#), [corenc::Mesh::CNode](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.5 GetNumberOfEdges()

```
virtual const int corenc::Mesh::CShape::GetNumberOfEdges ( ) const [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.6 GetNumberOfFacets()

```
virtual const int corenc::Mesh::CShape::GetNumberOfFacets ( ) const [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.7 GetNumberOfNodes()

```
virtual const int corenc::Mesh::CShape::GetNumberOfNodes ( ) const [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CEdge](#), [corenc::Mesh::CNode](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.8 Integrate() [1/3]

```
virtual const double corenc::Mesh::CShape::Integrate (
    const scalar_func & ,
    const std::vector< Point > & ) const [pure virtual]
```

6.43.2.9 Integrate() [2/3]

```
virtual const std::vector< double > corenc::Mesh::CShape::Integrate (
    const std::function< const std::vector< double >(const Point &)> & ,
    const std::vector< Point > & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CEdge](#), [corenc::Mesh::CNode](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.10 Integrate() [3/3]

```
virtual const Point corenc::Mesh::CShape::Integrate (
    const vector_func & ,
    const std::vector< Point > & ) const [pure virtual]
```

6.43.2.11 SetEdge()

```
virtual void corenc::Mesh::CShape::SetEdge (
    const int ,
    const int ) [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.12 SetFacet()

```
virtual void corenc::Mesh::CShape::SetFacet (
    const int ,
    const int ) [inline], [virtual]
```

Reimplemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

6.43.2.13 SetNode()

```
virtual void corenc::Mesh::CShape::SetNode (
    const int ,
    const int ) [pure virtual]
```

Implemented in [corenc::Mesh::CCube](#), [corenc::Mesh::CEdge](#), [corenc::Mesh::CNode](#), [corenc::Mesh::CRectangle](#), [corenc::Mesh::CTriangle](#), and [corenc::Mesh::CTriangleLinear](#).

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

- [CoreNCFEM/FiniteElements/Shape.h](#)

6.44 corenc::Mesh::CShapeFunction< Type > Class Template Reference

```
#include <ShapeFunction.h>
```

Public Member Functions

- [CShapeFunction](#) ()
- [CShapeFunction](#) (const [Point](#) *)
- virtual [~CShapeFunction](#) ()
- virtual const int [GetNumberOfShapeFunctions](#) () const =0
- virtual const double [GetShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const =0
- virtual const [Point](#) [GetNormal](#) () const =0
- virtual void [ReverseNormal](#) ()=0
- virtual const double [GetMeasure](#) () const =0

6.44.1 Constructor & Destructor Documentation

6.44.1.1 CShapeFunction() [1/2]

```
template<class Type >
corenc::Mesh::CShapeFunction< Type >::CShapeFunction ( ) [inline]
```

6.44.1.2 CShapeFunction() [2/2]

```
template<class Type >
corenc::Mesh::CShapeFunction< Type >::CShapeFunction (
    const Point * ) [inline]
```

6.44.1.3 ~CShapeFunction()

```
template<class Type >
virtual corenc::Mesh::CShapeFunction< Type >::~~CShapeFunction ( ) [inline], [virtual]
```

6.44.2 Member Function Documentation

6.44.2.1 GetGradShapeFunction()

```
template<class Type >
virtual const Point corenc::Mesh::CShapeFunction< Type >::GetGradShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CCubeBasis](#), [corenc::Mesh::CEdgeLinearBasis](#), [corenc::Mesh::CEdgeConstantBasis](#), [corenc::Mesh::CEdgeMultiBasis](#), [corenc::Mesh::CEdgeHermiteBasis](#), [corenc::Mesh::CEdge2ndBasis](#), [corenc::Mesh::CNodeBasis](#), [corenc::Mesh::CRectangleBasis](#), [corenc::Mesh::CRectangleHBasis](#), [corenc::Mesh::CRectangleBasis2x](#), [corenc::Mesh::CRectangleBasis2](#), [corenc::Mesh::CRectangleConstantBasis](#), [corenc::Mesh::CTriangleBasis](#), [corenc::Mesh::CTriangleLagrangeBasis](#), [corenc::Mesh::CTriangleLinearBasis](#), and [corenc::Mesh::CTriangleBasis](#).

6.44.2.2 GetMeasure()

```
template<class Type >
virtual const double corenc::Mesh::CShapeFunction< Type >::GetMeasure ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CCubeBasis](#), [corenc::Mesh::CEdgeLinearBasis](#), [corenc::Mesh::CEdgeConstantBasis](#), [corenc::Mesh::CEdgeMultiBasis](#), [corenc::Mesh::CEdgeHermiteBasis](#), [corenc::Mesh::CEdge2ndBasis](#), [corenc::Mesh::CNodeBasis](#), [corenc::Mesh::CRectangleBasis](#), [corenc::Mesh::CRectangleHBasis](#), [corenc::Mesh::CRectangleBasis2x](#), [corenc::Mesh::CRectangleBasis2](#), [corenc::Mesh::CRectangleConstantBasis](#), [corenc::Mesh::CTriangleBasis](#), [corenc::Mesh::CTriangleLagrangeBasis](#), and [corenc::Mesh::CTriangleLinearBasis](#).

6.44.2.3 GetNormal()

```
template<class Type >
virtual const Point corenc::Mesh::CShapeFunction< Type >::GetNormal ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CCubeBasis](#), [corenc::Mesh::CEdgeLinearBasis](#), [corenc::Mesh::CEdgeConstantBasis](#), [corenc::Mesh::CEdgeMultiBasis](#), [corenc::Mesh::CEdgeHermiteBasis](#), [corenc::Mesh::CEdge2ndBasis](#), [corenc::Mesh::CNodeBasis](#), [corenc::Mesh::CRectangleBasis](#), [corenc::Mesh::CRectangleHBasis](#), [corenc::Mesh::CRectangleBasis2x](#), [corenc::Mesh::CRectangleBasis2](#), [corenc::Mesh::CRectangleConstantBasis](#), [corenc::Mesh::CTriangleBasis](#), [corenc::Mesh::CTriangleLagrangeBasis](#), [corenc::Mesh::CTriangleLinearBasis](#), and [corenc::Mesh::CTriangleBasis](#).

6.44.2.4 GetNumberOfShapeFunctions()

```
template<class Type >
virtual const int corenc::Mesh::CShapeFunction< Type >::GetNumberOfShapeFunctions ( ) const [pure virtual]
```

Implemented in [corenc::Mesh::CCubeBasis](#), [corenc::Mesh::CEdgeLinearBasis](#), [corenc::Mesh::CEdgeConstantBasis](#), [corenc::Mesh::CEdgeMultiBasis](#), [corenc::Mesh::CEdgeHermiteBasis](#), [corenc::Mesh::CEdge2ndBasis](#), [corenc::Mesh::CNodeBasis](#), [corenc::Mesh::CRectangleBasis](#), [corenc::Mesh::CRectangleHBasis](#), [corenc::Mesh::CRectangleBasis2x](#), [corenc::Mesh::CRectangleBasis2](#), [corenc::Mesh::CRectangleConstantBasis](#), [corenc::Mesh::CTriangleBasis](#), [corenc::Mesh::CTriangleLagrangeBasis](#), [corenc::Mesh::CTriangleLinearBasis](#), and [corenc::Mesh::CTriangleBasis](#).

6.44.2.5 GetShapeFunction()

```
template<class Type >
virtual const double corenc::Mesh::CShapeFunction< Type >::GetShapeFunction (
    const int ,
    const Point & ) const [pure virtual]
```

Implemented in [corenc::Mesh::CCubeBasis](#), [corenc::Mesh::CEdgeLinearBasis](#), [corenc::Mesh::CEdgeConstantBasis](#), [corenc::Mesh::CEdgeMultiBasis](#), [corenc::Mesh::CEdgeHermiteBasis](#), [corenc::Mesh::CEdge2ndBasis](#), [corenc::Mesh::CNodeBasis](#), [corenc::Mesh::CRectangleBasis](#), [corenc::Mesh::CRectangleHBasis](#), [corenc::Mesh::CRectangleBasis2x](#), [corenc::Mesh::CRectangleBasis2](#), [corenc::Mesh::CRectangleConstantBasis](#), [corenc::Mesh::CTriangleBasis](#), [corenc::Mesh::CTriangleLagrangeBasis](#), [corenc::Mesh::CTriangleLinearBasis](#), and [corenc::Mesh::CTriangleBasis](#).

6.44.2.6 ReverseNormal()

```
template<class Type >
virtual void corenc::Mesh::CShapeFunction< Type >::ReverseNormal ( ) [pure virtual]
```

Implemented in [corenc::Mesh::CCubeBasis](#), [corenc::Mesh::CEdgeLinearBasis](#), [corenc::Mesh::CEdgeConstantBasis](#), [corenc::Mesh::CEdgeMultiBasis](#), [corenc::Mesh::CEdgeHermiteBasis](#), [corenc::Mesh::CEdge2ndBasis](#), [corenc::Mesh::CNodeBasis](#), [corenc::Mesh::CRectangleBasis](#), [corenc::Mesh::CRectangleHBasis](#), [corenc::Mesh::CRectangleBasis2x](#), [corenc::Mesh::CRectangleBasis2](#), [corenc::Mesh::CRectangleConstantBasis](#), [corenc::Mesh::CTriangleBasis](#), [corenc::Mesh::CTriangleLagrangeBasis](#), [corenc::Mesh::CTriangleLinearBasis](#), and [corenc::Mesh::CTriangleBasis](#).

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

- [CoreNCFEM/FiniteElements/ShapeFunction.h](#)

6.45 Methods::CSMethod Class Reference

```
#include <CSMethod.h>
```

Public Member Functions

- [CSMethod \(\)](#)
- virtual [~CSMethod \(\)](#)

6.45.1 Constructor & Destructor Documentation

6.45.1.1 CSMethod()

```
Methods::CSMethod::CSMethod ( ) [inline]
```

6.45.1.2 ~CSMethod()

```
virtual Methods::CSMethod::~~CSMethod ( ) [inline], [virtual]
```

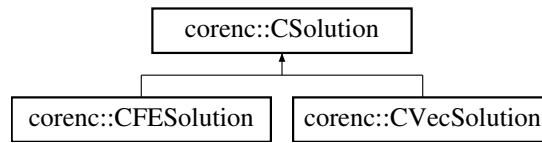
The documentation for this class was generated from the following file:

- [CoreNCFEM/Methods/CSMethod.h](#)

6.46 corenc::CSolution Class Reference

```
#include <FESolution.h>
```

Inheritance diagram for corenc::CSolution:



Public Member Functions

- [CSolution\(\)](#)
- virtual [~CSolution\(\)](#)

6.46.1 Constructor & Destructor Documentation

6.46.1.1 CSolution()

```
corenc::CSolution::CSolution ( ) [inline]
```

6.46.1.2 ~CSolution()

```
virtual corenc::CSolution::~~CSolution ( ) [inline], [virtual]
```

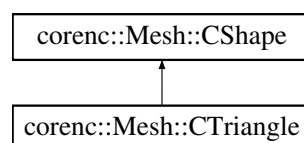
The documentation for this class was generated from the following file:

- CoreNCFEM/[FESolution.h](#)

6.47 corenc::Mesh::CTriangle Class Reference

```
#include <Triangle.h>
```

Inheritance diagram for corenc::Mesh::CTriangle:



Public Member Functions

- [CTriangle](#) ()
- [CTriangle](#) (const int n1, const int n2, const int n3, const int order)
- [CTriangle](#) (const int n1, const int n2, const int n3, const int e1, const int e2, const int e3, const int order)
- [CTriangle](#) (const int *, const int order)
- [CTriangle](#) (const int *, const int *, const int order)
- [CTriangle](#) (const [CTriangle](#) &)
- [CTriangle](#) & [operator=](#) (const [CTriangle](#) &t)
- const bool [operator==](#) (const [CTriangle](#) &t)
- std::istream & [operator>>](#) (std::istream &is)
- [~CTriangle](#) ()
- const int [GetNode](#) (const int) const
- const int [GetNode](#) (const [NODES](#) &) const
- const int [GetEdge](#) (const int) const
- const int [GetFacet](#) (const int) const
- const int [GetNumberOfNodes](#) () const
- const int [GetNumberOfEdges](#) () const
- const int [GetNumberOfFacets](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- void [SetNode](#) (const int k, const int node)
- const int [IncreaseOrder](#) ()
- void [SetEdge](#) (const int k, const int edge)
- void [SetFacet](#) (const int k, const int facet)

6.47.1 Constructor & Destructor Documentation

6.47.1.1 [CTriangle](#)() [1/6]

```
CTriangle::CTriangle ( )
```

6.47.1.2 [CTriangle](#)() [2/6]

```
CTriangle::CTriangle (
    const int n1,
    const int n2,
    const int n3,
    const int order )
```

6.47.1.3 CTriangle() [3/6]

```
CTriangle::CTriangle (
    const int n1,
    const int n2,
    const int n3,
    const int e1,
    const int e2,
    const int e3,
    const int order )
```

6.47.1.4 CTriangle() [4/6]

```
CTriangle::CTriangle (
    const int * nodes,
    const int order )
```

6.47.1.5 CTriangle() [5/6]

```
CTriangle::CTriangle (
    const int * nodes,
    const int * edges,
    const int order )
```

6.47.1.6 CTriangle() [6/6]

```
CTriangle::CTriangle (
    const CTriangle & t )
```

6.47.1.7 ~CTriangle()

```
corenc::Mesh::CTriangle::~~CTriangle ( ) [inline]
```

6.47.2 Member Function Documentation

6.47.2.1 GetEdge()

```
const int CTriangle::GetEdge (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.2 GetFacet()

```
const int CTriangle::GetFacet (
    const int ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.3 GetNode() [1/2]

```
const int CTriangle::GetNode (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.4 GetNode() [2/2]

```
const int CTriangle::GetNode (
    const NODES & node ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.5 GetNumberOfEdges()

```
const int CTriangle::GetNumberOfEdges ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.6 GetNumberOfFacets()

```
const int CTriangle::GetNumberOfFacets ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.7 GetNumberOfNodes()

```
const int CTriangle::GetNumberOfNodes ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.8 IncreaseOrder()

```
const int CTriangle::IncreaseOrder ( )
```

6.47.2.9 Integrate() [1/3]

```
const double CTriangle::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.47.2.10 Integrate() [2/3]

```
const Point CTriangle::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.47.2.11 Integrate() [3/3]

```
const vector< double > CTriangle::Integrate (
    const std::function< const std::vector< double >(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.47.2.12 operator=()

```
CTriangle & corenc::Mesh::CTriangle::operator= (
    const CTriangle & t ) [inline]
```

6.47.2.13 operator==()

```
const bool corenc::Mesh::CTriangle::operator== (
    const CTriangle & t ) [inline]
```

6.47.2.14 operator>>()

```
std::istream & corenc::Mesh::CTriangle::operator>> (
    std::istream & is ) [inline]
```

6.47.2.15 SetEdge()

```
void CTriangle::SetEdge (
    const int k,
    const int edge ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.16 SetFacet()

```
void CTriangle::SetFacet (
    const int k,
    const int facet ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.47.2.17 SetNode()

```
void CTriangle::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CShape](#).

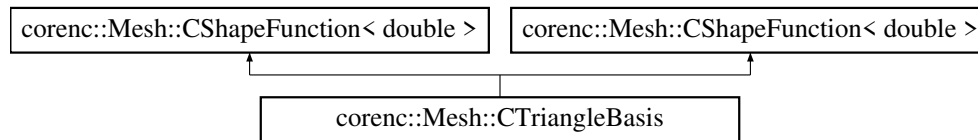
The documentation for this class was generated from the following files:

- [CoreNCFEM/FiniteElements/Triangle.h](#)
- [CoreNCFEM/FiniteElements/Triangle.cpp](#)

6.48 corenc::Mesh::CTriangleBasis Class Reference

```
#include <Triangle.h>
```

Inheritance diagram for corenc::Mesh::CTriangleBasis:



Public Member Functions

- [CTriangleBasis](#) ()
- [CTriangleBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CTriangleBasis](#) (const [Point](#) *, const int order)
- [CTriangleBasis](#) (const [CTriangleBasis](#) &)
- [CTriangleBasis](#) & [operator=](#) (const [CTriangleBasis](#) &t)
- [~CTriangleBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const
- [CTriangleBasis](#) ()
- [CTriangleBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CTriangleBasis](#) (const [Point](#) *, const int order)
- [CTriangleBasis](#) (const [CTriangleBasis](#) &)
- [CTriangleBasis](#) & [operator=](#) (const [CTriangleBasis](#) &t)
- [~CTriangleBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const

6.48.1 Constructor & Destructor Documentation

6.48.1.1 CTriangleBasis() [1/8]

```
CTriangleBasis::CTriangleBasis ( )
```

6.48.1.2 CTriangleBasis() [2/8]

```
CTriangleBasis::CTriangleBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const int order )
```

6.48.1.3 CTriangleBasis() [3/8]

```
CTriangleBasis::CTriangleBasis (
    const Point * p,
    const int order )
```

6.48.1.4 CTriangleBasis() [4/8]

```
CTriangleBasis::CTriangleBasis (
    const CTriangleBasis & t )
```

6.48.1.5 ~CTriangleBasis() [1/2]

```
corenc::Mesh::CTriangleBasis::~~CTriangleBasis ( ) [inline]
```

6.48.1.6 CTriangleBasis() [5/8]

```
corenc::Mesh::CTriangleBasis::CTriangleBasis ( )
```

6.48.1.7 CTriangleBasis() [6/8]

```
corenc::Mesh::CTriangleBasis::CTriangleBasis (
    const Point & ,
    const Point & ,
    const Point & ,
    const int order )
```


6.48.1.8 CTriangleBasis() [7/8]

```
corenc::Mesh::CTriangleBasis::CTriangleBasis (
    const Point * ,
    const int order )
```

6.48.1.9 CTriangleBasis() [8/8]

```
corenc::Mesh::CTriangleBasis::CTriangleBasis (
    const CTriangleBasis & )
```

6.48.1.10 ~CTriangleBasis() [2/2]

```
corenc::Mesh::CTriangleBasis::~~CTriangleBasis ( ) [inline]
```

6.48.2 Member Function Documentation**6.48.2.1 GetGradShapeFunction() [1/2]**

```
const Point CTriangleBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.2 GetGradShapeFunction() [2/2]

```
const Point corenc::Mesh::CTriangleBasis::GetGradShapeFunction (
    const int ,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.3 GetMeasure()

```
const double corenc::Mesh::CTriangleBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.4 GetNormal() [1/2]

```
const Point CTriangleBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.5 GetNormal() [2/2]

```
const Point corenc::Mesh::CTriangleBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.6 GetNumberOfShapeFunctions() [1/2]

```
const int CTriangleBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.7 GetNumberOfShapeFunctions() [2/2]

```
const int corenc::Mesh::CTriangleBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.8 GetShapeFunction() [1/2]

```
const double CTriangleBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.9 GetShapeFunction() [2/2]

```
const double corenc::Mesh::CTriangleBasis::GetShapeFunction (
    const int ,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.10 GetValue() [1/2]

```
const double CTriangleBasis::GetValue (
    const Point & p ) const
```

6.48.2.11 GetValue() [2/2]

```
const double corenc::Mesh::CTriangleBasis::GetValue (
    const Point & ) const
```

6.48.2.12 GetWeight()

```
const double CTriangleBasis::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.48.2.13 IncreaseOrder()

```
const int CTriangleBasis::IncreaseOrder ( )
```

6.48.2.14 operator=() [1/2]

```
CTriangleBasis & corenc::Mesh::CTriangleBasis::operator= (
    const CTriangleBasis & t ) [inline]
```

6.48.2.15 operator=() [2/2]

```
CTriangleBasis & corenc::Mesh::CTriangleBasis::operator= (
    const CTriangleBasis & t ) [inline]
```

6.48.2.16 ReverseNormal() [1/2]

```
void CTriangleBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.48.2.17 ReverseNormal() [2/2]

```
void corenc::Mesh::CTriangleBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

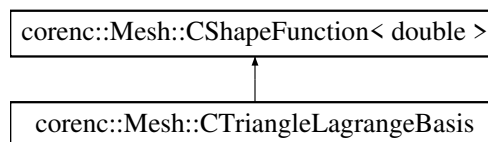
The documentation for this class was generated from the following files:

- CoreNCFEM/FiniteElements/[Triangle.h](#)
- CoreNCFEM/FiniteElements/[TriangleLinear.h](#)
- CoreNCFEM/FiniteElements/[Triangle.cpp](#)

6.49 corenc::Mesh::CTriangleLagrangeBasis Class Reference

```
#include <Triangle.h>
```

Inheritance diagram for corenc::Mesh::CTriangleLagrangeBasis:



Public Member Functions

- [CTriangleLagrangeBasis](#) ()
- [CTriangleLagrangeBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &, const int order)
- [CTriangleLagrangeBasis](#) (const [Point](#) *, const int order)
- [CTriangleLagrangeBasis](#) (const [CTriangleLagrangeBasis](#) &)
- [CTriangleLagrangeBasis](#) & operator= (const [CTriangleLagrangeBasis](#) &t)
- [~CTriangleLagrangeBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetAlpha](#) (const int i, const int j) const
- const double [GetMeasure](#) () const
- const double [GetWeight](#) (const int, const std::vector< [Point](#) > &verts, const std::function< const double(const [Point](#) &)> &f) const

6.49.1 Constructor & Destructor Documentation

6.49.1.1 CTriangleLagrangeBasis() [1/4]

```
CTriangleLagrangeBasis::CTriangleLagrangeBasis ( )
```

6.49.1.2 CTriangleLagrangeBasis() [2/4]

```
CTriangleLagrangeBasis::CTriangleLagrangeBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3,
    const int order )
```

6.49.1.3 CTriangleLagrangeBasis() [3/4]

```
CTriangleLagrangeBasis::CTriangleLagrangeBasis (
    const Point * p,
    const int order )
```

6.49.1.4 CTriangleLagrangeBasis() [4/4]

```
CTriangleLagrangeBasis::CTriangleLagrangeBasis (
    const CTriangleLagrangeBasis & t )
```

6.49.1.5 ~CTriangleLagrangeBasis()

```
corenc::Mesh::CTriangleLagrangeBasis::~~CTriangleLagrangeBasis ( ) [inline]
```

6.49.2 Member Function Documentation

6.49.2.1 GetAlpha()

```
const double corenc::Mesh::CTriangleLagrangeBasis::GetAlpha (
    const int i,
    const int j ) const [inline]
```

6.49.2.2 GetGradShapeFunction()

```
const Point CTriangleLagrangeBasis::GetGradShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.49.2.3 GetMeasure()

```
const double corenc::Mesh::CTriangleLagrangeBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.49.2.4 GetNormal()

```
const Point CTriangleLagrangeBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.49.2.5 GetNumberOfShapeFunctions()

```
const int CTriangleLagrangeBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.49.2.6 GetShapeFunction()

```
const double CTriangleLagrangeBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.49.2.7 GetValue()

```
const double CTriangleLagrangeBasis::GetValue (
    const Point & p ) const
```

6.49.2.8 GetWeight()

```
const double CTriangleLagrangeBasis::GetWeight (
    const int ,
    const std::vector< Point > & verts,
    const std::function< const double(const Point &)> & f ) const
```

6.49.2.9 IncreaseOrder()

```
const int CTriangleLagrangeBasis::IncreaseOrder ( )
```

6.49.2.10 operator=()

```
CTriangleLagrangeBasis & corenc::Mesh::CTriangleLagrangeBasis::operator= (
    const CTriangleLagrangeBasis & t ) [inline]
```

6.49.2.11 ReverseNormal()

```
void CTriangleLagrangeBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

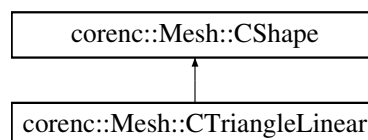
The documentation for this class was generated from the following files:

- CoreNCFEM/FiniteElements/[Triangle.h](#)
- CoreNCFEM/FiniteElements/[TriangleLagrange.cpp](#)

6.50 corenc::Mesh::CTriangleLinear Class Reference

```
#include <TriangleLinear.h>
```

Inheritance diagram for corenc::Mesh::CTriangleLinear:



Public Member Functions

- [CTriangleLinear](#) ()
- [CTriangleLinear](#) (const int n1, const int n2, const int n3)
- [CTriangleLinear](#) (const int n1, const int n2, const int n3, const int e1, const int e2, const int e3)
- [CTriangleLinear](#) (const int *)
- [CTriangleLinear](#) (const int *, const int *)
- [CTriangleLinear](#) (const [CTriangleLinear](#) &)
- [CTriangleLinear](#) & [operator=](#) (const [CTriangleLinear](#) &t)
- const bool [operator==](#) (const [CTriangleLinear](#) &t)
- std::istream & [operator>>](#) (std::istream &is)
- [~CTriangleLinear](#) ()
- const int [GetNode](#) (const int) const
- const int [GetNode](#) (const [NODES](#) &) const
- const int [GetEdge](#) (const int) const
- const int [GetFacet](#) (const int) const
- const int [GetNumberOfNodes](#) () const
- const int [GetNumberOfEdges](#) () const
- const int [GetNumberOfFacets](#) () const
- const double [Integrate](#) (const std::function< const double(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const [Point](#) [Integrate](#) (const std::function< const [Point](#)(const [Point](#) &)> &, const std::vector< [Point](#) > &v) const
- const std::vector< double > [Integrate](#) (const std::function< const std::vector< double >(const [Point](#) &)> &, const std::vector< [Point](#) > &) const
- void [SetNode](#) (const int k, const int node)
- const int [IncreaseOrder](#) ()
- void [SetEdge](#) (const int k, const int edge)
- void [SetFacet](#) (const int k, const int facet)

6.50.1 Constructor & Destructor Documentation

6.50.1.1 [CTriangleLinear](#)() [1/6]

```
CTriangleLinear::CTriangleLinear ( )
```

6.50.1.2 [CTriangleLinear](#)() [2/6]

```
CTriangleLinear::CTriangleLinear (
    const int n1,
    const int n2,
    const int n3 )
```


6.50.1.3 CTriangleLinear() [3/6]

```
CTriangleLinear::CTriangleLinear (
    const int n1,
    const int n2,
    const int n3,
    const int e1,
    const int e2,
    const int e3 )
```

6.50.1.4 CTriangleLinear() [4/6]

```
CTriangleLinear::CTriangleLinear (
    const int * nodes )
```

6.50.1.5 CTriangleLinear() [5/6]

```
CTriangleLinear::CTriangleLinear (
    const int * nodes,
    const int * edges )
```

6.50.1.6 CTriangleLinear() [6/6]

```
CTriangleLinear::CTriangleLinear (
    const CTriangleLinear & t )
```

6.50.1.7 ~CTriangleLinear()

```
corenc::Mesh::CTriangleLinear::~~CTriangleLinear ( ) [inline]
```

6.50.2 Member Function Documentation

6.50.2.1 GetEdge()

```
const int CTriangleLinear::GetEdge (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.2 GetFacet()

```
const int CTriangleLinear::GetFacet (
    const int ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.3 GetNode() [1/2]

```
const int CTriangleLinear::GetNode (
    const int n ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.4 GetNode() [2/2]

```
const int CTriangleLinear::GetNode (
    const NODES & node ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.5 GetNumberOfEdges()

```
const int CTriangleLinear::GetNumberOfEdges ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.6 GetNumberOfFacets()

```
const int CTriangleLinear::GetNumberOfFacets ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.7 GetNumberOfNodes()

```
const int CTriangleLinear::GetNumberOfNodes ( ) const [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.8 IncreaseOrder()

```
const int corenc::Mesh::CTriangleLinear::IncreaseOrder ( ) [inline]
```

6.50.2.9 Integrate() [1/3]

```
const double CTriangleLinear::Integrate (
    const std::function< const double(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.50.2.10 Integrate() [2/3]

```
const Point CTriangleLinear::Integrate (
    const std::function< const Point(const Point &)> & f,
    const std::vector< Point > & v ) const
```

6.50.2.11 Integrate() [3/3]

```
const vector< double > CTriangleLinear::Integrate (
    const std::function< const std::vector< double >(const Point &)> & f,
    const std::vector< Point > & v ) const [virtual]
```

Implements [corenc::Mesh::CShape](#).

6.50.2.12 operator=()

```
CTriangleLinear & corenc::Mesh::CTriangleLinear::operator= (
    const CTriangleLinear & t ) [inline]
```

6.50.2.13 operator==()

```
const bool corenc::Mesh::CTriangleLinear::operator== (
    const CTriangleLinear & t ) [inline]
```

6.50.2.14 operator>>()

```
std::istream & corenc::Mesh::CTriangleLinear::operator>> (
    std::istream & is ) [inline]
```

6.50.2.15 SetEdge()

```
void CTriangleLinear::SetEdge (
    const int k,
    const int edge ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.16 SetFacet()

```
void CTriangleLinear::SetFacet (
    const int k,
    const int facet ) [virtual]
```

Reimplemented from [corenc::Mesh::CShape](#).

6.50.2.17 SetNode()

```
void CTriangleLinear::SetNode (
    const int k,
    const int node ) [virtual]
```

Implements [corenc::Mesh::CShape](#).

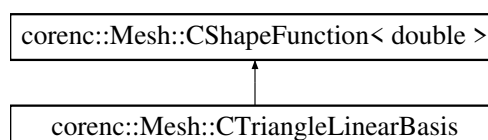
The documentation for this class was generated from the following files:

- CoreNCFEM/FiniteElements/[TriangleLinear.h](#)
- CoreNCFEM/FiniteElements/[TriangleLinear.cpp](#)

6.51 corenc::Mesh::CTriangleLinearBasis Class Reference

```
#include <TriangleLinear.h>
```

Inheritance diagram for corenc::Mesh::CTriangleLinearBasis:



Public Member Functions

- [CTriangleLinearBasis](#) ()
- [CTriangleLinearBasis](#) (const [Point](#) &, const [Point](#) &, const [Point](#) &)
- [CTriangleLinearBasis](#) (const [Point](#) *)
- [CTriangleLinearBasis](#) (const [CTriangleLinearBasis](#) &)
- [CTriangleLinearBasis](#) & [operator=](#) (const [CTriangleLinearBasis](#) &t)
- [~CTriangleLinearBasis](#) ()
- const int [GetNumberOfShapeFunctions](#) () const
- const double [GetShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetGradShapeFunction](#) (const int, const [Point](#) &) const
- const [Point](#) [GetNormal](#) () const
- void [ReverseNormal](#) ()
- const double [GetValue](#) (const [Point](#) &) const
- const int [IncreaseOrder](#) ()
- const double [GetMeasure](#) () const

6.51.1 Constructor & Destructor Documentation

6.51.1.1 CTriangleLinearBasis() [1/4]

```
CTriangleLinearBasis::CTriangleLinearBasis ( )
```

6.51.1.2 CTriangleLinearBasis() [2/4]

```
CTriangleLinearBasis::CTriangleLinearBasis (
    const Point & p1,
    const Point & p2,
    const Point & p3 )
```

6.51.1.3 CTriangleLinearBasis() [3/4]

```
CTriangleLinearBasis::CTriangleLinearBasis (
    const Point * p )
```

6.51.1.4 CTriangleLinearBasis() [4/4]

```
CTriangleLinearBasis::CTriangleLinearBasis (
    const CTriangleLinearBasis & t )
```

6.51.1.5 ~CTriangleLinearBasis()

```
corenc::Mesh::CTriangleLinearBasis::~~CTriangleLinearBasis ( ) [inline]
```

6.51.2 Member Function Documentation

6.51.2.1 GetGradShapeFunction()

```
const Point CTriangleLinearBasis::GetGradShapeFunction (
    const int k,
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.51.2.2 GetMeasure()

```
const double corenc::Mesh::CTriangleLinearBasis::GetMeasure ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.51.2.3 GetNormal()

```
const Point CTriangleLinearBasis::GetNormal ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.51.2.4 GetNumberOfShapeFunctions()

```
const int CTriangleLinearBasis::GetNumberOfShapeFunctions ( ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.51.2.5 GetShapeFunction()

```
const double CTriangleLinearBasis::GetShapeFunction (
    const int k,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

6.51.2.6 GetValue()

```
const double CTriangleLinearBasis::GetValue (
    const Point & p ) const
```

6.51.2.7 IncreaseOrder()

```
const int corenc::Mesh::CTriangleLinearBasis::IncreaseOrder ( ) [inline]
```

6.51.2.8 operator=()

```
CTriangleLinearBasis & corenc::Mesh::CTriangleLinearBasis::operator= (
    const CTriangleLinearBasis & t ) [inline]
```

6.51.2.9 ReverseNormal()

```
void CTriangleLinearBasis::ReverseNormal ( ) [virtual]
```

Implements [corenc::Mesh::CShapeFunction< double >](#).

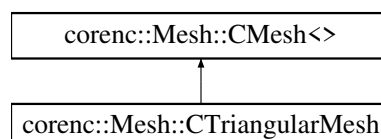
The documentation for this class was generated from the following files:

- CoreNCFEM/FiniteElements/[TriangleLinear.h](#)
- CoreNCFEM/FiniteElements/[TriangleLinear.cpp](#)

6.52 corenc::Mesh::CTriangularMesh Class Reference

```
#include <TriangularMesh.h>
```

Inheritance diagram for corenc::Mesh::CTriangularMesh:



Public Member Functions

- [CTriangularMesh](#) ()
- [CTriangularMesh](#) (const std::string &file_name)
- [CTriangularMesh](#) (const [CTriangularMesh](#) &)
- [CTriangularMesh](#) (const [Point](#) &p1, const [Point](#) &p2, const int nx, const int ny)
- [CTriangularMesh](#) & [operator=](#) (const [CTriangularMesh](#) &tr)
- [CTriangularMesh](#) * [Clone](#) () const
- const unsigned int [GetNumberOfElements](#) () const
- const unsigned int [GetNumberOfNodes](#) () const
- const unsigned int [GetNumberOfBoundaries](#) () const
- const int [FindElement](#) (const [Point](#) &) const
- const [Point](#) [GetNode](#) (const unsigned int) const
- const [CElement](#) * [GetElement](#) (const unsigned int) const
- const [CElement](#) * [GetBoundary](#) (const unsigned int) const
- const double [getMinSize](#) () const
- const double [getSolution](#) (const unsigned int element, const unsigned int node) const
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, const double value)
- const std::vector< double > [getSolution](#) () const
- const int [updateSolution](#) (const std::vector< double > &)
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, [CSolution](#) *value)
- const double [getParameter](#) ([Parameters](#), const unsigned int, const [Point](#) &) const
- const double [getParameter](#) ([Parameters](#), const unsigned int, const int) const
- const int [setParameter](#) ([Parameters](#), const double, const unsigned int)
- const int [setParameter](#) (const [CParameter](#) &, const unsigned int type)
- const int [updateSolution](#) (const unsigned int node, const double value)
- const int [refine_h](#) ()
- const int [refine_p](#) ()
- const int [refine_hp](#) ()
- const int [set4thOrder](#) ()
- const int [set2ndOrder](#) ()
- const int [set3rdOrder](#) ()
- const int [interpolate](#) (const int node) const
- const int [GetNumberOfNodes](#) () const
- [~CTriangularMesh](#) ()
- auto [GetElements](#) () -> decltype(m_elems)
- auto [GetBoundary](#) () -> decltype(m_edges)

6.52.1 Constructor & Destructor Documentation

6.52.1.1 CTriangularMesh() [1/4]

```
CTriangularMesh::CTriangularMesh ( )
```


6.52.1.2 CTriangularMesh() [2/4]

```
corenc::Mesh::CTriangularMesh::CTriangularMesh (
    const std::string & file_name )
```

6.52.1.3 CTriangularMesh() [3/4]

```
CTriangularMesh::CTriangularMesh (
    const CTriangularMesh & tr )
```

6.52.1.4 CTriangularMesh() [4/4]

```
CTriangularMesh::CTriangularMesh (
    const Point & p1,
    const Point & p2,
    const int nx,
    const int ny )
```

6.52.1.5 ~CTriangularMesh()

```
CTriangularMesh::~~CTriangularMesh ( )
```

6.52.2 Member Function Documentation

6.52.2.1 Clone()

```
CTriangularMesh * corenc::Mesh::CTriangularMesh::Clone ( ) const [inline]
```

6.52.2.2 FindElement()

```
const int CTriangularMesh::FindElement (
    const Point & test ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.3 GetBoundary() [1/2]

```
auto corenc::Mesh::CTriangularMesh::GetBoundary ( ) -> decltype(m_edges)    [inline]
```

6.52.2.4 GetBoundary() [2/2]

```
const CElement * CTriangularMesh::GetBoundary (
    const unsigned int n ) const    [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.5 GetElement()

```
const CElement * CTriangularMesh::GetElement (
    const unsigned int n ) const    [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.6 GetElements()

```
auto corenc::Mesh::CTriangularMesh::GetElements ( ) -> decltype(m_elems)    [inline]
```

6.52.2.7 getMinSize()

```
const double corenc::Mesh::CTriangularMesh::getMinSize ( ) const    [inline], [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.8 GetNode()

```
const Mesh::Point CTriangularMesh::GetNode (
    const unsigned int n ) const    [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.9 GetNumberOfBoundaries()

```
const unsigned int CTriangularMesh::GetNumberOfBoundaries ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.10 GetNumberOfElements()

```
const unsigned int CTriangularMesh::GetNumberOfElements ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.11 GetNumberOfINodes()

```
const int CTriangularMesh::GetNumberOfINodes ( ) const
```

6.52.2.12 GetNumberOfNodes()

```
const unsigned int CTriangularMesh::GetNumberOfNodes ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.13 getParameter() [1/2]

```
const double CTriangularMesh::getParameter (
    Parameters param,
    const unsigned int l,
    const int i ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.14 getParameter() [2/2]

```
const double CTriangularMesh::getParameter (
    Parameters param,
    const unsigned int l,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.15 `getSolution()` [1/2]

```
const std::vector< double > CTriangularMesh::getSolution ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.16 `getSolution()` [2/2]

```
const double CTriangularMesh::getSolution (
    const unsigned int element,
    const unsigned int node ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.17 `interpolate()`

```
const int CTriangularMesh::interpolate (
    const int node ) const
```

6.52.2.18 `operator=()`

```
CTriangularMesh & corenc::Mesh::CTriangularMesh::operator= (
    const CTriangularMesh & tr ) [inline]
```

6.52.2.19 `refine_h()`

```
const int CTriangularMesh::refine_h ( )
```

6.52.2.20 `refine_hp()`

```
const int corenc::Mesh::CTriangularMesh::refine_hp ( )
```

6.52.2.21 `refine_p()`

```
const int CTriangularMesh::refine_p ( )
```

6.52.2.22 set2ndOrder()

```
const int CTriangularMesh::set2ndOrder ( )
```

6.52.2.23 set3rdOrder()

```
const int CTriangularMesh::set3rdOrder ( )
```

6.52.2.24 set4thOrder()

```
const int CTriangularMesh::set4thOrder ( )
```

6.52.2.25 setParameter() [1/2]

```
const int CTriangularMesh::setParameter (
    const CParameter & p,
    const unsigned int type )
```

6.52.2.26 setParameter() [2/2]

```
const int CTriangularMesh::setParameter (
    Parameters param,
    const double ,
    const unsigned int ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.27 updateSolution() [1/4]

```
const int CTriangularMesh::updateSolution (
    const std::vector< double > & ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.28 updateSolution() [2/4]

```
const int CTriangularMesh::updateSolution (
    const unsigned int element,
    const unsigned int node,
    const double value ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.29 updateSolution() [3/4]

```
const int CTriangularMesh::updateSolution (
    const unsigned int element,
    const unsigned int node,
    CSolution * value ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.52.2.30 updateSolution() [4/4]

```
const int CTriangularMesh::updateSolution (
    const unsigned int node,
    const double value ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

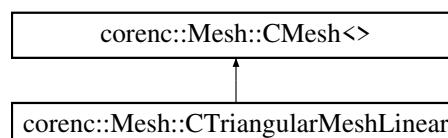
The documentation for this class was generated from the following files:

- CoreNCFEM/Grids/[TriangularMesh.h](#)
- CoreNCFEM/Grids/[TriangularMesh.cpp](#)

6.53 corenc::Mesh::CTriangularMeshLinear Class Reference

```
#include <TriangularMeshLinear.h>
```

Inheritance diagram for corenc::Mesh::CTriangularMeshLinear:



Public Member Functions

- [CTriangularMeshLinear](#) ()
- [CTriangularMeshLinear](#) (const std::string &file_name)
- [CTriangularMeshLinear](#) (const [CTriangularMeshLinear](#) &)
- const unsigned int [GetNumberOfElements](#) () const
- const unsigned int [GetNumberOfNodes](#) () const
- const unsigned int [GetNumberOfBoundaries](#) () const
- const int [FindElement](#) (const [Point](#) &) const
- const [Point](#) [GetNode](#) (const unsigned int) const
- const [CElement](#) * [GetElement](#) (const unsigned int) const
- const [CElement](#) * [GetBoundary](#) (const unsigned int) const
- const double [getMinSize](#) () const
- const double [getSolution](#) (const unsigned int element, const unsigned int node) const
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, const double value)
- const std::vector< double > [getSolution](#) () const
- const int [updateSolution](#) (const std::vector< double > &)
- const int [updateSolution](#) (const unsigned int element, const unsigned int node, [CSolution](#) *value)
- const double [getParameter](#) ([Parameters](#), const unsigned int, const [Point](#) &) const
- const double [getParameter](#) ([Parameters](#), const unsigned int, const int) const
- const int [setParameter](#) ([Parameters](#), const double, const unsigned int)
- const int [setParameter](#) (const [CParameter](#) &, const unsigned int type)
- const int [updateSolution](#) (const unsigned int node, const double value)
- const int [refine_h](#) ()
- [~CTriangularMeshLinear](#) ()
- auto [GetElements](#) () -> decltype(m_elems)
- auto [GetBoundary](#) () -> decltype(m_edges)

6.53.1 Constructor & Destructor Documentation

6.53.1.1 CTriangularMeshLinear() [1/3]

```
CTriangularMeshLinear::CTriangularMeshLinear ( )
```

6.53.1.2 CTriangularMeshLinear() [2/3]

```
corenc::Mesh::CTriangularMeshLinear::CTriangularMeshLinear (
    const std::string & file_name )
```

6.53.1.3 CTriangularMeshLinear() [3/3]

```
corenc::Mesh::CTriangularMeshLinear::CTriangularMeshLinear (
    const CTriangularMeshLinear & )
```

6.53.1.4 ~CTriangularMeshLinear()

```
CTriangularMeshLinear::~~CTriangularMeshLinear ( )
```

6.53.2 Member Function Documentation

6.53.2.1 FindElement()

```
const int CTriangularMeshLinear::FindElement (
    const Point & ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.2 GetBoundary() [1/2]

```
auto corenc::Mesh::CTriangularMeshLinear::GetBoundary ( ) -> decltype(m_edges) [inline]
```

6.53.2.3 GetBoundary() [2/2]

```
const CElement * CTriangularMeshLinear::GetBoundary (
    const unsigned int n ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.4 GetElement()

```
const CElement * CTriangularMeshLinear::GetElement (
    const unsigned int n ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.5 GetElements()

```
auto corenc::Mesh::CTriangularMeshLinear::GetElements ( ) -> decltype(m_elems) [inline]
```


6.53.2.6 getMinSize()

```
const double corenc::Mesh::CTriangularMeshLinear::getMinSize ( ) const [inline], [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.7 GetNode()

```
const Mesh::Point CTriangularMeshLinear::GetNode (
    const unsigned int n ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.8 GetNumberOfBoundaries()

```
const unsigned int CTriangularMeshLinear::GetNumberOfBoundaries ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.9 GetNumberOfElements()

```
const unsigned int CTriangularMeshLinear::GetNumberOfElements ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.10 GetNumberOfNodes()

```
const unsigned int CTriangularMeshLinear::GetNumberOfNodes ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.11 getParameter() [1/2]

```
const double CTriangularMeshLinear::getParameter (
    Parameters param,
    const unsigned int l,
    const int i ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.12 `getParameter()` [2/2]

```
const double CTriangularMeshLinear::getParameter (
    Parameters param,
    const unsigned int l,
    const Point & p ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.13 `getSolution()` [1/2]

```
const std::vector< double > CTriangularMeshLinear::getSolution ( ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.14 `getSolution()` [2/2]

```
const double CTriangularMeshLinear::getSolution (
    const unsigned int element,
    const unsigned int node ) const [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.15 `refine_h()`

```
const int CTriangularMeshLinear::refine_h ( )
```

6.53.2.16 `setParameter()` [1/2]

```
const int CTriangularMeshLinear::setParameter (
    const CParameter & p,
    const unsigned int type )
```

6.53.2.17 `setParameter()` [2/2]

```
const int CTriangularMeshLinear::setParameter (
    Parameters param,
    const double ,
    const unsigned int ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.18 updateSolution() [1/4]

```
const int CTriangularMeshLinear::updateSolution (
    const std::vector< double > & ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.19 updateSolution() [2/4]

```
const int CTriangularMeshLinear::updateSolution (
    const unsigned int element,
    const unsigned int node,
    const double value ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.20 updateSolution() [3/4]

```
const int CTriangularMeshLinear::updateSolution (
    const unsigned int element,
    const unsigned int node,
    CSolution * value ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

6.53.2.21 updateSolution() [4/4]

```
const int CTriangularMeshLinear::updateSolution (
    const unsigned int node,
    const double value ) [virtual]
```

Implements [corenc::Mesh::CMesh<>](#).

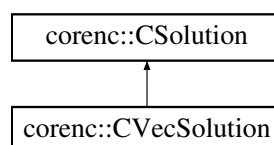
The documentation for this class was generated from the following files:

- CoreNCFEM/Grids/[TriangularMeshLinear.h](#)
- CoreNCFEM/Grids/[TriangularMeshLinear.cpp](#)

6.54 corenc::CVecSolution Class Reference

```
#include <FESolution.h>
```

Inheritance diagram for corenc::CVecSolution:



Public Member Functions

- [CVecSolution](#) ()
- [~CVecSolution](#) ()

Public Attributes

- `std::vector< double >` [m_w](#)

6.54.1 Constructor & Destructor Documentation

6.54.1.1 CVecSolution()

```
corenc::CVecSolution::CVecSolution ( ) [inline]
```

6.54.1.2 ~CVecSolution()

```
corenc::CVecSolution::~~CVecSolution ( ) [inline]
```

6.54.2 Member Data Documentation

6.54.2.1 m_w

```
std::vector<double> corenc::CVecSolution::m_w
```

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

- [CoreNCFEM/FESolution.h](#)

6.55 corenc::solvers::dg_shallow_water< Mesh > Class Template Reference

```
#include <dg_solver_shallow_water.h>
```

Public Member Functions

- [dg_shallow_water](#) ()
- [~dg_shallow_water](#) ()
- const int [solve](#) (const double t0, const double t1, const Mesh &mesh, [vector_solution](#) &sol, const std::function< const std::vector< double >(const std::vector< double > &)> &, const std::function< const std::vector< double >(const std::vector< double > &)> &, const std::function< const std::vector< double >(const std::vector< double > &)> &) const
- const int [solve](#) (const double t0, const double t1, const Mesh &mesh, [vector_solution](#) &sol, std::vector< double > &bath, std::vector< double > &ze, std::vector< double > &dzx, std::vector< double > &dzy, std::vector< double > &dbx, std::vector< double > &dby, const std::function< const std::vector< double >(const std::vector< double > &, const int)> &, const std::function< const std::vector< double >(const std::vector< double > &, const int)> &, const std::function< const std::vector< double >(const std::vector< double > &, const int)> &, const bool WRITE_FILE) const

6.55.1 Constructor & Destructor Documentation

6.55.1.1 dg_shallow_water()

```
template<class Mesh >
corenc::solvers::dg_shallow_water< Mesh >::dg_shallow_water
```

6.55.1.2 ~dg_shallow_water()

```
template<class Mesh >
corenc::solvers::dg_shallow_water< Mesh >::~~dg_shallow_water
```

6.55.2 Member Function Documentation

6.55.2.1 solve() [1/2]

```
template<class Mesh >
const int corenc::solvers::dg_shallow_water< Mesh >::solve (
    const double t0,
    const double t1,
    const Mesh & mesh,
    vector\_solution & sol,
    const std::function< const std::vector< double >(const std::vector< double >
&)> & R,
    const std::function< const std::vector< double >(const std::vector< double >
&)> & G,
    const std::function< const std::vector< double >(const std::vector< double >
&)> & F ) const
```

6.55.2.2 solve() [2/2]

```
template<class Mesh >
const int corenc::solvers::dg_shallow_water< Mesh >::solve (
    const double t0,
    const double t1,
    const Mesh & mesh,
    vector_solution & sol,
    std::vector< double > & bath,
    std::vector< double > & ze,
    std::vector< double > & dzx,
    std::vector< double > & dzy,
    std::vector< double > & dbx,
    std::vector< double > & dby,
    const std::function< const std::vector< double >(const std::vector< double > &,
const int)> & R,
    const std::function< const std::vector< double >(const std::vector< double > &,
const int)> & G,
    const std::function< const std::vector< double >(const std::vector< double > &,
const int)> & F,
    const bool WRITE_FILE ) const
```

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

- Solvers/[dg_solver_shallow_water.h](#)

6.56 corenc::solvers::dg_solver< _Problem, _Mesh, _Result > Class Template Reference

```
#include <dg_solver.h>
```

Public Member Functions

- [dg_solver](#) ()
- [~dg_solver](#) ()
- const int [elliptic_solver](#) (_Problem *, _Mesh *, _Result *)
- const double [get_value](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const _Method *, const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p, const int i) const
- const [Mesh::Point](#) [get_gradvalue](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const [Mesh::Point](#) [get_gradvalue](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p, const int i) const

6.56.1 Constructor & Destructor Documentation

6.56.1.1 dg_solver()

```
template<class _Problem , class _Mesh , class _Result >
corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::dg_solver ( ) [inline]
```

6.56.1.2 ~dg_solver()

```
template<class _Problem , class _Mesh , class _Result >
corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::~~dg_solver ( ) [inline]
```

6.56.2 Member Function Documentation

6.56.2.1 elliptic_solver()

```
template<class _Problem , class _Mesh , class _Result >
const int corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::elliptic_solver (
    _Problem * problem,
    _Mesh * mesh,
    _Result * result )
```

6.56.2.2 get_gradvalue() [1/2]

```
template<class _Problem , class _Mesh , class _Result >
const Mesh::Point corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::get_gradvalue (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

6.56.2.3 get_gradvalue() [2/2]

```
template<class _Problem , class _Mesh , class _Result >
const Mesh::Point corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::get_gradvalue (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p,
    const int i ) const
```

6.56.2.4 get_value() [1/3]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::get_value (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

6.56.2.5 `get_value()` [2/3]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::get_value (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p,
    const int i ) const
```

6.56.2.6 `get_value()` [3/3]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::dg_solver< _Problem, _Mesh, _Result >::get_value (
    const _Method * method2,
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

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

- Solvers/[dg_solver.h](#)

6.57 `corenc::solvers::dg_solver_shallow_water` Class Reference

```
#include <dg_solver_shallow_water.h>
```

Public Member Functions

- [dg_solver_shallow_water](#) ()
- [~dg_solver_shallow_water](#) ()
- const int [solve](#) () const
- const int [solve](#) (const double t0, const double t1, const size_t nx, const size_t ny, const double x0, const double x1, const double y0, const double y1, const double g, const double H, const std::function< const std::vector< double >(const std::vector< double > &)> &, const std::function< const std::vector< double >(const std::vector< double > &)> &, const std::function< const std::vector< double >(const std::vector< double > &)> &) const

6.57.1 Constructor & Destructor Documentation

6.57.1.1 `dg_solver_shallow_water()`

```
dg_solver_shallow_water::dg_solver_shallow_water ( )
```


6.57.1.2 ~dg_solver_shallow_water()

```
dg_solver_shallow_water::~dg_solver_shallow_water ( )
```

6.57.2 Member Function Documentation

6.57.2.1 solve() [1/2]

```
const int dg_solver_shallow_water::solve ( ) const
```

6.57.2.2 solve() [2/2]

```
const int corenc::solvers::dg_solver_shallow_water::solve (
    const double t0,
    const double t1,
    const size_t nx,
    const size_t ny,
    const double x0,
    const double x1,
    const double y0,
    const double y1,
    const double g,
    const double H,
    const std::function< const std::vector< double >(const std::vector< double >
&)> & ,
    const std::function< const std::vector< double >(const std::vector< double >
&)> & ,
    const std::function< const std::vector< double >(const std::vector< double >
&)> & ) const
```

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

- [Solvers/dg_solver_shallow_water.h](#)
- [Solvers/dg_solver_shallow_water.cpp](#)

6.58 corenc::method::DGMethod< Problem, Grid, Matrix > Class Template Reference

```
#include <DGMethod.h>
```

Public Member Functions

- [DGMethod](#) ()
- [DGMethod](#) (Problem *p, Grid *g, Matrix *m, std::vector< double > *rhs)
- [DGMethod](#) (Problem *p, Grid *g, Matrix *m, Matrix *rm, std::vector< double > *rhs)
- [DGMethod](#) (const std::shared_ptr< Grid > &grid)
- [DGMethod](#) (Grid *grid)
- [DGMethod](#) (const [DGMethod](#) &meth)
- void [Discretization](#) ()
- const double [GetValue](#) (const [Mesh::Point](#) &) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec, const int num) const
- const double [GetEffective](#) (const std::vector< double > &vec) const
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &, const int)> [GetValue](#), std::vector< double > &sol)
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &)> [GetValue](#), std::vector< double > &sol, const int)
- void [LoadSolution](#) (const std::vector< double > &vec)
- const std::vector< double > [SetSolution](#) (const int sol, const int liq, const double, const double, const double)
- void [GetSolution](#) (std::vector< double > &vec)
- void [Rediscretization](#) (const std::shared_ptr< Grid > &)
- void [Rediscretization](#) ()
- void [SetTimeStep](#) (const double &step)
- Matrix * [GetGlobalMatrix](#) () const
- Grid * [GetMesh](#) ()
- const std::vector< double > [GetRightVector](#) () const
- void [OutDatFormat](#) (const [Mesh::Point](#) &min, const [Mesh::Point](#) &max, const std::string &file_name, const std::vector< double > &vec) const
- void [OutMeshFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- void [OutMeshTimeFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- [~DGMethod](#) ()

Static Public Member Functions

- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int nfem)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int n)

6.58.1 Constructor & Destructor Documentation

6.58.1.1 DGMethod() [1/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethod< Problem, Grid, Matrix >::DGMethod ( ) [inline]
```

6.58.1.2 DGMethod() [2/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethod< Problem, Grid, Matrix >::DGMethod (
    Problem * p,
    Grid * g,
    Matrix * m,
    std::vector< double > * rhs ) [inline]
```

6.58.1.3 DGMethod() [3/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethod< Problem, Grid, Matrix >::DGMethod (
    Problem * p,
    Grid * g,
    Matrix * m,
    Matrix * rm,
    std::vector< double > * rhs ) [inline]
```

6.58.1.4 DGMethod() [4/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethod< Problem, Grid, Matrix >::DGMethod (
    const std::shared_ptr< Grid > & grid ) [inline]
```

6.58.1.5 DGMethod() [5/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethod< Problem, Grid, Matrix >::DGMethod (
    Grid * grid ) [inline]
```

6.58.1.6 DGMethod() [6/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethod< Problem, Grid, Matrix >::DGMethod (
    const DGMethod< Problem, Grid, Matrix > & meth ) [inline]
```

6.58.1.7 ~DGMethod()

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethod< Problem, Grid, Matrix >::~~DGMethod
```

6.58.2 Member Function Documentation

6.58.2.1 Discretization()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::Discretization
```

6.58.2.2 GetEffective()

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethod< Problem, Grid, Matrix >::GetEffective (
    const std::vector< double > & vec ) const
```

6.58.2.3 GetGlobalMatrix()

```
template<class Problem , class Grid , class Matrix >
Matrix * corenc::method::DGMethod< Problem, Grid, Matrix >::GetGlobalMatrix
```

6.58.2.4 GetGradSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::DGMethod< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.58.2.5 GetGradSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::DGMethod< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int n ) [static]
```

6.58.2.6 GetMesh()

```
template<class Problem , class Grid , class Matrix >
Grid * corenc::method::DGMethod< Problem, Grid, Matrix >::GetMesh ( ) [inline]
```

6.58.2.7 GetRightVector()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::DGMethod< Problem, Grid, Matrix >::GetRightVector
```

6.58.2.8 GetSolution() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethod< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.58.2.9 GetSolution() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethod< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int nfem ) [static]
```

6.58.2.10 GetSolution() [3/3]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::GetSolution (
    std::vector< double > & vec )
```

6.58.2.11 GetValue() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethod< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p ) const
```

6.58.2.12 GetValue() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethod< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec ) const
```

6.58.2.13 GetValue() [3/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethod< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec,
    const int num ) const
```

6.58.2.14 LoadSolution()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::LoadSolution (
    const std::vector< double > & vec )
```

6.58.2.15 OutDatFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::OutDatFormat (
    const Mesh::Point & min,
    const Mesh::Point & max,
    const std::string & file_name,
    const std::vector< double > & vec ) const
```

6.58.2.16 OutMeshFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::OutMeshFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.58.2.17 OutMeshTimeFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::OutMeshTimeFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.58.2.18 ProjectSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &)>
    GetValue,
    std::vector< double > & sol,
    const int )
```

6.58.2.19 ProjectSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &,
const int)> GetValue,
    std::vector< double > & sol )
```

6.58.2.20 Rediscretization() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::Rediscretization
```

6.58.2.21 Rediscretization() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::Rediscretization (
    const std::shared_ptr< Grid > & grid )
```

6.58.2.22 SetSolution()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::DGMethod< Problem, Grid, Matrix >::SetSolution (
    const int sol,
    const int liq,
    const double s,
    const double l,
    const double m )
```

6.58.2.23 SetTimeStep()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethod< Problem, Grid, Matrix >::SetTimeStep (
    const double & step ) [inline]
```

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

- CoreNCFEM/Methods/DGMethod.h

6.59 corenc::method::DGMethodZero< Problem, Grid, Matrix > Class Template Reference

```
#include <DGMethodZero.h>
```

Public Member Functions

- [DGMethodZero](#) ()
- [DGMethodZero](#) (Problem *p, Grid *g, Matrix *m, std::vector< double > *rhs)
- [DGMethodZero](#) (Problem *p, Grid *g, Matrix *m, Matrix *rm, std::vector< double > *rhs)
- [DGMethodZero](#) (const std::shared_ptr< Grid > &grid)
- [DGMethodZero](#) (Grid *grid)
- [DGMethodZero](#) (const [DGMethodZero](#) &meth)
- void [Discretization](#) ()
- const double [GetValue](#) (const [Mesh::Point](#) &) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec, const int num) const
- const double [GetEffective](#) (const std::vector< double > &vec) const
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &, const int)> [GetValue](#), std::vector< double > &sol)
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &)> [GetValue](#), std::vector< double > &sol, const int)
- void [LoadSolution](#) (const std::vector< double > &vec)
- const std::vector< double > [SetSolution](#) (const int sol, const int liq, const double, const double, const double)
- void [GetSolution](#) (std::vector< double > &vec)
- void [Rediscretization](#) (const std::shared_ptr< Grid > &)
- void [Rediscretization](#) ()
- void [SetTimeStep](#) (const double &step)

- Matrix * [GetGlobalMatrix](#) () const
- Grid * [GetMesh](#) ()
- const std::vector< double > [GetRightVector](#) () const
- void [OutDatFormat](#) (const [Mesh::Point](#) &min, const [Mesh::Point](#) &max, const std::string &file_name, const std::vector< double > &vec) const
- void [OutMeshFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- void [OutMeshTimeFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- [~DGMethodZero](#) ()

Static Public Member Functions

- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int nfem)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int n)

6.59.1 Constructor & Destructor Documentation

6.59.1.1 DGMethodZero() [1/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethodZero< Problem, Grid, Matrix >::DGMethodZero ( ) [inline]
```

6.59.1.2 DGMethodZero() [2/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethodZero< Problem, Grid, Matrix >::DGMethodZero (
    Problem * p,
    Grid * g,
    Matrix * m,
    std::vector< double > * rhs ) [inline]
```

6.59.1.3 DGMethodZero() [3/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethodZero< Problem, Grid, Matrix >::DGMethodZero (
    Problem * p,
    Grid * g,
    Matrix * m,
    Matrix * rm,
    std::vector< double > * rhs ) [inline]
```

6.59.1.4 DGMethodZero() [4/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethodZero< Problem, Grid, Matrix >::DGMethodZero (
    const std::shared_ptr< Grid > & grid ) [inline]
```

6.59.1.5 DGMethodZero() [5/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethodZero< Problem, Grid, Matrix >::DGMethodZero (
    Grid * grid ) [inline]
```

6.59.1.6 DGMethodZero() [6/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethodZero< Problem, Grid, Matrix >::DGMethodZero (
    const DGMethodZero< Problem, Grid, Matrix > & meth ) [inline]
```

6.59.1.7 ~DGMethodZero()

```
template<class Problem , class Grid , class Matrix >
corenc::method::DGMethodZero< Problem, Grid, Matrix >::~~DGMethodZero
```

6.59.2 Member Function Documentation**6.59.2.1 Discretization()**

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::Discretization
```

6.59.2.2 GetEffective()

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetEffective (
    const std::vector< double > & vec ) const
```

6.59.2.3 GetGlobalMatrix()

```
template<class Problem , class Grid , class Matrix >
Matrix * corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetGlobalMatrix
```

6.59.2.4 GetGradSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.59.2.5 GetGradSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int n ) [static]
```

6.59.2.6 GetMesh()

```
template<class Problem , class Grid , class Matrix >
Grid * corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetMesh ( ) [inline]
```

6.59.2.7 GetRightVector()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetRight↔
Vector
```

6.59.2.8 GetSolution() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.59.2.9 GetSolution() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int nfem ) [static]
```

6.59.2.10 GetSolution() [3/3]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetSolution (
    std::vector< double > & vec )
```

6.59.2.11 GetValue() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p ) const
```

6.59.2.12 GetValue() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec ) const
```

6.59.2.13 GetValue() [3/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::DGMethodZero< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec,
    const int num ) const
```

6.59.2.14 LoadSolution()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::LoadSolution (
    const std::vector< double > & vec )
```

6.59.2.15 OutDatFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::OutDatFormat (
    const Mesh::Point & min,
    const Mesh::Point & max,
    const std::string & file_name,
    const std::vector< double > & vec ) const
```

6.59.2.16 OutMeshFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::OutMeshFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.59.2.17 OutMeshTimeFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::OutMeshTimeFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.59.2.18 ProjectSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &)>
    GetValue,
    std::vector< double > & sol,
    const int )
```

6.59.2.19 ProjectSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &,
const int)> GetValue,
    std::vector< double > & sol )
```

6.59.2.20 Rediscretization() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::Rediscretization
```

6.59.2.21 Rediscretization() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::Rediscretization (
    const std::shared_ptr< Grid > & grid )
```

6.59.2.22 SetSolution()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::DGMethodZero< Problem, Grid, Matrix >::Set↔
Solution (
    const int sol,
    const int liq,
    const double s,
    const double l,
    const double m )
```

6.59.2.23 SetTimeStep()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::DGMethodZero< Problem, Grid, Matrix >::SetTimeStep (
    const double & step ) [inline]
```

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

- CoreNCFEM/Methods/[DGMethodZero.h](#)

6.60 corenc::method::DGSolution< Grid > Class Template Reference

```
#include <DGSolution.h>
```

Public Member Functions

- [DGSolution](#) ()
- [DGSolution](#) (const std::vector< double > &w)
- [DGSolution](#) (const [DGSolution](#)< Grid > &dg)
- [DGSolution](#)< Grid > & [operator=](#) (const [DGSolution](#)< Grid > &dg)
- [~DGSolution](#) ()
- const double [getWeight](#) (const Grid &g, const [Mesh::Point](#) &p) const
- const std::vector< double > [getWeights](#) () const
- const int [updateWeight](#) (const unsigned int i, const double val)

6.60.1 Constructor & Destructor Documentation

6.60.1.1 DGSolution() [1/3]

```
template<class Grid >
corenc::method::DGSolution< Grid >::DGSolution ( ) [inline]
```

6.60.1.2 DGSolution() [2/3]

```
template<class Grid >
corenc::method::DGSolution< Grid >::DGSolution (
    const std::vector< double > & w ) [inline]
```

6.60.1.3 DGSolution() [3/3]

```
template<class Grid >
corenc::method::DGSolution< Grid >::DGSolution (
    const DGSolution< Grid > & dg ) [inline]
```

6.60.1.4 ~DGSolution()

```
template<class Grid >
corenc::method::DGSolution< Grid >::~~DGSolution ( ) [inline]
```

6.60.2 Member Function Documentation

6.60.2.1 getWeight()

```
template<class Grid >
const double corenc::method::DGSolution< Grid >::getWeight (
    const Grid & g,
    const Mesh::Point & p ) const [inline]
```

6.60.2.2 getWeights()

```
template<class Grid >
const std::vector< double > corenc::method::DGSolution< Grid >::getWeights ( ) const [inline]
```

6.60.2.3 operator=()

```
template<class Grid >
DGSolution< Grid > & corenc::method::DGSolution< Grid >::operator= (
    const DGSolution< Grid > & dg ) [inline]
```

6.60.2.4 updateWeight()

```
template<class Grid >
const int corenc::method::DGSolution< Grid >::updateWeight (
    const unsigned int i,
    const double val ) [inline]
```

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

- CoreNCFEM/Methods/[DGSolution.h](#)

6.61 corenc::solvers::eigen_solver< Matrix, Solver > Class Template Reference

```
#include <eigen_solver.h>
```


Public Member Functions

- [eigen_solver](#) ()
- [~eigen_solver](#) ()
- void [rayleigh](#) (Matrix *A, Matrix *B, Solver *esl, std::complex< double > *mu0, double *x0, const int n) const

6.61.1 Constructor & Destructor Documentation

6.61.1.1 eigen_solver()

```
template<class Matrix , class Solver >
corenc::solvers::eigen_solver< Matrix, Solver >::eigen_solver ( ) [inline]
```

6.61.1.2 ~eigen_solver()

```
template<class Matrix , class Solver >
corenc::solvers::eigen_solver< Matrix, Solver >::~~eigen_solver ( ) [inline]
```

6.61.2 Member Function Documentation

6.61.2.1 rayleigh()

```
template<class Matrix , class Solver >
void corenc::solvers::eigen_solver< Matrix, Solver >::rayleigh (
    Matrix * A,
    Matrix * B,
    Solver * esl,
    std::complex< double > * mu0,
    double * x0,
    const int n ) const [inline]
```

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

- Solvers/[eigen_solver.h](#)

6.62 Algebra::ESolver Class Reference

```
#include <MatrixSkyline.h>
```

Public Member Functions

- [ESolver](#) (const [MatrixSkyline](#) &matrix, const std::vector< double > &rightvector)
- [ESolver](#) ()
- [ESolver](#) ([Solvers](#) kek)
- void [Reload](#) (const [MatrixSkyline](#) &matrix, const std::vector< double > &right)
- void [Solve](#) ([Solvers](#))
- const std::vector< double > [Solve](#) ([MatrixSkyline](#) &, const std::vector< double > &rhs, std::vector< double > &sol, std::vector< double > &residual, const int iter, const double eps)
- const std::vector< double > [Solve](#) ([MatrixDiag](#) &, const std::vector< double > &rhs, std::vector< double > &sol, std::vector< double > &residual, const int iter, const double eps)
- double [BiCGStab](#) (const int _maxiter)
- double [GMRES](#) (const int _maxiter)
- void [GMRES](#) ([MatrixSkyline](#) &, const std::vector< double > &rhs, std::vector< double > &sol, std::vector< double > &residual, const int iter, const double eps)
- void [BiCGStab](#) ([MatrixSkyline](#) &, const std::vector< double > &rhs, std::vector< double > &sol, std::vector< double > &residual, const int iter, const double eps)
- void [Gauss](#) ([MatrixSkyline](#) &, const std::vector< double > &rhs, std::vector< double > &sol, std::vector< double > &residual, const int iter, const double eps)
- void [Gauss](#) ([Matrix](#) &, const std::vector< double > &rhs, std::vector< double > &sol)
- void [Gauss](#) (const [Matrix](#) &, double *in_out)
- void [Gauss](#) (const [Matrix](#) &, double *in, double *out)
- void [Gauss](#) (const [Matrix](#) &, const double *in, double *out)
- void [Pardiso](#) ([MatrixSkyline](#) &, const std::vector< double > &rhs, std::vector< double > &sol)
- void [BiCGStabPrecond](#) ()
- const std::vector< double > [GetSolution](#) () const
- void [GetSolution](#) (std::vector< double > &sol) const
- void [MatrixprodVector](#) (double *res, std::vector< double > &x, [MatrixSkyline](#) &m)
- void [MatrixprodVector](#) (double *res, double *x, [MatrixSkyline](#) &m)
- void [MatrixprodVector](#) (double *res, double *x, const [Matrix](#) &m)
- void [MatrixprodVector](#) (double *res, const double *x, const [Matrix](#) &m)
- [~ESolver](#) ()
- auto [GetSolution](#) () -> decltype(m_solution)

6.62.1 Constructor & Destructor Documentation

6.62.1.1 ESolver() [1/3]

```
Algebra::ESolver::ESolver (
    const MatrixSkyline & matrix,
    const std::vector< double > & rightvector ) [inline]
```

6.62.1.2 ESolver() [2/3]

```
Algebra::ESolver::ESolver ( ) [inline]
```

6.62.1.3 ESolver() [3/3]

```
Algebra::ESolver::ESolver (
    Solvers kek ) [inline]
```

6.62.1.4 ~ESolver()

```
ESolver::~~ESolver ( )
```

6.62.2 Member Function Documentation

6.62.2.1 BiCGStab() [1/2]

```
double ESolver::BiCGStab (
    const int _maxiter )
```

6.62.2.2 BiCGStab() [2/2]

```
void ESolver::BiCGStab (
    MatrixSkyline & matrix,
    const std::vector< double > & rhs,
    std::vector< double > & sol,
    std::vector< double > & residual,
    const int iter,
    const double eps )
```

6.62.2.3 BiCGStabPrecond()

```
void ESolver::BiCGStabPrecond ( )
```

6.62.2.4 Gauss() [1/5]

```
void ESolver::Gauss (
    const Matrix & matrix,
    const double * in,
    double * out )
```

6.62.2.5 Gauss() [2/5]

```
void ESolver::Gauss (
    const Matrix & matrix,
    double * in,
    double * out )
```

6.62.2.6 Gauss() [3/5]

```
void ESolver::Gauss (
    const Matrix & matrix,
    double * in_out )
```

6.62.2.7 Gauss() [4/5]

```
void ESolver::Gauss (
    Matrix & matrix,
    const std::vector< double > & rhs,
    std::vector< double > & sol )
```

6.62.2.8 Gauss() [5/5]

```
void ESolver::Gauss (
    MatrixSkyline & matrix,
    const std::vector< double > & rhs,
    std::vector< double > & sol,
    std::vector< double > & residual,
    const int iter,
    const double eps )
```

6.62.2.9 GetSolution() [1/3]

```
auto Algebra::ESolver::GetSolution ( ) -> decltype(m_solution)    [inline]
```

6.62.2.10 GetSolution() [2/3]

```
const std::vector< double > Algebra::ESolver::GetSolution ( ) const    [inline]
```

6.62.2.11 GetSolution() [3/3]

```
void Algebra::ESolver::GetSolution (
    std::vector< double > & sol ) const
```

6.62.2.12 GMRES() [1/2]

```
double ESolver::GMRES (
    const int _maxiter )
```

6.62.2.13 GMRES() [2/2]

```
void ESolver::GMRES (
    MatrixSkyline & matrix,
    const std::vector< double > & rhs,
    std::vector< double > & sol,
    std::vector< double > & residual,
    const int iter,
    const double eps )
```

6.62.2.14 MatrixprodVector() [1/4]

```
void ESolver::MatrixprodVector (
    double * res,
    const double * x,
    const Matrix & m )
```

6.62.2.15 MatrixprodVector() [2/4]

```
void ESolver::MatrixprodVector (
    double * res,
    double * x,
    const Matrix & m )
```

6.62.2.16 MatrixprodVector() [3/4]

```
void ESolver::MatrixprodVector (
    double * res,
    double * x,
    MatrixSkyline & m )
```

6.62.2.17 MatrixprodVector() [4/4]

```
void Algebra::ESolver::MatrixprodVector (
    double * res,
    std::vector< double > & x,
    MatrixSkyline & m )
```

6.62.2.18 Pardiso()

```
void ESolver::Pardiso (
    MatrixSkyline & matrix,
    const std::vector< double > & rhs,
    std::vector< double > & sol )
```

6.62.2.19 Reload()

```
void ESolver::Reload (
    const MatrixSkyline & matrix,
    const std::vector< double > & right )
```

6.62.2.20 Solve() [1/3]

```
const std::vector< double > ESolver::Solve (
    MatrixDiag & matrix,
    const std::vector< double > & rhs,
    std::vector< double > & sol,
    std::vector< double > & residual,
    const int iter,
    const double eps )
```

6.62.2.21 Solve() [2/3]

```
const std::vector< double > ESolver::Solve (
    MatrixSkyline & matrix,
    const std::vector< double > & rhs,
    std::vector< double > & sol,
    std::vector< double > & residual,
    const int iter,
    const double eps )
```

6.62.2.22 Solve() [3/3]

```
void ESolver::Solve (
    Solvers solver )
```

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

- CoreNCA/[MatrixSkyline.h](#)
- CoreNCA/[MatrixSkyline.cpp](#)

6.63 corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 > Class Template Reference

```
#include <FEAnalysis.h>
```

Public Member Functions

- [FEAnalysis](#) ()
- [~FEAnalysis](#) ()
- const double [L2Norm](#) (const Method1 &method1, const Method2 &method2, const Mesh1 &mesh1, const Mesh2 &mesh2, const std::vector< double > &w1, const std::vector< double > &w2) const

6.63.1 Constructor & Destructor Documentation

6.63.1.1 FEAnalysis()

```
template<class Method1 , class Method2 , class Mesh1 , class Mesh2 >
corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >::FEAnalysis ( ) [inline]
```

6.63.1.2 ~FEAnalysis()

```
template<class Method1 , class Method2 , class Mesh1 , class Mesh2 >
corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >::~~FEAnalysis ( ) [inline]
```

6.63.2 Member Function Documentation

6.63.2.1 L2Norm()

```
template<class Method1 , class Method2 , class Mesh1 , class Mesh2 >
const double corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >::L2Norm (
    const Method1 & method1,
    const Method2 & method2,
    const Mesh1 & mesh1,
    const Mesh2 & mesh2,
    const std::vector< double > & w1,
    const std::vector< double > & w2 ) const
```

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

- CoreNCFEM/Methods/[FEAnalysis.h](#)

6.64 corenc::solvers::fem_solver< _Problem, _Mesh, _Result > Class Template Reference

```
#include <fem_solver.h>
```

Public Member Functions

- [fem_solver](#) ()
- [~fem_solver](#) ()
- const int [elliptic_solver](#) (_Problem *, _Mesh *, _Result *)
- const int [elliptic_solver_gauss](#) (_Problem *, _Mesh *, _Result *)
- const double [get_value](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const [_Method2](#) *, const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const [_Method](#) *, const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p, const int i) const
- const [Mesh::Point](#) [get_gradvalue](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const [Mesh::Point](#) [get_gradvalue](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p, const int i) const

6.64.1 Constructor & Destructor Documentation

6.64.1.1 fem_solver()

```
template<class _Problem , class _Mesh , class _Result >
corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::fem_solver ( ) [inline]
```

6.64.1.2 ~fem_solver()

```
template<class _Problem , class _Mesh , class _Result >
corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::~~fem_solver ( ) [inline]
```


6.64.2 Member Function Documentation

6.64.2.1 elliptic_solver()

```
template<class _Problem , class _Mesh , class _Result >
const int corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::elliptic_solver (
    _Problem * problem,
    _Mesh * mesh,
    _Result * result )
```

6.64.2.2 elliptic_solver_gauss()

```
template<class _Problem , class _Mesh , class _Result >
const int corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::elliptic_solver_gauss (
    _Problem * problem,
    _Mesh * mesh,
    _Result * result )
```

6.64.2.3 get_gradvalue() [1/2]

```
template<class _Problem , class _Mesh , class _Result >
const Mesh::Point corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::get_gradvalue (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

6.64.2.4 get_gradvalue() [2/2]

```
template<class _Problem , class _Mesh , class _Result >
const Mesh::Point corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::get_gradvalue (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p,
    const int i ) const
```

6.64.2.5 `get_value()` [1/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::get_value (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

6.64.2.6 `get_value()` [2/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::get_value (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p,
    const int i ) const
```

6.64.2.7 `get_value()` [3/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::get_value (
    const _Method * ,
    const _Mesh & ,
    const _Result & ,
    const Mesh::Point & p ) const
```

6.64.2.8 `get_value()` [4/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver< _Problem, _Mesh, _Result >::get_value (
    const _Method2 * method2,
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

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

- Solvers/[fem_solver.h](#)

6.65 `corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >` Class Template Reference

```
#include <fem_solver_lib.h>
```

Public Member Functions

- [fem_solver_lib](#) ()
- [~fem_solver_lib](#) ()
- const int [elliptic_solver](#) (_Problem *, _Mesh *, _Result *)
- const int [elliptic_solver_gauss](#) (_Problem *, _Mesh *, _Result *)
- const double [get_value](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const _Method2 *, const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const _Method *, const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const double [get_value](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p, const int i) const
- const [Mesh::Point](#) [get_gradvalue](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p) const
- const [Mesh::Point](#) [get_gradvalue](#) (const _Mesh &, const _Result &, const [Mesh::Point](#) &p, const int i) const

6.65.1 Constructor & Destructor Documentation

6.65.1.1 fem_solver_lib()

```
template<class _Problem , class _Mesh , class _Result >
corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::fem_solver_lib ( ) [inline]
```

6.65.1.2 ~fem_solver_lib()

```
template<class _Problem , class _Mesh , class _Result >
corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::~~fem_solver_lib ( ) [inline]
```

6.65.2 Member Function Documentation

6.65.2.1 elliptic_solver()

```
template<class _Problem , class _Mesh , class _Result >
const int corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::elliptic_solver (
    _Problem * problem,
    _Mesh * mesh,
    _Result * result )
```

6.65.2.2 elliptic_solver_gauss()

```
template<class _Problem , class _Mesh , class _Result >
const int corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::elliptic_solver_gauss (
    _Problem * problem,
    _Mesh * mesh,
    _Result * result )
```

6.65.2.3 get_gradvalue() [1/2]

```
template<class _Problem , class _Mesh , class _Result >
const Mesh::Point corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::get_gradvalue (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

6.65.2.4 get_gradvalue() [2/2]

```
template<class _Problem , class _Mesh , class _Result >
const Mesh::Point corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::get_gradvalue (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p,
    const int i ) const
```

6.65.2.5 get_value() [1/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::get_value (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

6.65.2.6 get_value() [2/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::get_value (
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p,
    const int i ) const
```

6.65.2.7 get_value() [3/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::get_value (
    const _Method * ,
    const _Mesh & ,
    const _Result & ,
    const Mesh::Point & p ) const
```

6.65.2.8 get_value() [4/4]

```
template<class _Problem , class _Mesh , class _Result >
const double corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >::get_value (
    const _Method2 * method2,
    const _Mesh & mesh,
    const _Result & res,
    const Mesh::Point & p ) const
```

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

- Solvers/fem_solver_lib.h

6.66 corenc::method::FEMethod< Problem, Grid, Matrix > Class Template Reference

```
#include <FEMethod.h>
```

Public Member Functions

- [FEMethod](#) ()
- [FEMethod](#) (Problem *p, Grid *g, Matrix *m, std::vector< double > *rhs)
- [FEMethod](#) (Problem *p, Grid *g, Matrix *m, Matrix *rm, std::vector< double > *rhs)
- [FEMethod](#) (const std::shared_ptr< Grid > &grid)
- [FEMethod](#) (Grid *grid)
- [FEMethod](#) (const [FEMethod](#) &meth)
- [FEMethod](#) & operator= (const [FEMethod](#) &fem)
- void [Discretization](#) ()
- const double [GetValue](#) (const [Mesh::Point](#) &) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec, const int num) const
- const double [GetEffective](#) (const std::vector< double > &vec) const
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &, const int)> [GetValue](#), std::vector< double > &sol)
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &)> [GetValue](#), std::vector< double > &sol, const int)
- void [LoadSolution](#) (const std::vector< double > &vec)
- const std::vector< double > [SetSolution](#) (const int sol, const int liq, const double, const double, const double)
- void [GetSolution](#) (std::vector< double > &vec)

- void [Rediscretization](#) (const std::shared_ptr< Grid > &)
- void [Rediscretization](#) ()
- void [SetTimeStep](#) (const double &step)
- Matrix * [GetGlobalMatrix](#) () const
- Grid * [GetMesh](#) ()
- const std::vector< double > [GetRightVector](#) () const
- void [OutDatFormat](#) (const [Mesh::Point](#) &min, const [Mesh::Point](#) &max, const std::string &file_name, const std::vector< double > &vec) const
- void [OutMeshFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- void [OutMeshTimeFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- [~FEMethod](#) ()

Static Public Member Functions

- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int nfem)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int n)

6.66.1 Constructor & Destructor Documentation

6.66.1.1 FEMethod() [1/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethod< Problem, Grid, Matrix >::FEMethod ( ) [inline]
```

6.66.1.2 FEMethod() [2/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethod< Problem, Grid, Matrix >::FEMethod (
    Problem * p,
    Grid * g,
    Matrix * m,
    std::vector< double > * rhs ) [inline]
```

6.66.1.3 FEMethod() [3/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethod< Problem, Grid, Matrix >::FEMethod (
    Problem * p,
    Grid * g,
    Matrix * m,
    Matrix * rm,
    std::vector< double > * rhs ) [inline]
```

6.66.1.4 FEMethod() [4/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethod< Problem, Grid, Matrix >::FEMethod (
    const std::shared_ptr< Grid > & grid ) [inline]
```

6.66.1.5 FEMethod() [5/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethod< Problem, Grid, Matrix >::FEMethod (
    Grid * grid ) [inline]
```

6.66.1.6 FEMethod() [6/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethod< Problem, Grid, Matrix >::FEMethod (
    const FEMethod< Problem, Grid, Matrix > & meth ) [inline]
```

6.66.1.7 ~FEMethod()

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethod< Problem, Grid, Matrix >::~~FEMethod
```

6.66.2 Member Function Documentation

6.66.2.1 Discretization()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::Discretization
```

6.66.2.2 GetEffective()

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethod< Problem, Grid, Matrix >::GetEffective (
    const std::vector< double > & vec ) const
```

6.66.2.3 GetGlobalMatrix()

```
template<class Problem , class Grid , class Matrix >
Matrix * corenc::method::FEMethod< Problem, Grid, Matrix >::GetGlobalMatrix
```

6.66.2.4 GetGradSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::FEMethod< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.66.2.5 GetGradSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::FEMethod< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int n ) [static]
```

6.66.2.6 GetMesh()

```
template<class Problem , class Grid , class Matrix >
Grid * corenc::method::FEMethod< Problem, Grid, Matrix >::GetMesh ( ) [inline]
```


6.66.2.7 GetRightVector()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::FEMethod< Problem, Grid, Matrix >::GetRightVector
```

6.66.2.8 GetSolution() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethod< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.66.2.9 GetSolution() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethod< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int nfem ) [static]
```

6.66.2.10 GetSolution() [3/3]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::GetSolution (
    std::vector< double > & vec )
```

6.66.2.11 GetValue() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethod< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p ) const
```

6.66.2.12 GetValue() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethod< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec ) const
```

6.66.2.13 GetValue() [3/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethod< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec,
    const int num ) const
```

6.66.2.14 LoadSolution()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::LoadSolution (
    const std::vector< double > & vec )
```

6.66.2.15 operator=()

```
template<class Problem , class Grid , class Matrix >
FEMethod & corenc::method::FEMethod< Problem, Grid, Matrix >::operator= (
    const FEMethod< Problem, Grid, Matrix > & fem ) [inline]
```

6.66.2.16 OutDatFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::OutDatFormat (
    const Mesh::Point & min,
    const Mesh::Point & max,
    const std::string & file_name,
    const std::vector< double > & vec ) const
```

6.66.2.17 OutMeshFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::OutMeshFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.66.2.18 OutMeshTimeFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::OutMeshTimeFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.66.2.19 ProjectSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &)>
    GetValue,
    std::vector< double > & sol,
    const int )
```

6.66.2.20 ProjectSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &,
const int)> GetValue,
    std::vector< double > & sol )
```

6.66.2.21 Rediscretization() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::Rediscretization
```

6.66.2.22 Rediscretization() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::Rediscretization (
    const std::shared_ptr< Grid > & grid )
```

6.66.2.23 SetSolution()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::FEMethod< Problem, Grid, Matrix >::SetSolution (
    const int sol,
    const int liq,
    const double s,
    const double l,
    const double m )
```

6.66.2.24 SetTimeStep()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethod< Problem, Grid, Matrix >::SetTimeStep (
    const double & step ) [inline]
```

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

- CoreNCFEM/Methods/[FEMethod.h](#)

6.67 corenc::method::FEMethodZero< Problem, Grid, Matrix > Class Template Reference

```
#include <FEMethodZero.h>
```

Public Member Functions

- [FEMethodZero](#) ()
- [FEMethodZero](#) (Problem *p, Grid *g, Matrix *m, std::vector< double > *rhs)
- [FEMethodZero](#) (Problem *p, Grid *g, Matrix *m, Matrix *rm, std::vector< double > *rhs)
- [FEMethodZero](#) (const std::shared_ptr< Grid > &grid)
- [FEMethodZero](#) (Grid *grid)
- [FEMethodZero](#) (const [FEMethodZero](#) &meth)
- void [Discretization](#) ()
- const double [GetValue](#) (const [Mesh::Point](#) &) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec) const
- const double [GetValue](#) (const [Mesh::Point](#) &, const std::vector< double > &vec, const int num) const
- const double [GetEffective](#) (const std::vector< double > &vec) const
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &, const int)> [GetValue](#), std::vector< double > &sol)
- void [ProjectSolution](#) (std::vector< double > &, std::function< const double(const [Mesh::Point](#) &, const std::vector< double > &)> [GetValue](#), std::vector< double > &sol, const int)
- void [LoadSolution](#) (const std::vector< double > &vec)
- const std::vector< double > [SetSolution](#) (const int sol, const int liq, const double, const double, const double)
- void [GetSolution](#) (std::vector< double > &vec)
- void [Rediscretization](#) (const std::shared_ptr< Grid > &)
- void [Rediscretization](#) ()
- void [SetTimeStep](#) (const double &step)

- Matrix * [GetGlobalMatrix](#) () const
- Grid * [GetMesh](#) ()
- const std::vector< double > [GetRightVector](#) () const
- void [OutDatFormat](#) (const [Mesh::Point](#) &min, const [Mesh::Point](#) &max, const std::string &file_name, const std::vector< double > &vec) const
- void [OutMeshFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- void [OutMeshTimeFormat](#) (const std::string &file_name, const std::vector< double > &vec)
- [~FEMethodZero](#) ()

Static Public Member Functions

- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int nfem)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p)
- static const [Mesh::Point](#) [GetGradSolution](#) (const Grid &g, const std::vector< double > &weights, const [Mesh::Point](#) &p, const int n)

6.67.1 Constructor & Destructor Documentation

6.67.1.1 FEMethodZero() [1/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethodZero< Problem, Grid, Matrix >::FEMethodZero ( ) [inline]
```

6.67.1.2 FEMethodZero() [2/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethodZero< Problem, Grid, Matrix >::FEMethodZero (
    Problem * p,
    Grid * g,
    Matrix * m,
    std::vector< double > * rhs ) [inline]
```

6.67.1.3 FEMethodZero() [3/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethodZero< Problem, Grid, Matrix >::FEMethodZero (
    Problem * p,
    Grid * g,
    Matrix * m,
    Matrix * rm,
    std::vector< double > * rhs ) [inline]
```

6.67.1.4 FEMethodZero() [4/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethodZero< Problem, Grid, Matrix >::FEMethodZero (
    const std::shared_ptr< Grid > & grid ) [inline]
```

6.67.1.5 FEMethodZero() [5/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethodZero< Problem, Grid, Matrix >::FEMethodZero (
    Grid * grid ) [inline]
```

6.67.1.6 FEMethodZero() [6/6]

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethodZero< Problem, Grid, Matrix >::FEMethodZero (
    const FEMethodZero< Problem, Grid, Matrix > & meth ) [inline]
```

6.67.1.7 ~FEMethodZero()

```
template<class Problem , class Grid , class Matrix >
corenc::method::FEMethodZero< Problem, Grid, Matrix >::~~FEMethodZero
```

6.67.2 Member Function Documentation**6.67.2.1 Discretization()**

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::Discretization
```

6.67.2.2 GetEffective()

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetEffective (
    const std::vector< double > & vec ) const
```

6.67.2.3 GetGlobalMatrix()

```
template<class Problem , class Grid , class Matrix >
Matrix * corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetGlobalMatrix
```

6.67.2.4 GetGradSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.67.2.5 GetGradSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
const Mesh::Point corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetGradSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int n ) [static]
```

6.67.2.6 GetMesh()

```
template<class Problem , class Grid , class Matrix >
Grid * corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetMesh ( ) [inline]
```

6.67.2.7 GetRightVector()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetRight↔
Vector
```

6.67.2.8 GetSolution() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p ) [static]
```

6.67.2.9 GetSolution() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & weights,
    const Mesh::Point & p,
    const int nfem ) [static]
```

6.67.2.10 GetSolution() [3/3]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetSolution (
    std::vector< double > & vec )
```

6.67.2.11 GetValue() [1/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p ) const
```

6.67.2.12 GetValue() [2/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec ) const
```

6.67.2.13 GetValue() [3/3]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::FEMethodZero< Problem, Grid, Matrix >::GetValue (
    const Mesh::Point & p,
    const std::vector< double > & vec,
    const int num ) const
```


6.67.2.14 LoadSolution()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::LoadSolution (
    const std::vector< double > & vec )
```

6.67.2.15 OutDatFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::OutDatFormat (
    const Mesh::Point & min,
    const Mesh::Point & max,
    const std::string & file_name,
    const std::vector< double > & vec ) const
```

6.67.2.16 OutMeshFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::OutMeshFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.67.2.17 OutMeshTimeFormat()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::OutMeshTimeFormat (
    const std::string & file_name,
    const std::vector< double > & vec )
```

6.67.2.18 ProjectSolution() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &)>
    GetValue,
    std::vector< double > & sol,
    const int )
```

6.67.2.19 ProjectSolution() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::ProjectSolution (
    std::vector< double > & sol,
    std::function< const double(const Mesh::Point &, const std::vector< double > &,
const int)> GetValue,
    std::vector< double > & sol )
```

6.67.2.20 Rediscretization() [1/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::Rediscretization
```

6.67.2.21 Rediscretization() [2/2]

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::Rediscretization (
    const std::shared_ptr< Grid > & grid )
```

6.67.2.22 SetSolution()

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::FEMethodZero< Problem, Grid, Matrix >::Set↔
Solution (
    const int sol,
    const int liq,
    const double s,
    const double l,
    const double m )
```

6.67.2.23 SetTimeStep()

```
template<class Problem , class Grid , class Matrix >
void corenc::method::FEMethodZero< Problem, Grid, Matrix >::SetTimeStep (
    const double & step ) [inline]
```

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

- CoreNCFEM/Methods/[FEMethodZero.h](#)

6.68 corenc::method::FVMethod1d Class Reference

```
#include <FVMethod.h>
```

Public Member Functions

- [FVMethod1d](#) ()
- [~FVMethod1d](#) ()

Static Public Member Functions

- static const int [Solve](#) ([Mesh::CMesh](#)< [CFESolution](#) > *mesh, const std::function< const double(const double)> &flux_func, const [FVFlux](#) &flux_type, std::vector< double > &new_solution, const double time_↵ step)
- static const double [GetSolution](#) (const [Mesh::CMesh1D](#) &g, const [Mesh::Point](#) &p)

6.68.1 Constructor & Destructor Documentation

6.68.1.1 FVMethod1d()

```
FVMethod1d::FVMethod1d ( )
```

6.68.1.2 ~FVMethod1d()

```
FVMethod1d::~~FVMethod1d ( )
```

6.68.2 Member Function Documentation

6.68.2.1 GetSolution()

```
const double FVMethod1d::GetSolution (
    const Mesh::CMesh1D & g,
    const Mesh::Point & p ) [static]
```

6.68.2.2 Solve()

```
const int FVMethodId::Solve (
    Mesh::CMesh< CFESolution > * mesh,
    const std::function< const double(const double)> & flux_func,
    const FVFlux & flux_type,
    std::vector< double > & new_solution,
    const double time_step ) [static]
```

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

- CoreNCFEM/Methods/[FVMethod.h](#)
- CoreNCFEM/Methods/[FVMethod.cpp](#)

6.69 corenc::Mesh::Gauss1dim Struct Reference

```
#include <Point.h>
```

Static Public Attributes

- static const int [m_order](#) = 13
- static const double [m_a](#) []
- static const double [m_sqrt35](#) = sqrt(3./5.)
- static const double [m_w](#) []

6.69.1 Member Data Documentation

6.69.1.1 m_a

```
const double Gauss1dim::m_a [static]
```

Initial value:

```
=
{
    0,
    -0.2304583159551348,
    0.2304583159551348,
    -0.4484927510364469,
    0.4484927510364469,
    -0.6423493394403402,
    0.6423493394403402,
    -0.8015780907333099,
    0.8015780907333099,
    -0.9175983992229779,
    0.9175983992229779,
    -0.9841830547185881,
    0.9841830547185881
}
```

6.69.1.2 m_order

```
const int Gauss1dim::m_order = 13 [static]
```

6.69.1.3 m_sqrt35

```
const double Gauss1dim::m_sqrt35 = sqrt(3./5.) [static]
```

6.69.1.4 m_w

```
const double Gauss1dim::m_w [static]
```

Initial value:

```
=
{
    0.2325515532308739,
    0.2262831802628972,
    0.2262831802628972,
    0.2078160475368885,
    0.2078160475368885,
    0.1781459807619457,
    0.1781459807619457,
    0.1388735102197872,
    0.1388735102197872,
    0.0921214998377285,
    0.0921214998377285,
    0.0404840047653159,
    0.0404840047653159
}
```

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

- [CoreNCFEM/Point.h](#)
- [CoreNCFEM/Point.cpp](#)

6.70 corenc::Mesh::Gauss1dimN< N > Struct Template Reference

```
#include <Point.h>
```

Static Public Attributes

- static const int [m_order](#)
- static const double [m_a](#) []
- static const double [m_w](#) []

6.70.1 Member Data Documentation

6.70.1.1 m_a

```
template<int N>
const double corenc::Mesh::GaussldimN< N >::m_a[] [static]
```

6.70.1.2 m_order

```
template<int N>
const int corenc::Mesh::GaussldimN< N >::m_order [static]
```

6.70.1.3 m_w

```
template<int N>
const double corenc::Mesh::GaussldimN< N >::m_w[] [static]
```

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

- CoreNCFEM/[Point.h](#)

6.71 corenc::GaussianKernel Struct Reference

```
#include <GaussianField.h>
```

Public Member Functions

- const double [gpexp](#) (const [Mesh::Point](#) &a) const
- const double [gpstep](#) (const [Mesh::Point](#) &a) const
- [GaussianKernel](#) (const int _n, const std::vector< [Mesh::Point](#) > ¢ers)
- const double [get_gp](#) (const std::vector< double > &a, const [Mesh::Point](#) &p) const

Public Attributes

- int [N](#)
- std::vector< [Mesh::Point](#) > [_centrs](#)

6.71.1 Constructor & Destructor Documentation

6.71.1.1 GaussianKernel()

```
corenc::GaussianKernel::GaussianKernel (
    const int _n,
    const std::vector< Mesh::Point > & centers ) [inline]
```

6.71.2 Member Function Documentation

6.71.2.1 get_gp()

```
const double corenc::GaussianKernel::get_gp (
    const std::vector< double > & a,
    const Mesh::Point & p ) const [inline]
```

6.71.2.2 gpexp()

```
const double corenc::GaussianKernel::gpexp (
    const Mesh::Point & a ) const [inline]
```

6.71.2.3 gpstep()

```
const double corenc::GaussianKernel::gpstep (
    const Mesh::Point & a ) const [inline]
```

6.71.3 Member Data Documentation

6.71.3.1 _centrs

```
std::vector<Mesh::Point> corenc::GaussianKernel::_centrs
```

6.71.3.2 N

```
int corenc::GaussianKernel::N
```

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

- CoreNCFEM/[GaussianField.h](#)

6.72 corenc::GaussianProcess Struct Reference

```
#include <GaussianField.h>
```

Public Member Functions

- [GaussianProcess](#) (const double L, const size_t num)
- const double [He](#) (const int i, const double x) const
- const double [phi](#) (const int i, const double x) const

Public Attributes

- double [sigma2](#)
- double [l](#)
- double [a](#)
- double [b](#)
- double [c](#)
- double [A](#)
- double [B](#)
- size_t [K](#) = 1
- std::vector< double > [lambda](#)

6.72.1 Constructor & Destructor Documentation

6.72.1.1 GaussianProcess()

```
corenc::GaussianProcess::GaussianProcess (  
    const double L,  
    const size_t num ) [inline]
```

6.72.2 Member Function Documentation

6.72.2.1 He()

```
const double corenc::GaussianProcess::He (  
    const int i,  
    const double x ) const [inline]
```


6.72.2.2 phi()

```
const double corenc::GaussianProcess::phi (  
    const int i,  
    const double x ) const [inline]
```

6.72.3 Member Data Documentation

6.72.3.1 a

```
double corenc::GaussianProcess::a
```

6.72.3.2 A

```
double corenc::GaussianProcess::A
```

6.72.3.3 b

```
double corenc::GaussianProcess::b
```

6.72.3.4 B

```
double corenc::GaussianProcess::B
```

6.72.3.5 c

```
double corenc::GaussianProcess::c
```

6.72.3.6 K

```
size_t corenc::GaussianProcess::K = 1
```

6.72.3.7 l

```
double corenc::GaussianProcess::l
```

6.72.3.8 lambda

```
std::vector<double> corenc::GaussianProcess::lambda
```

6.72.3.9 sigma2

```
double corenc::GaussianProcess::sigma2
```

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

- CoreNCFEM/[GaussianField.h](#)

6.73 corenc::Mesh::GaussRectangular Struct Reference

```
#include <Point.h>
```

Static Public Attributes

- static const double [m_ra](#) [] = { -[m_c](#), [m_c](#), 0, 0, -[m_a](#), [m_a](#), -[m_a](#), [m_a](#), -[m_b](#), [m_b](#), -[m_b](#), [m_b](#) }
- static const double [m_rb](#) [] = { 0, 0, -[m_c](#), [m_c](#), -[m_a](#), -[m_a](#), [m_a](#), [m_a](#), -[m_b](#), -[m_b](#), [m_b](#), [m_b](#) }
- static const double [m_rw](#) [] = { [m_wc](#), [m_wc](#), [m_wc](#), [m_wc](#), [m_wa](#), [m_wa](#), [m_wa](#), [m_wa](#), [m_wb](#), [m_wb](#), [m_wb](#), [m_wb](#) }
- static const double [m_a](#) = sqrt((114. - 3. * sqrt(583.)) / 287)
- static const double [m_b](#) = sqrt((114. + 3. * sqrt(583.)) / 287)
- static const double [m_c](#) = sqrt(6. / 7)
- static const double [m_wa](#) = 307. / 810 + 923. / (270.*sqrt(583.))
- static const double [m_wb](#) = 307. / 810 - 923. / (270.*sqrt(583.))
- static const double [m_wc](#) = 98./405

6.73.1 Member Data Documentation

6.73.1.1 m_a

```
const double GaussRectangular::m_a = sqrt((114. - 3. * sqrt(583.)) / 287) [static]
```

6.73.1.2 m_b

```
const double GaussRectangular::m_b = sqrt((114. + 3. * sqrt(583.)) / 287) [static]
```

6.73.1.3 m_c

```
const double GaussRectangular::m_c = sqrt(6. / 7) [static]
```

6.73.1.4 m_ra

```
const double GaussRectangular::m_ra = { -m_c, m_c, 0, 0, -m_a, m_a, -m_a, m_a, -m_b, m_b,  
-m_b, m_b } [static]
```

6.73.1.5 m_rb

```
const double GaussRectangular::m_rb = { 0, 0, -m_c, m_c, -m_a, -m_a, m_a, m_a, -m_b, -m_b,  
m_b, m_b } [static]
```

6.73.1.6 m_rw

```
const double GaussRectangular::m_rw = { m_wc, m_wc, m_wc, m_wc, m_wa, m_wa, m_wa, m_wa, m_wb,  
m_wb, m_wb, m_wb } [static]
```

6.73.1.7 m_wa

```
const double GaussRectangular::m_wa = 307. / 810 + 923. / (270.*sqrt(583.)) [static]
```

6.73.1.8 m_wb

```
const double GaussRectangular::m_wb = 307. / 810 - 923. / (270.*sqrt(583.)) [static]
```

6.73.1.9 m_wc

```
const double GaussRectangular::m_wc = 98./405 [static]
```

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

- [CoreNCFEM/Point.h](#)
- [CoreNCFEM/Point.cpp](#)

6.74 corenc::Mesh::GaussRectangularCubic Struct Reference

```
#include <Point.h>
```

Static Public Attributes

- static const double [m_ra](#) []
- static const double [m_rb](#) []
- static const double [m_rc](#) []
- static const double [m_rw](#) []
- static const double [m_a](#) = $\sqrt{6. / 7}$
- static const double [m_b](#) = $\sqrt{(960 - 33. * \sqrt{238.}) / 2726}$
- static const double [m_c](#) = $\sqrt{(960 + 33. * \sqrt{238.}) / 2726}$
- static const double [m_w1](#) = 1078. / 3645
- static const double [m_w2](#) = 343. / 3645
- static const double [m_w3](#) = $43. / 135 + 829. * \sqrt{238.} / 136323$
- static const double [m_w4](#) = $43. / 135 - 829. * \sqrt{238.} / 136323$
- static const int [m_s](#) { 34 }

6.74.1 Member Data Documentation

6.74.1.1 m_a

```
const double GaussRectangularCubic::m_a = sqrt(6. / 7) [static]
```

6.74.1.2 m_b

```
const double GaussRectangularCubic::m_b = sqrt((960 - 33. * sqrt(238.)) / 2726) [static]
```

6.74.1.3 m_c

```
const double GaussRectangularCubic::m_c = sqrt((960 + 33. * sqrt(238.)) / 2726) [static]
```

6.74.1.4 m_ra

```
const double GaussRectangularCubic::m_ra [static]
```

6.74.1.5 m_rb

```
const double GaussRectangularCubic::m_rb [static]
```

6.74.1.6 m_rc

```
const double GaussRectangularCubic::m_rc [static]
```

6.74.1.7 m_rw

```
const double GaussRectangularCubic::m_rw [static]
```

6.74.1.8 m_s

```
const int corenc::Mesh::GaussRectangularCubic::m_s { 34 } [static]
```

6.74.1.9 m_w1

```
const double GaussRectangularCubic::m_w1 = 1078. / 3645 [static]
```

6.74.1.10 m_w2

```
const double GaussRectangularCubic::m_w2 = 343. / 3645 [static]
```

6.74.1.11 m_w3

```
const double GaussRectangularCubic::m_w3 = 43. / 135 + 829. * sqrt(238.) / 136323 [static]
```

6.74.1.12 m_w4

```
const double GaussRectangularCubic::m_w4 = 43. / 135 - 829. * sqrt(238.) / 136323 [static]
```

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

- CoreNCFEM/[Point.h](#)
- CoreNCFEM/[Point.cpp](#)

6.75 corenc::Mesh::GaussTetrahedron Struct Reference

```
#include <Point.h>
```

Static Public Attributes

- static const double [m_la](#) [] = { 1. / 4, 11. / 14, 5. / 70, 5. / 70, 5. / 70, [m_psq](#), [m_msq](#), [m_msq](#), [m_msq](#), [m_psq](#), [m_psq](#) }
- static const double [m_lb](#) [] = { 1. / 4, 5. / 70, 11. / 14, 5. / 70, 5. / 70, [m_msq](#), [m_psq](#), [m_msq](#), [m_psq](#), [m_msq](#), [m_psq](#) }
- static const double [m_lc](#) [] = { 1. / 4, 5. / 70, 5. / 70, 11. / 14, 5. / 70, [m_msq](#), [m_msq](#), [m_psq](#), [m_psq](#), [m_psq](#), [m_msq](#) }
- static const double [m_ld](#) [] = { 1. / 4, 1. / 6, 1. / 6, 1. / 6, 1. / 3 }
- static const double [m_w](#) []
- static const double [m_psq](#) = (1 + sqrt(5. / 14)) / 4
- static const double [m_msq](#) = (1 - sqrt(5. / 14)) / 4

6.75.1 Member Data Documentation

6.75.1.1 m_la

```
const double GaussTetrahedron::m_la = { 1. / 4, 11. / 14, 5. / 70, 5. / 70, 5. / 70, m\_psq, m\_msq, m\_msq, m\_msq, m\_psq, m\_psq } [static]
```

6.75.1.2 m_lb

```
const double GaussTetrahedron::m_lb = { 1. / 4, 5. / 70, 11. / 14, 5. / 70, 5. / 70,
m_msq, m_psq, m_msq, m_psq, m_msq, m_psq } [static]
```

6.75.1.3 m_lc

```
const double GaussTetrahedron::m_lc = { 1. / 4, 5. / 70, 5. / 70, 11. / 14, 5. / 70,
m_msq, m_msq, m_psq, m_psq, m_psq, m_msq } [static]
```

6.75.1.4 m_ld

```
const double GaussTetrahedron::m_ld = { 1. / 4, 1. / 6, 1. / 6, 1. / 6, 1. / 3 } [static]
```

6.75.1.5 m_msq

```
const double GaussTetrahedron::m_msq = (1 - sqrt(5. / 14)) / 4 [static]
```

6.75.1.6 m_psq

```
const double GaussTetrahedron::m_psq = (1 + sqrt(5. / 14)) / 4 [static]
```

6.75.1.7 m_w

```
const double GaussTetrahedron::m_w [static]
```

Initial value:

```
= { -74. / 5625, 343. / 45000, 343. / 45000, 343. / 45000, 343. / 45000,
56. / 2250, 56. / 2250, 56. / 2250, 56. / 2250, 56. / 2250, 56. / 2250 }
```

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

- CoreNCFEM/[Point.h](#)
- CoreNCFEM/[Point.cpp](#)

6.76 corenc::Mesh::GaussTriangle Struct Reference

```
#include <Point.h>
```

Static Public Attributes

- static const double `m_tra` []
- static const double `m_trb` []
- static const double `m_sqrt15` = `sqrt(15.)`
- static const double `m_trw` []
- static const int `m_order` = 7

6.76.1 Member Data Documentation

6.76.1.1 `m_order`

```
const int GaussTriangle::m_order = 7 [static]
```

6.76.1.2 `m_sqrt15`

```
const double GaussTriangle::m_sqrt15 = sqrt(15.) [static]
```

6.76.1.3 `m_tra`

```
const double GaussTriangle::m_tra [static]
```

Initial value:

```
=  
{  
    1. / 3,  
    (6 + m_sqrt15) / 21,  
    (6 + m_sqrt15) / 21,  
    (9 - 2 * m_sqrt15) / 21,  
    (9 + 2 * m_sqrt15) / 21,  
    (6 - m_sqrt15) / 21,  
    (6 - m_sqrt15) / 21,  
}
```

6.76.1.4 `m_trb`

```
const double GaussTriangle::m_trb [static]
```

Initial value:

```
=  
{  
    1. / 3,  
    (9. - 2.*m_sqrt15) / 21,  
    (6. + m_sqrt15) / 21,  
    (6. + m_sqrt15) / 21,  
    (6. - m_sqrt15) / 21,  
    (9. + 2. * m_sqrt15) / 21,  
    (6. - m_sqrt15) / 21  
}
```


6.76.1.5 m_trw

```
const double GaussTriangle::m_trw [static]
```

Initial value:

```
=
{
    9. / 80,
    (155. + m_sqrt15) / 2400,
    (155. + m_sqrt15) / 2400,
    (155. + m_sqrt15) / 2400,
    (155. - m_sqrt15) / 2400,
    (155. - m_sqrt15) / 2400,
    (155. - m_sqrt15) / 2400
}
```

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

- [CoreNCFEM/Point.h](#)
- [CoreNCFEM/Point.cpp](#)

6.77 Algebra::Matrix Class Reference

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

Public Member Functions

- [Matrix](#) (const unsigned int &size, const std::vector< std::set< unsigned int > > &nonzero)
- [Matrix](#) ()
- [~Matrix](#) ()
- void [NullRow](#) (const int row)
- double & [operator\(\)](#) (const int i, const int j)
- const int [GetSize](#) () const
- void [NullMatrix](#) ()
- [Matrix](#) & [operator=](#) (const [Matrix](#) &)
- [Matrix](#) (const [Matrix](#) &matrix)
- void [Create](#) (const unsigned int &size, const std::vector< std::set< unsigned int > > &nonzero)
- void [Create](#) (const unsigned int &size)
- const double [GetElement](#) (const int i, const int j)
- void [AddElement](#) (const unsigned int i, const unsigned int j, const double a)

6.77.1 Detailed Description

The Dense [Matrix](#) Class

6.77.2 Constructor & Destructor Documentation

6.77.2.1 Matrix() [1/3]

```
Algebra::Matrix::Matrix (
    const unsigned int & size,
    const std::vector< std::set< unsigned int > > & nonzero )
```

6.77.2.2 Matrix() [2/3]

```
Algebra::Matrix::Matrix ( ) [inline]
```

6.77.2.3 ~Matrix()

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

6.77.2.4 Matrix() [3/3]

```
Algebra::Matrix::Matrix (
    const Matrix & matrix )
```

6.77.3 Member Function Documentation

6.77.3.1 AddElement()

```
void Algebra::Matrix::AddElement (
    const unsigned int i,
    const unsigned int j,
    const double a ) [inline]
```

6.77.3.2 Create() [1/2]

```
void Algebra::Matrix::Create (
    const unsigned int & size )
```

6.77.3.3 Create() [2/2]

```
void Algebra::Matrix::Create (
    const unsigned int & size,
    const std::vector< std::set< unsigned int > > & nonzero )
```

6.77.3.4 GetElement()

```
const double Algebra::Matrix::GetElement (
    const int i,
    const int j ) [inline]
```

6.77.3.5 GetSize()

```
const int Algebra::Matrix::GetSize ( ) const [inline]
```

6.77.3.6 NullMatrix()

```
void Algebra::Matrix::NullMatrix ( )
```

6.77.3.7 NullRow()

```
void Algebra::Matrix::NullRow (
    const int row )
```

6.77.3.8 operator()()

```
double & Algebra::Matrix::operator() (
    const int i,
    const int j ) [inline]
```

6.77.3.9 operator=()

```
Algebra::Matrix & Algebra::Matrix::operator= (
    const Matrix & matrix )
```

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

- [CoreNCA/Matrix.h](#)
- [CoreNCA/Matrix.cpp](#)

6.78 Algebra::MatrixDiag Class Reference

```
#include <MatrixDiag.h>
```

Public Member Functions

- [MatrixDiag](#) (const unsigned int &size, const std::vector< std::set< unsigned int > > &nonzero)
- [MatrixDiag](#) ()
- [~MatrixDiag](#) ()
- void [NullRow](#) (const int row)
- double & [operator\(\)](#) (const int i, const int j)
- const int [GetSize](#) () const
- void [NullMatrix](#) ()
- [MatrixDiag](#) & [operator=](#) (const [MatrixDiag](#) &)
- [MatrixDiag](#) (const [MatrixDiag](#) &matrix)
- void [Create](#) (const unsigned int &size, const std::vector< std::set< unsigned int > > &nonzero)
- void [AddElement](#) (const unsigned int i, const unsigned int j, const double a)

6.78.1 Detailed Description

The diagonal matrix class

6.78.2 Constructor & Destructor Documentation

6.78.2.1 MatrixDiag() [1/3]

```
Algebra::MatrixDiag::MatrixDiag (
    const unsigned int & size,
    const std::vector< std::set< unsigned int > > & nonzero )
```

6.78.2.2 MatrixDiag() [2/3]

```
Algebra::MatrixDiag::MatrixDiag ( ) [inline]
```

6.78.2.3 ~MatrixDiag()

```
Algebra::MatrixDiag::~~MatrixDiag ( )
```

6.78.2.4 MatrixDiag() [3/3]

```
Algebra::MatrixDiag::MatrixDiag (
    const MatrixDiag & matrix )
```

6.78.3 Member Function Documentation

6.78.3.1 AddElement()

```
void Algebra::MatrixDiag::AddElement (
    const unsigned int i,
    const unsigned int j,
    const double a ) [inline]
```

6.78.3.2 Create()

```
void Algebra::MatrixDiag::Create (
    const unsigned int & size,
    const std::vector< std::set< unsigned int > > & nonzero )
```

6.78.3.3 GetSize()

```
const int Algebra::MatrixDiag::GetSize ( ) const [inline]
```

6.78.3.4 NullMatrix()

```
void Algebra::MatrixDiag::NullMatrix ( )
```

6.78.3.5 NullRow()

```
void Algebra::MatrixDiag::NullRow (
    const int row )
```

6.78.3.6 operator()()

```
double & Algebra::MatrixDiag::operator() (
    const int i,
    const int j ) [inline]
```

6.78.3.7 operator=()

```
Algebra::MatrixDiag & Algebra::MatrixDiag::operator= (
    const MatrixDiag & matrix )
```

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

- [CoreNCA/MatrixDiag.h](#)
- [CoreNCA/MatrixDiag.cpp](#)

6.79 Algebra::MatrixSkyline Class Reference

```
#include <MatrixSkyline.h>
```

Public Member Functions

- [MatrixSkyline](#) (const unsigned int &Size, const std::vector< std::set< unsigned int > > &nonzero)
- [MatrixSkyline](#) ()
- [~MatrixSkyline](#) ()
- void [NullRow](#) (int row)
- double & [operator\(\)](#) (const int i, const int j)
- const double [operator\(\)](#) (const int i, const int j) const
- const double [GetElement](#) (const int i, const int j) const
- const int [GetSize](#) () const
- void [NullMatrix](#) ()
- [MatrixSkyline](#) & [operator=](#) (const [MatrixSkyline](#) &)
- [MatrixSkyline](#) (const [MatrixSkyline](#) &matrix)
- void [Create](#) (const unsigned int &Size, const std::vector< std::set< unsigned int > > &nonzero)
- void [AddElement](#) (const unsigned int i, const unsigned int j, const double a)

Static Public Member Functions

- static const [MatrixSkyline](#) `diff_skymatrix` (const [MatrixSkyline](#) &matrix, const [MatrixSkyline](#) &B, const double scal)
- static const [MatrixSkyline](#) `diff_skymatrix` (const [MatrixSkyline](#) &matrix, const [MatrixSkyline](#) &B, const double a, const double b)
- static const [MatrixSkyline](#) `transpose_sky` (const [MatrixSkyline](#) &matrix)

6.79.1 Detailed Description

The sparse (skyline) matrix format

6.79.2 Constructor & Destructor Documentation

6.79.2.1 [MatrixSkyline\(\)](#) [1/3]

```
Algebra::MatrixSkyline::MatrixSkyline (
    const unsigned int & Size,
    const std::vector< std::set< unsigned int > > & nonzero )
```

6.79.2.2 [MatrixSkyline\(\)](#) [2/3]

```
Algebra::MatrixSkyline::MatrixSkyline ( ) [inline]
```

6.79.2.3 [~MatrixSkyline\(\)](#)

```
MatrixSkyline::~~MatrixSkyline ( )
```

6.79.2.4 [MatrixSkyline\(\)](#) [3/3]

```
MatrixSkyline::MatrixSkyline (
    const MatrixSkyline & matrix )
```

6.79.3 Member Function Documentation

6.79.3.1 AddElement()

```
void Algebra::MatrixSkyline::AddElement (
    const unsigned int i,
    const unsigned int j,
    const double a ) [inline]
```

6.79.3.2 Create()

```
void MatrixSkyline::Create (
    const unsigned int & Size,
    const std::vector< std::set< unsigned int > > & nonzero )
```

6.79.3.3 diff_skymatrix() [1/2]

```
static const MatrixSkyline Algebra::MatrixSkyline::diff_skymatrix (
    const MatrixSkyline & matrix,
    const MatrixSkyline & B,
    const double a,
    const double b ) [inline], [static]
```

6.79.3.4 diff_skymatrix() [2/2]

```
static const MatrixSkyline Algebra::MatrixSkyline::diff_skymatrix (
    const MatrixSkyline & matrix,
    const MatrixSkyline & B,
    const double scal ) [inline], [static]
```

6.79.3.5 GetElement()

```
const double Algebra::MatrixSkyline::GetElement (
    const int i,
    const int j ) const [inline]
```

6.79.3.6 GetSize()

```
const int Algebra::MatrixSkyline::GetSize ( ) const [inline]
```


6.79.3.7 NullMatrix()

```
void MatrixSkyline::NullMatrix ( )
```

6.79.3.8 NullRow()

```
void MatrixSkyline::NullRow (
    int row )
```

6.79.3.9 operator>() [1/2]

```
double & Algebra::MatrixSkyline::operator() (
    const int i,
    const int j ) [inline]
```

6.79.3.10 operator>() [2/2]

```
const double Algebra::MatrixSkyline::operator() (
    const int i,
    const int j ) const [inline]
```

6.79.3.11 operator=()

```
MatrixSkyline & MatrixSkyline::operator= (
    const MatrixSkyline & matrix )
```

6.79.3.12 transpose_sky()

```
static const MatrixSkyline Algebra::MatrixSkyline::transpose_sky (
    const MatrixSkyline & matrix ) [inline], [static]
```

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

- CoreNCA/[MatrixSkyline.h](#)
- CoreNCA/[MatrixSkyline.cpp](#)

6.80 corenc::multi_vector< T > Class Template Reference

```
#include <multi_vector.h>
```

Public Member Functions

- [multi_vector](#) ()
- [multi_vector](#) (const size_t block, const size_t dim)
- [multi_vector](#) (const size_t dim)
- [~multi_vector](#) ()
- const T [get](#) (const size_t i...) const
- const T [get](#) (const std::vector< size_t > &i) const
- const int [set](#) (const T &element, const std::vector< size_t > &index)
- const int [fill_inc](#) ()
- void [resize](#) (const size_t block)
- void [resize](#) (const size_t block, const size_t dim)
- const size_t [size](#) () const
- const size_t [totalsize](#) () const

6.80.1 Constructor & Destructor Documentation

6.80.1.1 multi_vector() [1/3]

```
template<class T >
corenc::multi_vector< T >::multi_vector
```

6.80.1.2 multi_vector() [2/3]

```
template<class T >
corenc::multi_vector< T >::multi_vector (
    const size_t block,
    const size_t dim )
```

6.80.1.3 multi_vector() [3/3]

```
template<class T >
corenc::multi_vector< T >::multi_vector (
    const size_t dim )
```

6.80.1.4 ~multi_vector()

```
template<class T >
corenc::multi_vector< T >::~~multi_vector
```

6.80.2 Member Function Documentation

6.80.2.1 fill_inc()

```
template<class T >
const int corenc::multi_vector< T >::fill_inc
```

6.80.2.2 get() [1/2]

```
template<class T >
const T corenc::multi_vector< T >::get (
    const size_t i... ) const
```

6.80.2.3 get() [2/2]

```
template<class T >
const T corenc::multi_vector< T >::get (
    const std::vector< size_t > & i ) const
```

6.80.2.4 resize() [1/2]

```
template<class T >
void corenc::multi_vector< T >::resize (
    const size_t block )
```

6.80.2.5 resize() [2/2]

```
template<class T >
void corenc::multi_vector< T >::resize (
    const size_t block,
    const size_t dim )
```

6.80.2.6 set()

```
template<class T >
const int corenc::multi_vector< T >::set (
    const T & element,
    const std::vector< size_t > & index )
```

6.80.2.7 size()

```
template<class T >
const size_t corenc::multi_vector< T >::size
```

6.80.2.8 totalsize()

```
template<class T >
const size_t corenc::multi_vector< T >::totalsize
```

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

- CoreNCFEM/[multi_vector.h](#)

6.81 corenc::Mesh::parameter< T > Class Template Reference

```
#include <Parameter.h>
```

Public Types

- using [cfunc](#) = std::function< const T(const int, const int, const [Point](#) &)>
- using [cfunc_old](#) = std::function< const T(const int, const [Point](#) &)>

Public Member Functions

- [parameter](#) ()
- [parameter](#) (const [cfunc](#) &func)
- [parameter](#) (const [cfunc_old](#) &func)
- [parameter](#) (const double _p)
- [parameter](#) (const [Mesh::Point](#) _p)
- [parameter](#) (const [parameter](#)< T > &_p)
- [~parameter](#) ()
- const T [get](#) (const [Point](#) &p) const
- const T [get](#) (const int number, const [Point](#) &p) const
- const T [get](#) (const int element, const int node, const [Point](#) &p) const
- void [set](#) (const [cfunc](#) &func)

6.81.1 Member Typedef Documentation

6.81.1.1 cfunc

```
template<class T >
using corenc::Mesh::parameter< T >::cfunc = std::function<const T(const int, const int, const
Point&)>
```

6.81.1.2 cfunc_old

```
template<class T >
using corenc::Mesh::parameter< T >::cfunc_old = std::function<const T(const int, const Point&)>
```

6.81.2 Constructor & Destructor Documentation

6.81.2.1 parameter() [1/6]

```
template<class T >
corenc::Mesh::parameter< T >::parameter ( ) [inline]
```

6.81.2.2 parameter() [2/6]

```
template<class T >
corenc::Mesh::parameter< T >::parameter (
    const cfunc & func ) [inline]
```

6.81.2.3 parameter() [3/6]

```
template<class T >
corenc::Mesh::parameter< T >::parameter (
    const cfunc_old & func ) [inline]
```

6.81.2.4 parameter() [4/6]

```
template<class T >
corenc::Mesh::parameter< T >::parameter (
    const double _p ) [inline]
```

6.81.2.5 parameter() [5/6]

```
template<class T >
corenc::Mesh::parameter< T >::parameter (
    const Mesh::Point _p ) [inline]
```

6.81.2.6 parameter() [6/6]

```
template<class T >
corenc::Mesh::parameter< T >::parameter (
    const parameter< T > & _p ) [inline]
```

6.81.2.7 ~parameter()

```
template<class T >
corenc::Mesh::parameter< T >::~~parameter ( ) [inline]
```

6.81.3 Member Function Documentation**6.81.3.1 get() [1/3]**

```
template<class T >
const T corenc::Mesh::parameter< T >::get (
    const int element,
    const int node,
    const Point & p ) const [inline]
```

6.81.3.2 get() [2/3]

```
template<class T >
const T corenc::Mesh::parameter< T >::get (
    const int number,
    const Point & p ) const [inline]
```

6.81.3.3 get() [3/3]

```
template<class T >
const T corenc::Mesh::parameter< T >::get (
    const Point & p ) const [inline]
```

6.81.3.4 set()

```
template<class T >
void corenc::Mesh::parameter< T >::set (
    const cfunc & func ) [inline]
```

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

- [CoreNCFEM/Parameter.h](#)

6.82 corenc::Mesh::Point Class Reference

```
#include <Point.h>
```

Public Member Functions

- [Point](#) ()
- [Point](#) (const double _x, const double _y)
- [Point](#) (const double _x, const double _y, const double _z)
- [Point](#) (const [Point](#) &p)
- const double [Jacobian](#) () const
- [Point](#) & [operator=](#) (const [Point](#) &p)
- const bool [operator==](#) (const [Point](#) &p)
- const bool [operator<](#) (const [Point](#) &p2)
- const [Point](#) [operator*](#) (const double rhs)
- [Point](#) & [operator+=](#) (const [Point](#) &rhs)
- [Point](#) & [operator*+=](#) (const double rhs)

Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)

Friends

- const bool [operator!=](#) (const [Point](#) &p1, const [Point](#) &p2)
- const double [operator*](#) (const [Point](#) &lhs, const [Point](#) &rhs)
- const [Point](#) [operator*](#) (const [Point](#) &lhs, const double rhs)
- const [Point](#) [operator*](#) (const double lhs, const [Point](#) &rhs)
- const [Point](#) [operator+](#) (const [Point](#) &lhs, const [Point](#) &rhs)
- const [Point](#) [operator-](#) (const [Point](#) &lhs, const [Point](#) &rhs)

6.82.1 Constructor & Destructor Documentation

6.82.1.1 Point() [1/4]

```
corenc::Mesh::Point::Point ( ) [inline]
```

6.82.1.2 Point() [2/4]

```
corenc::Mesh::Point::Point (
    const double _x,
    const double _y ) [inline]
```

6.82.1.3 Point() [3/4]

```
corenc::Mesh::Point::Point (
    const double _x,
    const double _y,
    const double _z ) [inline]
```

6.82.1.4 Point() [4/4]

```
corenc::Mesh::Point::Point (
    const Point & p ) [inline]
```

6.82.2 Member Function Documentation

6.82.2.1 Jacobian()

```
const double corenc::Mesh::Point::Jacobian ( ) const [inline]
```

6.82.2.2 operator*()

```
const Point corenc::Mesh::Point::operator* (
    const double rhs ) [inline]
```


6.82.2.3 operator*=()

```
Point & corenc::Mesh::Point::operator*= (
    const double rhs ) [inline]
```

6.82.2.4 operator+=()

```
Point & corenc::Mesh::Point::operator+= (
    const Point & rhs ) [inline]
```

6.82.2.5 operator<()

```
const bool corenc::Mesh::Point::operator< (
    const Point & p2 ) [inline]
```

6.82.2.6 operator=()

```
Point & corenc::Mesh::Point::operator= (
    const Point & p ) [inline]
```

6.82.2.7 operator==()

```
const bool corenc::Mesh::Point::operator== (
    const Point & p ) [inline]
```

6.82.3 Friends And Related Function Documentation

6.82.3.1 operator"!=

```
const bool operator!= (
    const Point & p1,
    const Point & p2 ) [friend]
```

6.82.3.2 operator* [1/3]

```
const Point operator* (  
    const double lhs,  
    const Point & rhs ) [friend]
```

6.82.3.3 operator* [2/3]

```
const Point operator* (  
    const Point & lhs,  
    const double rhs ) [friend]
```

6.82.3.4 operator* [3/3]

```
const double operator* (  
    const Point & lhs,  
    const Point & rhs ) [friend]
```

6.82.3.5 operator+

```
const Point operator+ (  
    const Point & lhs,  
    const Point & rhs ) [friend]
```

6.82.3.6 operator-

```
const Point operator- (  
    const Point & lhs,  
    const Point & rhs ) [friend]
```

6.82.4 Member Data Documentation

6.82.4.1 x

```
double corenc::Mesh::Point::x
```

6.82.4.2 y

```
double corenc::Mesh::Point::y
```

6.82.4.3 z

```
double corenc::Mesh::Point::z
```

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

- CoreNCFEM/[Point.h](#)

6.83 corenc::Mesh::point_source< T > Class Template Reference

```
#include <Parameter.h>
```

Public Member Functions

- [point_source](#) ()
- [point_source](#) (const [Mesh::Point](#) &p, const T &val)
- const T [get_value](#) () const
- const [Mesh::Point](#) [get_point](#) () const
- [point_source](#)< T > & [operator=](#) (const [point_source](#)< T > &ps)

6.83.1 Constructor & Destructor Documentation

6.83.1.1 point_source() [1/2]

```
template<class T >
corenc::Mesh::point_source< T >::point_source ( ) [inline]
```

6.83.1.2 point_source() [2/2]

```
template<class T >
corenc::Mesh::point_source< T >::point_source (
    const Mesh::Point & p,
    const T & val ) [inline]
```

6.83.2 Member Function Documentation

6.83.2.1 `get_point()`

```
template<class T >
const Mesh::Point corenc::Mesh::point_source< T >::get_point ( ) const [inline]
```

6.83.2.2 `get_value()`

```
template<class T >
const T corenc::Mesh::point_source< T >::get_value ( ) const [inline]
```

6.83.2.3 `operator=()`

```
template<class T >
point_source< T > & corenc::Mesh::point_source< T >::operator= (
    const point_source< T > & ps ) [inline]
```

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

- CoreNCFEM/[Parameter.h](#)

6.84 `corenc::method::RungeKutta< Problem, Type >` Class Template Reference

```
#include <RungeKutta.h>
```

Public Member Functions

- [RungeKutta](#) ()
- [RungeKutta](#) (const double step, const double final, Problem *problem, const Type *solution)
- const Type [discretize](#) (const Type &solution, const std::function< const Type(const double time, const double time_step, const Type &curr_sol, Type *result)> &func)
- const Type [explicitEuler](#) (const Type &solution, const std::function< const Type(const double time, const double time_step, const Type &curr_sol, Type *result)> &func)
- void [updateTimestep](#) (const double step)
- [~RungeKutta](#) ()

6.84.1 Constructor & Destructor Documentation

6.84.1.1 RungeKutta() [1/2]

```
template<class Problem , class Type >
corenc::method::RungeKutta< Problem, Type >::RungeKutta ( ) [inline]
```

6.84.1.2 RungeKutta() [2/2]

```
template<class Problem , class Type >
corenc::method::RungeKutta< Problem, Type >::RungeKutta (
    const double step,
    const double final,
    Problem * problem,
    const Type * solution ) [inline]
```

6.84.1.3 ~RungeKutta()

```
template<class Problem , class Type >
corenc::method::RungeKutta< Problem, Type >::~~RungeKutta ( ) [inline]
```

6.84.2 Member Function Documentation

6.84.2.1 discretize()

```
template<class Problem , class Type >
const Type corenc::method::RungeKutta< Problem, Type >::discretize (
    const Type & solution,
    const std::function< const Type(const double time, const double time_step, const
Type &curr_sol, Type *result)> & func )
```

6.84.2.2 explicitEuler()

```
template<class Problem , class Type >
const Type corenc::method::RungeKutta< Problem, Type >::explicitEuler (
    const Type & solution,
    const std::function< const Type(const double time, const double time_step, const
Type &curr_sol, Type *result)> & func )
```

6.84.2.3 updateTimeStep()

```
template<class Problem , class Type >
void corenc::method::RungeKutta< Problem, Type >::updateTimeStep (
    const double step ) [inline]
```

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

- CoreNCFEM/Methods/[RungeKutta.h](#)

6.85 corenc::method::STSolution< Grid > Class Template Reference

```
#include <DGSolution.h>
```

Public Member Functions

- [STSolution](#) ()
- [STSolution](#) (const Grid &g)
- [STSolution](#) (const std::vector< [DGSolution](#)< Grid > > &w, const std::vector< double > time, const Grid &g)
- [STSolution](#) (const [STSolution](#)< Grid > &st)
- [STSolution](#)< Grid > & [operator=](#) (const [STSolution](#)< Grid > &st)
- [~STSolution](#) ()
- const double [getWeight](#) (const [Mesh::Point](#) &p, const double time) const
- const int [updateWeight](#) (const std::vector< double > time, const std::vector< [DGSolution](#)< Grid > > w)
- const int [addTimeLayer](#) (const double time, const [DGSolution](#)< Grid > w)
- const std::vector< [DGSolution](#)< Grid > > [getWeights](#) () const

6.85.1 Constructor & Destructor Documentation

6.85.1.1 STSolution() [1/4]

```
template<class Grid >
corenc::method::STSolution< Grid >::STSolution ( ) [inline]
```

6.85.1.2 STSolution() [2/4]

```
template<class Grid >
corenc::method::STSolution< Grid >::STSolution (
    const Grid & g ) [inline]
```

6.85.1.3 STSolution() [3/4]

```
template<class Grid >
corenc::method::STSolution< Grid >::STSolution (
    const std::vector< DGSolution< Grid > > & w,
    const std::vector< double > time,
    const Grid & g ) [inline]
```

6.85.1.4 STSolution() [4/4]

```
template<class Grid >
corenc::method::STSolution< Grid >::STSolution (
    const STSolution< Grid > & st ) [inline]
```

6.85.1.5 ~STSolution()

```
template<class Grid >
corenc::method::STSolution< Grid >::~~STSolution ( ) [inline]
```

6.85.2 Member Function Documentation

6.85.2.1 addTimeLayer()

```
template<class Grid >
const int corenc::method::STSolution< Grid >::addTimeLayer (
    const double time,
    const DGSolution< Grid > w ) [inline]
```

6.85.2.2 getWeight()

```
template<class Grid >
const double corenc::method::STSolution< Grid >::getWeight (
    const Mesh::Point & p,
    const double time ) const [inline]
```

6.85.2.3 getWeights()

```
template<class Grid >
const std::vector< DGSolution< Grid > > corenc::method::STSolution< Grid >::getWeights ( )
const [inline]
```

6.85.2.4 operator=()

```
template<class Grid >
STSolution< Grid > & corenc::method::STSolution< Grid >::operator= (
    const STSolution< Grid > & st ) [inline]
```

6.85.2.5 updateWeight()

```
template<class Grid >
const int corenc::method::STSolution< Grid >::updateWeight (
    const std::vector< double > time,
    const std::vector< DGSolution< Grid > > w ) [inline]
```

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

- CoreNCFEM/Methods/DGSolution.h

6.86 corenc::method::system_dg_method< Problem, Grid, Matrix > Class Template Reference

```
#include <system_dg_method.h>
```

Public Member Functions

- [system_dg_method](#) ()
- [system_dg_method](#) (Problem *p, Grid *g, Matrix *m, const size_t sys_size, std::vector< double > *rhs)
- [~system_dg_method](#) ()
- const int [Assemble](#) ()
- const int [changeFlux](#) (const DGFlux flux_type)
- const Matrix * [GetGlobalMatrix](#) () const
- const std::vector< double > [GetSolution](#) () const
- const double [GetSolution](#) (const std::vector< double > &point) const
- const double [GetMaxSolution](#) () const
- const double [GetMinSolution](#) () const
- const double [GetSolution](#) (const std::vector< double > &dg_sol, const Mesh::Point &p)
- const int [toDGSolution](#) (const Grid &g, std::vector< double > &dg_result) const
- const int [updateWeights](#) (const std::vector< double > &dg_result)
- const int [DGtostandart](#) (const std::vector< double > &dg_result)

Static Public Member Functions

- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &dg_sol, const [Mesh::Point](#) &p)

6.86.1 Constructor & Destructor Documentation

6.86.1.1 system_dg_method() [1/2]

```
template<class Problem , class Grid , class Matrix >
corenc::method::system_dg_method< Problem, Grid, Matrix >::system_dg_method ( ) [inline]
```

6.86.1.2 system_dg_method() [2/2]

```
template<class Problem , class Grid , class Matrix >
corenc::method::system_dg_method< Problem, Grid, Matrix >::system_dg_method (
    Problem * p,
    Grid * g,
    Matrix * m,
    const size_t sys_size,
    std::vector< double > * rhs ) [inline]
```

6.86.1.3 ~system_dg_method()

```
template<class Problem , class Grid , class Matrix >
corenc::method::system_dg_method< Problem, Grid, Matrix >::~~system_dg_method ( ) [inline]
```

6.86.2 Member Function Documentation

6.86.2.1 Assemble()

```
template<class Problem , class Grid , class Matrix >
const int corenc::method::system_dg_method< Problem, Grid, Matrix >::Assemble
```

6.86.2.2 changeFlux()

```
template<class Problem , class Grid , class Matrix >
const int corenc::method::system_dg_method< Problem, Grid, Matrix >::changeFlux (
    const DGFlux flux_type ) [inline]
```

6.86.2.3 DGtostandart()

```
template<class Problem , class Grid , class Matrix >
const int corenc::method::system_dg_method< Problem, Grid, Matrix >::DGtostandart (
    const std::vector< double > & dg_result ) [inline]
```

6.86.2.4 GetGlobalMatrix()

```
template<class Problem , class Grid , class Matrix >
const Matrix * corenc::method::system_dg_method< Problem, Grid, Matrix >::GetGlobalMatrix ( )
const [inline]
```

6.86.2.5 GetMaxSolution()

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::system_dg_method< Problem, Grid, Matrix >::GetMaxSolution
```

6.86.2.6 GetMinSolution()

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::system_dg_method< Problem, Grid, Matrix >::GetMinSolution
```

6.86.2.7 GetSolution() [1/4]

```
template<class Problem , class Grid , class Matrix >
const std::vector< double > corenc::method::system_dg_method< Problem, Grid, Matrix >::Get↵
Solution ( ) const [inline]
```

6.86.2.8 GetSolution() [2/4]

```
template<class Problem , class Grid , class Matrix >
static const double corenc::method::system_dg_method< Problem, Grid, Matrix >::GetSolution (
    const Grid & g,
    const std::vector< double > & dg_sol,
    const Mesh::Point & p ) [inline], [static]
```

6.86.2.9 GetSolution() [3/4]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::system_dg_method< Problem, Grid, Matrix >::GetSolution (
    const std::vector< double > & dg_sol,
    const Mesh::Point & p ) [inline]
```

6.86.2.10 GetSolution() [4/4]

```
template<class Problem , class Grid , class Matrix >
const double corenc::method::system_dg_method< Problem, Grid, Matrix >::GetSolution (
    const std::vector< double > & point ) const
```

6.86.2.11 toDGSolution()

```
template<class Problem , class Grid , class Matrix >
const int corenc::method::system_dg_method< Problem, Grid, Matrix >::toDGSolution (
    const Grid & g,
    std::vector< double > & dg_result ) const [inline]
```

6.86.2.12 updateWeights()

```
template<class Problem , class Grid , class Matrix >
const int corenc::method::system_dg_method< Problem, Grid, Matrix >::updateWeights (
    const std::vector< double > & dg_result ) [inline]
```

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

- CoreNCFEM/Methods/[system_dg_method.h](#)

6.87 corenc::method::system_dg_method< Grid, bool, bool > Class Template Reference

```
#include <system_dg_method.h>
```

Static Public Member Functions

- static const double [GetSolution](#) (const Grid &g, const std::vector< double > &dg_sol, const [Mesh::Point](#) &p)

6.87.1 Member Function Documentation

6.87.1.1 GetSolution()

```
template<class Grid >
static const double corenc::method::system_dg_method< Grid, bool, bool >::GetSolution (
    const Grid & g,
    const std::vector< double > & dg_sol,
    const Mesh::Point & p ) [inline], [static]
```

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

- CoreNCFEM/Methods/[system_dg_method.h](#)

6.88 corenc::test_case_elliptic_fem Class Reference

```
#include <test_case_elliptic_fem.h>
```

Public Member Functions

- [test_case_elliptic_fem](#) ()
- [~test_case_elliptic_fem](#) ()
- const int [elliptic_fem_2d_tri](#) () const
- const int [elliptic_fem_solver](#) () const
- const int [elliptic_fem_square_lin_basis](#) () const
- const int [elliptic_fem_hp_fixed](#) (const int h_ref_max, const int p_ref_max) const
- const int [elliptic_fem_hp_fixed_triangle](#) (const int h_ref_max, const int p_ref_max) const
- const int [elliptic_fem_hp_lagrange_triangle](#) (const int h_ref_max, const int p_ref_max) const
- const int [elliptic_fem_hxhy_fixed_triangle](#) (const int hx_max, const int hy_max) const
- const int [conv_diff_fem_fixed_triangle](#) (const int h_ref_max, const int p_ref_max) const
- const int [global_matrix](#) (const int h_ref_max, const int p_ref_max) const
- const int [elliptic_2layer_fem_2d_tri_h](#) () const
- const int [elliptic_fem_2d_rect_source](#) () const
- const int [elliptic_gaussian_triangle](#) () const
- const int [mass_matrix_3rd_order](#) () const
- const int [strees_matrix_3rd_order](#) () const
- const int [mass_matrix_4th_order](#) () const
- const int [stress_matrix_4th_order](#) () const
- const int [homotopy_conv_diff_fem](#) (const double step) const

6.88.1 Constructor & Destructor Documentation

6.88.1.1 test_case_elliptic_fem()

```
test_case_elliptic_fem::test_case_elliptic_fem ( )
```

6.88.1.2 ~test_case_elliptic_fem()

```
test_case_elliptic_fem::~~test_case_elliptic_fem ( )
```

6.88.2 Member Function Documentation

6.88.2.1 conv_diff_fem_fixed_triangle()

```
const int test_case_elliptic_fem::conv_diff_fem_fixed_triangle (
    const int h_ref_max,
    const int p_ref_max ) const
```

6.88.2.2 elliptic_2layer_fem_2d_tria_h()

```
const int test_case_elliptic_fem::elliptic_2layer_fem_2d_tria_h ( ) const
```

6.88.2.3 elliptic_fem_2d_rect_source()

```
const int test_case_elliptic_fem::elliptic_fem_2d_rect_source ( ) const
```

6.88.2.4 elliptic_fem_2d_tria()

```
const int test_case_elliptic_fem::elliptic_fem_2d_tria ( ) const
```

6.88.2.5 elliptic_fem_hp_fixed()

```
const int test_case_elliptic_fem::elliptic_fem_hp_fixed (
    const int h_ref_max,
    const int p_ref_max ) const
```

6.88.2.6 elliptic_fem_hp_fixed_triangle()

```
const int test_case_elliptic_fem::elliptic_fem_hp_fixed_triangle (
    const int h_ref_max,
    const int p_ref_max ) const
```

6.88.2.7 elliptic_fem_hp_lagrange_triangle()

```
const int test_case_elliptic_fem::elliptic_fem_hp_lagrange_triangle (
    const int h_ref_max,
    const int p_ref_max ) const
```

6.88.2.8 elliptic_fem_hxhy_fixed_triangle()

```
const int test_case_elliptic_fem::elliptic_fem_hxhy_fixed_triangle (
    const int hx_max,
    const int hy_max ) const
```

6.88.2.9 elliptic_fem_solver()

```
const int test_case_elliptic_fem::elliptic_fem_solver ( ) const
```

6.88.2.10 elliptic_fem_square_lin_basis()

```
const int test_case_elliptic_fem::elliptic_fem_square_lin_basis ( ) const
```

6.88.2.11 elliptic_gaussian_triangle()

```
const int test_case_elliptic_fem::elliptic_gaussian_triangle ( ) const
```

6.88.2.12 global_matrix()

```
const int test_case_elliptic_fem::global_matrix (
    const int h_ref_max,
    const int p_ref_max ) const
```

6.88.2.13 homotopy_conv_diff_fem()

```
const int test_case_elliptic_fem::homotopy_conv_diff_fem (
    const double step ) const
```

6.88.2.14 mass_matrix_3rd_order()

```
const int test_case_elliptic_fem::mass_matrix_3rd_order ( ) const
```

6.88.2.15 mass_matrix_4th_order()

```
const int test_case_elliptic_fem::mass_matrix_4th_order ( ) const
```

6.88.2.16 strees_matrix_3rd_order()

```
const int test_case_elliptic_fem::strees_matrix_3rd_order ( ) const
```

6.88.2.17 stress_matrix_4th_order()

```
const int test_case_elliptic_fem::stress_matrix_4th_order ( ) const
```

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

- Tests/[test_case_elliptic_fem.h](#)
- Tests/[test_case_elliptic_fem.cpp](#)

6.89 corenc::tests::test_case_rectanglebasis Class Reference

```
#include <test_case_rectanglebasis.h>
```

Public Member Functions

- [test_case_rectanglebasis](#) ()
- [~test_case_rectanglebasis](#) ()
- const int [mass_matrix](#) () const
- const int [stress_matrix](#) () const

6.89.1 Constructor & Destructor Documentation

6.89.1.1 test_case_rectanglebasis()

```
test_case_rectanglebasis::test_case_rectanglebasis ( )
```

6.89.1.2 ~test_case_rectanglebasis()

```
test_case_rectanglebasis::~~test_case_rectanglebasis ( )
```

6.89.2 Member Function Documentation

6.89.2.1 mass_matrix()

```
const int test_case_rectanglebasis::mass_matrix ( ) const
```

6.89.2.2 stress_matrix()

```
const int test_case_rectanglebasis::stress_matrix ( ) const
```

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

- Tests/FiniteElements/[test_case_rectanglebasis.h](#)
- Tests/FiniteElements/[test_case_rectanglebasis.cpp](#)

6.90 corenc::tests::test_case_regular_mesh Class Reference

```
#include <test_case_regular_mesh.h>
```


Public Member Functions

- [test_case_regular_mesh](#) ()
- [~test_case_regular_mesh](#) ()
- const int [construct_mesh](#) () const

6.90.1 Constructor & Destructor Documentation

6.90.1.1 test_case_regular_mesh()

```
test_case_regular_mesh::test_case_regular_mesh ( )
```

6.90.1.2 ~test_case_regular_mesh()

```
test_case_regular_mesh::~~test_case_regular_mesh ( )
```

6.90.2 Member Function Documentation

6.90.2.1 construct_mesh()

```
const int test_case_regular_mesh::construct_mesh ( ) const
```

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

- Tests/[test_case_regular_mesh.h](#)
- Tests/[test_case_regular_mesh.cpp](#)

6.91 corenc::test_case_solver Class Reference

```
#include <test_case_solver.h>
```

Public Member Functions

- [test_case_solver](#) ()
- [~test_case_solver](#) ()
- const int [gauss_solver](#) () const

6.91.1 Constructor & Destructor Documentation

6.91.1.1 test_case_solver()

```
test_case_solver::test_case_solver ( )
```

6.91.1.2 ~test_case_solver()

```
test_case_solver::~~test_case_solver ( )
```

6.91.2 Member Function Documentation

6.91.2.1 gauss_solver()

```
const int test_case_solver::gauss_solver ( ) const
```

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

- Tests/[test_case_solver.h](#)
- Tests/[test_case_solver.cpp](#)

6.92 corenc::tests::test_case_trianglebasis Class Reference

```
#include <test_case_trianglebasis.h>
```

Public Member Functions

- [test_case_trianglebasis](#) ()
- [~test_case_trianglebasis](#) ()
- const int [mass_matrix](#) () const
- const int [stress_matrix](#) () const

6.92.1 Constructor & Destructor Documentation

6.92.1.1 test_case_trianglebasis()

```
test_case_trianglebasis::test_case_trianglebasis ( )
```

6.92.1.2 ~test_case_trianglebasis()

```
test_case_trianglebasis::~~test_case_trianglebasis ( )
```

6.92.2 Member Function Documentation

6.92.2.1 mass_matrix()

```
const int test_case_trianglebasis::mass_matrix ( ) const
```

6.92.2.2 stress_matrix()

```
const int test_case_trianglebasis::stress_matrix ( ) const
```

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

- Tests/FiniteElements/[test_case_trianglebasis.h](#)
- Tests/FiniteElements/[test_case_trianglebasis.cpp](#)

6.93 corenc::test_cases Class Reference

```
#include <test_cases.h>
```

Public Member Functions

- [test_cases](#) ()
- [~test_cases](#) ()
- const int [perform](#) () const
- const int [perform](#) (const std::function< const int()> &) const
- const int [perform](#) (const std::function< const int(std::ostream &)> &, std::ostream &) const

6.93.1 Constructor & Destructor Documentation

6.93.1.1 test_cases()

```
test_cases::test_cases ( )
```

6.93.1.2 ~test_cases()

```
test_cases::~~test_cases ( )
```

6.93.2 Member Function Documentation

6.93.2.1 perform() [1/3]

```
const int test_cases::perform ( ) const
```

6.93.2.2 perform() [2/3]

```
const int test_cases::perform (
    const std::function< const int()> & f ) const
```

6.93.2.3 perform() [3/3]

```
const int corenc::test_cases::perform (
    const std::function< const int(std::ostream &)> & ,
    std::ostream & ) const
```

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

- Tests/[test_cases.h](#)
- Tests/[test_cases.cpp](#)

6.94 corenc::test_conv_diff Class Reference

```
#include <test_conv_diff.h>
```

Public Member Functions

- [test_conv_diff](#) ()
- [~test_conv_diff](#) ()
- void [conv_diff_fem](#) (const int h_ref_max, const int p_ref_max=1) const
- void [conv_diff_eigen](#) (const int h_ref_max, const int p_ref_max=1) const

6.94.1 Constructor & Destructor Documentation

6.94.1.1 test_conv_diff()

```
corenc::test_conv_diff::test_conv_diff ( ) [inline]
```

6.94.1.2 ~test_conv_diff()

```
corenc::test_conv_diff::~~test_conv_diff ( ) [inline]
```

6.94.2 Member Function Documentation

6.94.2.1 conv_diff_eigen()

```
void test_conv_diff::conv_diff_eigen (
    const int h_ref_max,
    const int p_ref_max = 1 ) const
```

6.94.2.2 conv_diff_fem()

```
void test_conv_diff::conv_diff_fem (
    const int h_ref_max,
    const int p_ref_max = 1 ) const
```

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

- Tests/[test_conv_diff.h](#)
- Tests/[test_conv_diff.cpp](#)

6.95 corenc::solvers::vector_solution Struct Reference

```
#include <dg_solver_shallow_water.h>
```

Public Member Functions

- [vector_solution](#) ()
- [vector_solution](#) (const int _size)

Public Attributes

- std::vector< double > [S](#) [3]

6.95.1 Constructor & Destructor Documentation

6.95.1.1 vector_solution() [1/2]

```
corenc::solvers::vector_solution::vector_solution ( ) [inline]
```

6.95.1.2 vector_solution() [2/2]

```
corenc::solvers::vector_solution::vector_solution (
    const int _size ) [inline]
```

6.95.2 Member Data Documentation

6.95.2.1 S

```
std::vector<double> corenc::solvers::vector_solution::S[3]
```

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

- Solvers/[dg_solver_shallow_water.h](#)

Chapter 7

File Documentation

7.1 colors.h File Reference

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

Namespaces

- namespace `corenc`
- namespace `corenc::color`

Variables

- const std::string `corenc::color::ESCAPE` = "\u001b[0m"
- const std::string `corenc::color::BLACK` = "\u001b[30m"
- const std::string `corenc::color::RED` = "\u001b[31m"
- const std::string `corenc::color::GREEN` = "\u001b[32m"
- const std::string `corenc::color::YELLOW` = "\u001b[33m"
- const std::string `corenc::color::BLUE` = "\u001b[34m"
- const std::string `corenc::color::MAGENTA` = "\u001b[35m"
- const std::string `corenc::color::CYAN` = "\u001b[36m"
- const std::string `corenc::color::WHITE` = "\u001b[37m"
- const std::string `corenc::color::PURPLE` = "\e[1;35m"
- const std::string `corenc::color::BBLACK` = "\u001b[30;1m"
- const std::string `corenc::color::BRED` = "\u001b[31;1m"
- const std::string `corenc::color::BGREEN` = "\u001b[32;1m"
- const std::string `corenc::color::BYELLOW` = "\u001b[33;1m"
- const std::string `corenc::color::BBLUE` = "\u001b[34;1m"
- const std::string `corenc::color::BMAGENTA` = "\u001b[35;1m"
- const std::string `corenc::color::BCYAN` = "\u001b[36;1m"
- const std::string `corenc::color::BWHITE` = "\u001b[37;1m"

7.2 colors.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_COLORS_H
2 #define CORENC_COLORS_H
3 #include <string>
4 #include <iostream>
5 namespace corenc
6 {
7     namespace color
8     {
9         const std::string ESCAPE = "\u001b[0m";
10        // 8-bit colors
11        const std::string BLACK = "\u001b[30m";
12        const std::string RED = "\u001b[31m";
13        const std::string GREEN = "\u001b[32m";
14        const std::string YELLOW = "\u001b[33m";
15        const std::string BLUE = "\u001b[34m";
16        const std::string MAGENTA = "\u001b[35m";
17        const std::string CYAN = "\u001b[36m";
18        const std::string WHITE = "\u001b[37m";
19        const std::string PURPLE = "\e[1;35m";
20        // 16-bit colors
21        const std::string BBLACK = "\u001b[30;1m";
22        const std::string BRED = "\u001b[31;1m";
23        const std::string BGREEN = "\u001b[32;1m";
24        const std::string BYELLOW = "\u001b[33;1m";
25        const std::string BBLUE = "\u001b[34;1m";
26        const std::string BMAGENTA = "\u001b[35;1m";
27        const std::string BCYAN = "\u001b[36;1m";
28        const std::string BWHITE = "\u001b[37;1m";
29        static void color_output(const std::string& text, const std::string& col)
30        {
31            std::cout << col << text << ESCAPE << std::endl;
32        }
33        static void color_output(std::ostream& os, const std::string& text, const std::string& col)
34        {
35            os << col << text << ESCAPE << std::endl;
36        }
37    }
38 }
39 #endif // CORENC_COLORS_H

```

7.3 CoreNCA/Matrix.cpp File Reference

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

7.4 CoreNCA/Matrix.h File Reference

```
#include <vector>
#include <set>
```

Classes

- class [Algebra::Matrix](#)

Namespaces

- namespace [Algebra](#)

7.5 Matrix.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_ALGEBRA_MATRIX_H
2 #define CORENC_ALGEBRA_MATRIX_H
3 #include <vector>
4 #include <set>
5 namespace Algebra
6 {
7     class ESolver;
11    class Matrix
12    {
13    public:
14        Matrix(const unsigned int& size, const std::vector<std::set<unsigned int>& nonzero);
15        Matrix() {};
16        ~Matrix();
17        void NullRow(const int row);
18        double& operator()(const int i, const int j)
19        {
20            return (*this).m_elem[i][j];
21        }
22        const int GetSize() const { return m_size; }
23        void NullMatrix();
24        Matrix& operator=(const Matrix&);
25        Matrix(const Matrix& matrix);
26        void Create(const unsigned int& size, const
27        std::vector<std::set<unsigned int>& nonzero);
28        void Create(const unsigned int& size);
29        const double GetElement(const int i, const int j)
30        {
31            return m_elem[i][j];
32        }
33        void AddElement(const unsigned int i, const unsigned int j,
34        const double a)
35        {
36            m_elem[i][j] += a;
37            return;
38        }
39    private:
40        std::vector<std::vector<double>> m_elem;
41        unsigned int m_size{ 0 };
42        friend ESolver;
43    };
44 #endif // !CORENC_ALGEBRA_MATRIX_H

```

7.6 CoreNCA/MatrixDiag.cpp File Reference

```
#include "MatrixDiag.h"
```

7.7 CoreNCA/MatrixDiag.h File Reference

```
#include <set>
#include <vector>
```

Classes

- class [Algebra::MatrixDiag](#)

Namespaces

- namespace [Algebra](#)

7.8 MatrixDiag.h

[Go to the documentation of this file.](#)

```

1  #ifndef CORENC_ALGEBRA_MATRIXDIAG_H
2  #define CORENC_ALGEBRA_MATRIXDIAG_H
3  #include <set>
4  #include <vector>
5
6  namespace Algebra
7  {
8      class ESolver;
9
10     class MatrixDiag
11     {
12     public:
13         MatrixDiag(const unsigned int& size, const std::vector<std::set<unsigned int>& nonzero);
14         MatrixDiag() {};
15         ~MatrixDiag();
16         void NullRow(const int row);
17         double& operator()(const int i, const int j)
18         {
19             return (*this).m_valDiag[i];
20         }
21         const int GetSize() const { return m_size; }
22         void NullMatrix();
23         MatrixDiag& operator=(const MatrixDiag&);
24         MatrixDiag(const MatrixDiag& matrix);
25         void Create(const unsigned int& size, const
26             std::vector<std::set<unsigned int>& nonzero);
27         void AddElement(const unsigned int i, const unsigned int j,
28             const double a)
29         {
30             if (i == j)
31             {
32                 m_valDiag[i] += a;
33                 return;
34             }
35             return;
36         }
37     private:
38         std::vector<double> m_valDiag;
39         unsigned int m_size{ 0 };
40         friend ESolver;
41     };
42 }
43
44 #endif // !CORENC_ALGEBRA_MATRIXDIAG_H

```

7.9 CoreNCA/MatrixSkyline.cpp File Reference

```

#include "MatrixSkyline.h"
#include <iostream>
#include <fstream>
#include <ostream>
#include <ctime>
#include <cmath>

```

Macros

- `#define N_MIN 4096`
- `#define _NOPE_`

7.9.1 Macro Definition Documentation

7.9.1.1 `_NOPE_`

```
#define _NOPE_
```

7.9.1.2 `N_MIN`

```
#define N_MIN 4096
```

7.10 CoreNCA/MatrixSkyline.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include "Matrix.h"
#include "MatrixDiag.h"
```

Classes

- class [Algebra::MatrixSkyline](#)
- class [Algebra::ESolver](#)

Namespaces

- namespace [Algebra](#)

Enumerations

- enum class [Algebra::Solvers](#) {
 [Algebra::BiCGStab](#) , [Algebra::GMRES](#) , [Algebra::GMRES_BiCGStab](#) , [Algebra::Gauss](#) ,
 [Algebra::PARDISO](#) }

7.11 MatrixSkyline.h

[Go to the documentation of this file.](#)

```

1  #ifndef CORENC_ALGEBRA_MATRIXSKYLINE_H_
2  #define CORENC_ALGEBRA_MATRIXSKYLINE_H_
3  #include <set>
4  #include <vector>
5  #include <memory>
6  #include "Matrix.h"
7  #include "MatrixDiag.h"
8  namespace Algebra
9  {
10     class ESolver;
11     enum class Solvers
12     {
13         BiCGStab,
14         GMRES,
15         GMRES_BiCGStab,
16         Gauss,
17         PARDISO
18     };
19     class MatrixSkyline
20     {
21     public:
22         MatrixSkyline(const unsigned int& Size, const std::vector<std::set<unsigned int>& nonzero);
23         MatrixSkyline(){};
24         ~MatrixSkyline();
25         void NullRow(int row);
26         double& operator()(const int i, const int j)
27         {
28             int ind;
29             /*for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
30                 if (m_colind[ind] == j)
31                     break;
32             return (*this).m_valL[ind];*/
33             if (i == j)
34             {
35                 return (*this).m_valDiag[i];
36             }
37             if (i < j)
38             {
39                 for (ind = m_rowptr[j]; ind < m_rowptr[j + 1]; ++ind)
40                     if (m_colind[ind] == i)
41                         break;
42                 return (*this).m_valU[ind];
43             }
44             else
45             {
46                 for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
47                     if (m_colind[ind] == j)
48                         break;
49                 return (*this).m_valL[ind];
50             }
51             //return (*this)[];
52         }
53         const double operator()(const int i, const int j) const
54         {
55             int ind;
56             /*for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
57                 if (m_colind[ind] == j)
58                     break;
59             return (*this).m_valL[ind];*/
60             if (i == j)
61             {
62                 return (*this).m_valDiag[i];
63             }
64             if (i < j)
65             {
66                 for (ind = m_rowptr[j]; ind < m_rowptr[j + 1]; ++ind)
67                     if (m_colind[ind] == i)
68                         break;
69                 if (ind < m_rowptr[j + 1])
70                     return (*this).m_valU[ind];
71                 return 0;
72             }
73             else
74             {
75                 for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
76                     if (m_colind[ind] == j)
77                         break;
78                 if (ind < m_rowptr[i + 1])
79                     return (*this).m_valL[ind];
80                 return 0;
81             }
82             return 0;
83         }
84     };
85 
```

```

86     }
87     const double      GetElement(const int i, const int j) const
88     {
89         int ind;
90         /*for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
91             if (m_colind[ind] == j)
92                 break;
93         return (*this).m_valL[ind];*/
94         if (i == j)
95         {
96             return (*this).m_valDiag[i];
97         }
98         if (i < j)
99         {
100             for (ind = m_rowptr[j]; ind < m_rowptr[j + 1]; ++ind)
101                 if (m_colind[ind] == i)
102                     break;
103             if (ind < m_rowptr[j + 1])
104                 return (*this).m_valU[ind];
105             return 0;
106         }
107         else
108         {
109             for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
110                 if (m_colind[ind] == j)
111                     break;
112             if (ind < m_rowptr[i + 1])
113                 return (*this).m_valL[ind];
114             return 0;
115         }
116         return 0;
117     }
118     const int          GetSize() const { return m_size; }
119     void               NullMatrix();
120     MatrixSkyline&     operator=(const MatrixSkyline&);
121     //MatrixSkyline&   operator-(const MatrixSkyline&);
122     //friend const MatrixSkyline operator-(const MatrixSkyline&, const MatrixSkyline&);
123     // A - scal * B
124     static const MatrixSkyline diff_skymatrix(const MatrixSkyline& matrix, const MatrixSkyline& B,
125     const double scal)
126     {
127         MatrixSkyline C(matrix);
128         if ((B.m_gsize == matrix.m_gsize) && (matrix.m_size == B.m_size))
129         {
130             for (int i = 0; i < B.m_gsize; ++i)
131             {
132                 C.m_valL[i] = matrix.m_valL[i] - scal * B.m_valL[i];
133                 C.m_valU[i] = matrix.m_valU[i] - scal * B.m_valU[i];
134             }
135             for (int i = 0; i < B.m_size; ++i)
136             {
137                 C.m_valDiag[i] = matrix.m_valDiag[i] - scal * B.m_valDiag[i];
138             }
139             return C;
140         }
141         static const MatrixSkyline diff_skymatrix(const MatrixSkyline& matrix, const MatrixSkyline& B,
142         const double a, const double b)
143         {
144             MatrixSkyline C(matrix);
145             if ((B.m_gsize == matrix.m_gsize) && (matrix.m_size == B.m_size))
146             {
147                 for (int i = 0; i < B.m_gsize; ++i)
148                 {
149                     C.m_valL[i] = a * matrix.m_valL[i] - b * B.m_valL[i];
150                     C.m_valU[i] = a * matrix.m_valU[i] - b * B.m_valU[i];
151                 }
152                 for (int i = 0; i < B.m_size; ++i)
153                 {
154                     C.m_valDiag[i] = a * matrix.m_valDiag[i] - b * B.m_valDiag[i];
155                 }
156                 return C;
157             }
158             static const MatrixSkyline transpose_sky(const MatrixSkyline& matrix)
159             {
160                 MatrixSkyline C(matrix);
161                 C.m_valL = matrix.m_valU;
162                 C.m_valU = matrix.m_valL;
163                 return C;
164             }
165             MatrixSkyline(const MatrixSkyline& matrix);
166             void          Create(const unsigned int& Size, const std::vector<std::set<unsigned int>&
167             nonzero);
168             void          AddElement(const unsigned int i, const unsigned int j, const double a)
169             {
170                 int ind;

```

```

170         /*for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
171             if (m_colind[ind] == j)
172                 break;
173         return (*this).m_valL[ind];*/
174     if (i == j)
175     {
176         m_valDiag[i] += a;
177         return;
178     }
179     if (i < j)
180     {
181         for (ind = m_rowptr[j]; ind < m_rowptr[j + 1]; ++ind)
182             if (m_colind[ind] == i)
183                 break;
184         m_valU[ind] += a;
185         return;
186     }
187     else
188     {
189         for (ind = m_rowptr[i]; ind < m_rowptr[i + 1]; ++ind)
190             if (m_colind[ind] == j)
191                 break;
192         m_valL[ind] += a;
193         return;
194     }
195 }
196
197 private:
198     //int*          m_rowptr = nullptr;
199     std::vector<int> m_rowptr;
200     //int*          m_colind = nullptr;
201     std::vector<int> m_colind;
202     //double*       m_valU = nullptr;
203     std::vector<double> m_valU;
204     //double*       m_valL = nullptr;
205     std::vector<double> m_valL;
206     //double*       m_valDiag = nullptr;
207     std::vector<double> m_valDiag;
208     int            m_size;
209     int            m_gsize;
210     friend         ESolver;
211     //friend         Matrix;
212 };
213 class ESolver
214 {
215 public:
216     ESolver(const MatrixSkyline& matrix, const std::vector<double>& rightvector) :
217         m_matrix{ matrix },
218         m_rightvector( rightvector ),
219         m_maxiter(20000),
220         m_eps(1e-10)
221     {
222         m_solution.resize(matrix.m_size);
223     }
224     ESolver(){};
225     ESolver(Solvers kek) :m_solver(kek){};
226     void Reload(const MatrixSkyline& matrix, const std::vector<double>& right);
227     void Solve(Solvers);
228     const std::vector<double> Solve(MatrixSkyline&, const std::vector<double>& rhs,
std::vector<double>& sol, std::vector<double>& residual, const int iter, const double eps);
229     const std::vector<double> Solve(MatrixDiag&, const std::vector<double>& rhs,
std::vector<double>& sol, std::vector<double>& residual, const int iter, const double eps);
230     double BiCGStab(const int _maxiter);
231     double GMRES(const int _maxiter);
232     void GMRES(MatrixSkyline&, const std::vector<double>& rhs,
std::vector<double>& sol, std::vector<double>& residual, const int iter, const double eps);
233     void BiCGStab(MatrixSkyline&, const std::vector<double>& rhs,
std::vector<double>& sol, std::vector<double>& residual, const int iter, const double eps);
234     void Gauss(MatrixSkyline&, const std::vector<double>& rhs,
std::vector<double>& sol, std::vector<double>& residual, const int iter, const double eps);
235     void Gauss(Matrix&, const std::vector<double>& rhs, std::vector<double>&
sol);
236     void Gauss(const Matrix&, double* in_out);
237     void Gauss(const Matrix&, double* in, double* out);
238     void Gauss(const Matrix&, const double* in, double* out);
239     void Pardiso(MatrixSkyline&, const std::vector<double>& rhs,
std::vector<double>& sol);
240     void BiCGStabPrecond();
241     const std::vector<double> GetSolution() const{ return m_solution; }
242     void GetSolution(std::vector<double>& sol) const;
243     void MatrixprodVector(double*res, std::vector<double>& x, MatrixSkyline& m);
244     void MatrixprodVector(double*res, double* x, MatrixSkyline& m);
245     void MatrixprodVector(double*res, double* x, const Matrix& m);
246     void MatrixprodVector(double*res, const double* x, const Matrix& m);
247     ~ESolver();
248 private:
249     MatrixSkyline m_matrix;

```

```

250         std::vector<std::vector<double>> H, V, W;
251         std::vector<double> m_solution;
252         std::vector<double> m_rightvector;
253         void GMRESBicGStab();
254         void MatrixprodVector(double*res, std::vector<double>& x, double*valDiag,
double*valL, double*valU, int*rowptr, int*colind, int size);
255         void MatrixprodVector(double*res, std::vector<double>& x,
256                                std::vector<double>&valDiag,
257                                std::vector<double>&valL,
258                                std::vector<double>&valU,
259                                std::vector<int>&rowptr,
260                                std::vector<int>&colind, int size);
261         //void MatrixprodVector(double*res, std::vector<double>& x, MatrixSkyline& m);
262         void MatrixprodVector(double*res, double* x,
263                                std::vector<double>&valDiag,
264                                std::vector<double>&valL,
265                                std::vector<double>&valU,
266                                std::vector<int>&rowptr,
267                                std::vector<int>&colind, int size);
268         void MatrixprodVector(double*res, double* x, double*valDiag, double*valL,
double*valU, int*rowptr, int*colind, int size);
269         void MatrixprodVector(double* res, double* x, double* val, int* rowptr, int*
colind, int size);
270         const double DotProd(double*, double*, int);
271         void zero_GMRES(std::vector<std::vector<double>>&, const int str, const int
stl);
272         void mult_Ht_H_slae(double, double*, int);
273         void gauss(std::vector<std::vector<double>>&, double*, double*, int);
274         int find_max(std::vector<std::vector<double>>&, int, int);
275         void Copy(double *x, double *y, int n);
276         void mult_Vy(double*, double*, int);
277         void mult_Vy(double*, double*, int, int);
278         const double DotProd(const std::vector<double>&, const std::vector<double>&, int);
279         const double Norm(double*, int);
280         const double Norm(const std::vector<double>&, int);
281         double m_eps;
282         Solvers m_solver;
283         int m_maxiter;
284         double* m_LUvalL = nullptr;
285         double* m_LUvalU = nullptr;
286         double* m_LUvalDiag = nullptr;
287         void LUPrec();
288         void LSolve(double*, double*);
289         void USolve(double*, double*);
290     public:
291         auto GetSolution() -> decltype(m_solution) { return m_solution; }
292     };
293 }
294 #endif // CORENC_ALGEBRA_MATRIXSKYLINE_H_

```

7.12 CoreNCFEM/FESolution.h File Reference

```

#include <vector>
#include "Point.h"

```

Classes

- class [corenc::CSolution](#)
- class [corenc::CFESolution](#)
- class [corenc::CVecSolution](#)
- class [corenc::CFEweights](#)

Namespaces

- namespace [corenc](#)

7.13 FESolution.h

[Go to the documentation of this file.](#)

```

1  #ifndef CORENC_FESOLUTION_H
2  #define CORENC_FESOLUTION_H
9  #include <vector>
10 #include "Point.h"
11 namespace corenc
12 {
13     class CSolution
14     {
15     public:
16         CSolution() {};
17         virtual ~CSolution() {};
18     };
19
20     class CFESolution :public CSolution
21     {
22     public:
23         CFESolution() :m_w{ 0 } {};
24         ~CFESolution() {}
25         CFESolution& operator=(const CFESolution& fe)
26         {
27             m_w = fe.m_w;
28             return *this;
29         }
30         CFESolution& operator=(const double fe)
31         {
32             m_w = fe;
33             return *this;
34         }
35         CFESolution(const CFESolution& fe) :m_w{ fe.m_w } {}
36         CFESolution(const double& fe) : m_w{ fe } {}
37         operator double() const { return m_w; }
38         /*double& operator=(const double fe)
39         {
40             m_w = fe;
41             return m_w;
42         }*/
43         const bool operator==(const CFESolution& fe)
44         {
45             if (fe.m_w == m_w)
46                 return true;
47             return false;
48         }
49         const bool operator!=(const CFESolution& fe)
50         {
51             if (fe.m_w != m_w)
52                 return true;
53             return false;
54         }
55         CFESolution& operator+=(const CFESolution& fe)
56         {
57             m_w += fe.m_w;
58             return *this;
59         }
60         CFESolution& operator-=(const CFESolution& fe)
61         {
62             m_w -= fe.m_w;
63             return *this;
64         }
65         CFESolution& operator*=(const CFESolution& fe)
66         {
67             m_w *= fe.m_w;
68             return *this;
69         }
70         CFESolution& operator/=(const CFESolution& fe)
71         {
72             m_w /= fe.m_w;
73             return *this;
74         }
75         friend const double operator*(const CFESolution& lhs, const CFESolution& rhs)
76         {
77             return lhs.m_w * rhs.m_w;
78         }
79         friend const double operator*(const CFESolution& lhs, const double rhs)
80         {
81             return lhs.m_w * rhs;
82         }
83         friend const double operator*(const double lhs, const CFESolution& rhs)
84         {
85             return lhs * rhs.m_w;
86         }
87         friend const double operator-(const CFESolution& lhs, const CFESolution& rhs)
88         {

```



```

89         return lhs.m_w - rhs.m_w;
90     }
91     friend const double operator+(const CFESolution& lhs, const CFESolution& rhs)
92     {
93         return lhs.m_w + rhs.m_w;
94     }
95     friend const double operator/(const CFESolution& lhs, const CFESolution& rhs)
96     {
97         return lhs.m_w / rhs.m_w;
98     }
99     private:
100         double m_w;
101     };
102     class CVecSolution :public CSolution
103     {
104     public:
105         CVecSolution() :m_w{ 0 } {};
106         ~CVecSolution() {};
107         std::vector<double> m_w;
108     };
109
110     class CFWeights
111     {
112     public:
113         CFWeights() {};
114         ~CFWeights()
115         {
116             if (m_w.size() > 0)
117                 std::vector<CFESolution>().swap(m_w);
118         };
119         const CFESolution          getWeight(const unsigned int i) const { return m_w[i]; };
120         const int                  updateWeight(const unsigned int i, const CFESolution& cfe)
121         {
122             if (i < m_w.size())
123             {
124                 m_w[i] = cfe;
125                 return 0;
126             }
127             return 1;
128         }
129     private:
130         std::vector<CFESolution>    m_w;
131     };
132 }
133
134 #endif // !CORENC_FESOLUTION_H
135

```

7.14 CoreNCFEM/FiniteElements/CRectangleBasis2x.cpp File Reference

```

#include "Rectangle.h"
#include <iostream>

```

7.15 CoreNCFEM/FiniteElements/Cube.cpp File Reference

```

#include "Cube.h"

```

7.16 CoreNCFEM/FiniteElements/Cube.h File Reference

```

#include <stdio.h>
#include "Shape.h"
#include "ShapeFunction.h"
#include <iostream>

```

Classes

- class `corenc::Mesh::CCube`
- class `corenc::Mesh::CCubeBasis`

Namespaces

- namespace `corenc`
- namespace `corenc::Mesh`

Macros

- `#define CORENC_MESH_CUBE_H_`

7.16.1 Macro Definition Documentation

7.16.1.1 CORENC_MESH_CUBE_H_

```
#define CORENC_MESH_CUBE_H_
```

7.17 Cube.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef CORENC_MESH_CUBE_H_
3 #define CORENC_MESH_CUBE_H_
4
5 #include <stdio.h>
6 #include "Shape.h"
7 #include "ShapeFunction.h"
8 #include <iostream>
9 namespace corenc
10 {
11     namespace Mesh
12     {
13         class CCube : public CShape
14         {
15         public:
16             CCube();
17             CCube(const int n1, const int n2, const int n3, const int n4, const int order);
18             CCube(const int n1, const int n2, const int n3, const int n4, const int e1, const int e2,
19                 const int e3, const int e4, const int order);
20             CCube(const int*, const int order);
21             CCube(const int*, const int*, const int order);
22             CCube(const CCube&);
23             CCube& operator=(const CCube& t)
24             {
25                 m_nodes = t.m_nodes;
26                 m_edges[0] = t.m_edges[0];
27                 m_edges[1] = t.m_edges[1];
28                 m_edges[2] = t.m_edges[2];
29                 m_edges[3] = t.m_edges[3];
30                 m_number = t.m_number;
31                 m_order = t.m_order;
32                 m_px = t.m_px;
33                 m_py = t.m_py;
34                 return *this;
35             }
36             const bool operator==(const CCube& t)
```

```

36         {
37             for (unsigned int i = 0; i < 4; ++i)
38                 if (m_nodes[i] == t.m_nodes[0])
39                     for (unsigned int j = 0; j < 4; ++j)
40                         if (m_nodes[j] == t.m_nodes[1])
41                             for (unsigned int k = 0; k < 4; ++k)
42                                 if (m_nodes[k] == t.m_nodes[2])
43                                     for (unsigned int l = 0; l < 4; ++l)
44                                         if (m_nodes[l] == t.m_nodes[3])
45                                             return true;
46             return false;
47         }
48     std::istream& operator>>(std::istream& is)
49     {
50         is >> m_nodes[0] >> m_nodes[1] >> m_nodes[2] >> m_nodes[3];
51         return is;
52     }
53     ~CCube() {}
54     const int GetNode(const int) const;
55     const int GetNode(const NODES&) const;
56     const int GetEdge(const int) const;
57     const int GetFacet(const int) const;
58     const int GetNumberOfNodes() const;
59     const int GetNumberOfEdges() const;
60     const int GetNumberOfFacets() const;
61     const double Integrate(const std::function<const
double(const Point&)>&, const std::vector<Point>& v) const;
62     const Point Integrate(const std::function<const
Point(const Point&)>&, const std::vector<Point>& v) const;
63     const std::vector<double> Integrate(const std::function<const
std::vector<double>(const Point&)>&, const std::vector<Point>&) const;
64     void SetNode(const int k, const int node);
65     const int IncreaseOrder();
66     const int SetOrder(const int px, const int py);
67     void SetEdge(const int k, const int edge);
68     void SetFacet(const int k, const int facet);
69     private:
70         std::vector<int> m_nodes;
71         int m_edges[4];
72         int m_order;
73         int m_number;
74         int m_px, m_py;
75     };
76
77     class CCubeBasis : public CShapeFunction<double>
78     {
79     public:
80         CCubeBasis();
81         CCubeBasis(const Point&, const Point&, const Point&, const Point&, const int order);
82         CCubeBasis(const Point*, const int order);
83         CCubeBasis(const CCubeBasis&);
84         CCubeBasis& operator=(const CCubeBasis& t)
85         {
86             m_normal = t.m_normal;
87             m_det = t.m_det;
88             m_order = t.m_order;
89             m_ldorder = t.m_ldorder;
90             m_number = t.m_number;
91             m_s = t.m_s;
92             m_sp = t.m_sp;
93             m_points = t.m_points;
94             m_hx = t.m_hx;
95             m_hy = t.m_hy;
96             return *this;
97         }
98         ~CCubeBasis() {}
99         const int GetNumberOfShapeFunctions() const;
100         //const DForm<0>* GetShapeFunction(const int, const
Point&) const;
101         const double GetShapeFunction(const int, const
Point&) const;
102         const Point GetGradShapeFunction(const int, const
Point&) const;
103         const Point GetNormal() const;
104         void ReverseNormal();
105         const double GetValue(const Point&) const;
106         const int IncreaseOrder();
107         //const int SetValue(const int, CSolution* value);
108         //CSolution* GetValue(const unsigned int);
109         //const CFESolution GetValue(const int) const;
110         const double GetMeasure() const { return m_det; };
111         const double GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
112         //const unsigned int GetOrder() const;
113         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
114     private:
115         int m_number;

```

```

116         int
117         int
118         Point
119         std::vector<Mesh::Point>
120         double
121         double
122         const double
123         const double
124         const double
125         const double
126         void
127         Point&);
128         void
129         const Point&);
130         const int
131         //std::vector<double>
132         //std::vector<CFESolution>
133         int
134         int
135     };
136 }
137 #endif // CORENC_MESH_Cube_H_

```

```

m_order;
m_ldorder;
m_normal;
m_points;
m_det;
m_hx, m_hy;
m_x1(const double) const;
m_x2(const double) const;
m_y1(const double) const;
m_y2(const double) const;
compD(const Point&, const Point&, const
compNormal(const Point&, const Point&,
createS();
    m_w[m_number];
m_w{ 4 };
m_s;
m_sp;

```

7.18 CoreNCFEM/FiniteElements/CubeHBasis.cpp File Reference

7.19 CoreNCFEM/FiniteElements/Edge.cpp File Reference

```

#include "Edge.h"
#include <iostream>

```

7.20 CoreNCFEM/FiniteElements/Edge.h File Reference

```

#include <stdio.h>
#include "Shape.h"
#include "ShapeFunction.h"
#include <iostream>
#include "../FESolution.h"

```

Classes

- class [corenc::Mesh::CEdge](#)
- class [corenc::Mesh::CEdgeLinearBasis](#)
- class [corenc::Mesh::CEdgeConstantBasis](#)
- class [corenc::Mesh::CEdgeMultiBasis](#)
- class [corenc::Mesh::CEdgeHermiteBasis](#)
- class [corenc::Mesh::CEdge2ndBasis](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.21 Edge.h

[Go to the documentation of this file.](#)

```

1  #ifndef Edge_hpp
2  #define Edge_hpp
3
4  #include <stdio.h>
5  #include "Shape.h"
6  #include "ShapeFunction.h"
7  #include <iostream>
8  #include "../FESolution.h"
9  namespace corenc
10 {
11     namespace Mesh
12     {
13         class CEdge : public CShape
14         {
15         public:
16             CEdge();
17             CEdge(const CEdge&);
18             CEdge(const int n1, const int n2);
19             CEdge(const int*);
20             CEdge& operator=(const CEdge& e)
21             {
22                 m_nodes = e.m_nodes;
23                 m_number = e.m_number;
24                 return *this;
25             }
26             friend const bool operator==(const CEdge& e1, const CEdge& e2)
27             {
28                 if (e1.m_nodes[0] == e2.m_nodes[0])
29                     if (e1.m_nodes[1] == e2.m_nodes[1])
30                         return true;
31                 if (e1.m_nodes[1] == e2.m_nodes[0])
32                     if (e1.m_nodes[0] == e2.m_nodes[1])
33                         return true;
34                 return false;
35             }
36             friend std::istream& operator>>(std::istream& is, CEdge& e)
37             {
38                 is >> e.m_nodes[0] >> e.m_nodes[1];
39                 --e.m_nodes[0]; --e.m_nodes[1];
40                 return is;
41             }
42             ~CEdge() {}
43             const int
44             const int
45             const int
46             void
47             const double
48             double(const Point&)>&, const std::vector<Point>& v) const;
49             const Point
50             std::function<const Point(const Point&)>&, const std::vector<Point>& v) const;
51             const int
52             const std::vector<double>
53             std::vector<double>(const Point&)>&, const std::vector<Point>&) const;
54             private:
55                 int
56                 std::vector<int>
57             };
58             class CEdgeLinearBasis : public CShapeFunction<double>
59             {
60             public:
61                 CEdgeLinearBasis();
62                 CEdgeLinearBasis(const Point&, const Point&);
63                 CEdgeLinearBasis(const Point*);
64                 CEdgeLinearBasis(const CEdgeLinearBasis&);
65                 CEdgeLinearBasis& operator=(const CEdgeLinearBasis& e)
66                 {
67                     m_number = e.m_number;
68                     m_p0 = e.m_p0;
69                     m_p1 = e.m_p1;
70                     m_normal = e.m_normal;
71                     m_mes = e.m_mes;
72                     return *this;
73                 }
74                 ~CEdgeLinearBasis() {}
75                 const int
76                 //const DForm<0>*
77             const;
78             const double
79             Point& const;
80             const Point
81             const Point& const;
82             GetNode(const int) const;
83             GetNode(const NODES&) const;
84             GetNumberOfNodes() const;
85             SetNode(const int k, const int node);
86             Integrate(const std::function<const
87             Integrate(const
88             IncreaseOrder();
89             Integrate(const std::function<const
90             const;
91             m_number;
92             m_nodes;
93             GetNumberOfShapeFunctions() const;
94             GetShapeFunction(const int)
95             GetShapeFunction(const int, const
96             GetGradShapeFunction(const int,

```

```

77         const Point                                     GetNormal() const;
78         void                                             ReverseNormal();
79         //const int                                     SetValue(const int, CSolution*
value);
80         //const int                                     SetValue(const int, const
CFESolution& value);
81         //const CFESolution                             GetValue(const Point&) const;
82         //const CFESolution                             GetValue(const int) const;
83         const int                                       IncreaseOrder();
84         const double                                    GetMeasure() const { return m_mes; };
85         const double                                    GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
86         //CSolution*                                    GetValue(const unsigned int);
87         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int)
const;
88     private:
89         int                                             m_number;
90         Point                                           m_p0, m_p1;
91         Point                                           m_normal;
92         double                                          m_mes;
93         void                                            CompNormal();
94         void                                            CompLenght();
95         //std::vector<double>                          m_w[2];
96         //std::vector<CFESolution>                     m_w;
97         //const std::function<const double(const Point&p)> m_psi[2];
98     };
99
100
101
102
103     class CEdgeConstantBasis : public CShapeFunction<double>
104     {
105     public:
106         CEdgeConstantBasis();
107         CEdgeConstantBasis(const Point&, const Point&);
108         CEdgeConstantBasis(const Point*);
109         CEdgeConstantBasis(const CEdgeConstantBasis&);
110         CEdgeConstantBasis& operator=(const CEdgeConstantBasis& e)
111         {
112             m_p0 = e.m_p0;
113             m_p1 = e.m_p1;
114             m_normal = e.m_normal;
115             m_mes = e.m_mes;
116             return *this;
117         }
118         ~CEdgeConstantBasis() {};
119         const int                                       GetNumberOfShapeFunctions() const;
120         //const DForm<0>*                               GetShapeFunction(const int)
const;
121         const double                                   GetShapeFunction(const int, const
Point&) const;
122         const Point                                   GetGradShapeFunction(const int,
const Point&) const;
123         const Point                                   GetNormal() const;
124         void                                           ReverseNormal();
125         const double                                   GetWeight(const int node, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
126         {
127             return f(verts[node]);
128         };
129         //const double                                   GetValue(const Point&) const;
130         //const int                                     SetValue(const unsigned int, const
double& value);
131         //const double                                   GetValue(const unsigned int) const;
132         //const int                                     SetValue(const int, CSolution*
value);
133         //const int                                     SetValue(const int, const
CFESolution& value);
134         const int                                       IncreaseOrder() { return 1; };
135         //const CFESolution                             GetValue(const Point&) const;
136         //CSolution*                                    GetValue(const unsigned int);
137         //const CFESolution                             GetValue(const int) const;
138         const double                                    GetMeasure() const { return 0.; };
139         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int)
const;
140     private:
141         static const int                               m_number = 1;
142         Point                                           m_p0;
143         Point                                           m_p1;
144         Point                                           m_normal;
145         double                                          m_mes;
146         void                                            CompNormal();
147         void                                            CompLenght();
148         //double                                          m_w;
149         //CFESolution                                    m_w;
150         //const std::function<const double(const Point&p)> m_psi[2];
151     };

```

```

152
153     class CEdgeMultiBasis : public CShapeFunction<double>
154     {
155     public:
156         CEdgeMultiBasis();
157         CEdgeMultiBasis(const Point&, const Point&);
158         CEdgeMultiBasis(const Point*);
159         CEdgeMultiBasis(const CEdgeMultiBasis&);
160         CEdgeMultiBasis& operator=(const CEdgeMultiBasis& e)
161         {
162             m_p0 = e.m_p0;
163             m_p1 = e.m_p1;
164             m_normal = e.m_normal;
165             m_mes = e.m_mes;
166             //m_w = e.m_w;
167             return *this;
168         }
169         ~CEdgeMultiBasis() {};
170         const int                                     GetNumberOfShapeFunctions() const;
171         //const DForm<0>*                               GetShapeFunction(const int)
172     const;                                           const double                               GetShapeFunction(const int, const
173     Point&) const;                                     const Point                               GetGradShapeFunction(const int,
174     const Point&) const;                                     const Point                               GetNormal() const;
175     void                                             ReverseNormal();
176     const double                                     GetWeight(const int node, const
177     std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
178     {
179         return f(verts[node]);
180     };
181     //const int                                     SetValue(const int, CSolution*
182     value);                                           IncreaseOrder() { return 1; };
183     const int                                     SetValue(const int, const
184     CFESolution& value);
185     //const CFESolution                               GetValue(const Point&) const;
186     //const CFESolution                               GetValue(const int) const;
187     const double                                     GetMeasure() const { return 0.; };
188     //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int)
189     const;
190     private:
191         static const int                             m_number = 2;
192         Point                                         m_p0;
193         Point                                         m_p1;
194         Point                                         m_normal;
195         double                                         m_mes;
196         void                                           CompNormal();
197         void                                           CompLenght();
198         //std::vector<double>                         m_w[m_number];
199         //std::vector<CFESolution>                     m_w;
200         //const std::function<const double(const Point&p)> m_psi[2];
201     };
202
203     class CEdgeHermiteBasis : public CShapeFunction<double>
204     {
205     public:
206         CEdgeHermiteBasis();
207         CEdgeHermiteBasis(const Point&, const Point&);
208         CEdgeHermiteBasis(const Point*);
209         CEdgeHermiteBasis(const CEdgeHermiteBasis&);
210         CEdgeHermiteBasis& operator=(const CEdgeHermiteBasis& e)
211         {
212             m_p0 = e.m_p0;
213             m_p1 = e.m_p1;
214             m_normal = e.m_normal;
215             m_mes = e.m_mes;
216             //m_w = e.m_w;
217             return *this;
218         }
219         ~CEdgeHermiteBasis() {};
220         const int                                     GetNumberOfShapeFunctions() const;
221         //const DForm<0>*                               GetShapeFunction(const int)
222     const;                                           const double                               GetShapeFunction(const int, const
223     Point&) const;                                     const Point                               GetGradShapeFunction(const int,
224     const Point&) const;                                     const int                               IncreaseOrder() { return 1; };
225     void                                             GetNormal() const;
226     const double                                     ReverseNormal();
227     std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
228     {
229         return f(verts[node]);
230     };

```

```

228         //const int                               SetValue(const int, CSolution*
value);
229         //const int                               SetValue(const int, const
CFESolution& value);
230         //const CFESolution                       GetValue(const Point&) const;
231         //const CFESolution                       GetValue(const int) const;
232         const double                             GetMeasure() const { return 0.; };
233         //const std::function<const DForm<0>*(const Point&)>   GetShapeFunction(const int)
const;
234     private:
235         static const int
236         Point
237         Point
238         Point
239         double
240         void
241         void
242         //std::vector<double>
243         //std::vector<CFESolution>
244         //const std::function<const double(const Point&p)>
245     };
246
247     class CEdge2ndBasis : public CShapeFunction<double>
248     {
249     public:
250         CEdge2ndBasis();
251         CEdge2ndBasis(const Point&, const Point&);
252         CEdge2ndBasis(const Point*);
253         CEdge2ndBasis(const CEdge2ndBasis&);
254         CEdge2ndBasis& operator=(const CEdge2ndBasis& e)
255         {
256             m_p0 = e.m_p0;
257             m_p1 = e.m_p1;
258             m_normal = e.m_normal;
259             m_mes = e.m_mes;
260             return *this;
261         }
262         ~CEdge2ndBasis() {};
263         const int
264         //const DForm<0>*
const;
265         const double
Point&) const;
266         const Point
const Point&) const;
267         const Point
const Point
268         void
269         const double
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
270     {
271         return f(verts[node]);
272     };
273         const int
274         //const int
value);
275         //const int
CFESolution& value);
276         //const CFESolution
const double
277         //const CFESolution
//const std::function<const DForm<0>*(const Point&)>
278         //const std::function<const DForm<0>*(const Point&)>
const;
279     private:
280         static const int
281         Point
282         Point
283         Point
284         double
285         void
286         void
287         //std::vector<CFESolution>
288         //const std::function<const double(const Point&p)>
289     };
290     }
291 }
292
293 #endif /* Edge_hpp */

```

7.22 CoreNCFEM/FiniteElements/FiniteElement.h File Reference

```

#include <functional>
#include <iostream>

```



```
#include <vector>
#include "../Point.h"
#include "../FESolution.h"
```

Classes

- class `corenc::Mesh::CElement< bool >`
- class `corenc::Mesh::CElement< T >`
- class `corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >`
- class `corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >`
- class `corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >`

Namespaces

- namespace `corenc`
- namespace `corenc::Mesh`

Typedefs

- using `corenc::Mesh::function_dp` = `std::function< const double(const Point &)>`

Enumerations

- enum `corenc::Mesh::Elements` {
`corenc::Mesh::Interval` = 0 , `corenc::Mesh::Triangle` = 1 , `corenc::Mesh::Rectangle` = 2 , `corenc::Mesh::Tetrahedron`
= 3 ,
`corenc::Mesh::Cube` = 4 }

7.23 FiniteElement.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_MESH_FINITEELEMENT_H_
2 #define CORENC_MESH_FINITEELEMENT_H_
3
4 #include <functional>
5 #include <iostream>
6 #include <vector>
7 #include "../Point.h"
8 #include "../FESolution.h"
9 namespace corenc
10 {
11     namespace Mesh
12     {
13         using function_dp = std::function<const double(const Point&)>;
14         enum Elements
15         {
16             Interval = 0,
17             Triangle = 1,
18             Rectangle = 2,
19             Tetrahedron = 3,
20             Cube = 4
21         };
22
23         template<class T = bool>
24         class CElement;
25
26         template<>
27         class CElement<bool>
```

```

28     {
29     public:
30         CElement() {}
31         virtual ~CElement() {}
32         virtual const int
33         virtual CElement*>*
34         virtual const int
35         virtual const int
36         virtual const int
37         virtual void
38     = 0;
39         virtual void
40         virtual const int
41         //virtual void          SetShapeFunction(const unsigned int, const std::function<const
DiffForm(const Point&)>&) = 0;
42         //virtual const DiffForm* GetShapeFunction(const int, const Point&) const = 0;
43         //virtual const double
const std::vector<double>&) const = 0;
44         virtual const double
const = 0;
45         virtual const Point
Point&) const = 0;
46         //virtual const std::vector<double>
const std::vector<double>&) const = 0;
47         virtual const Point
48         //virtual const std::vector<double>
49         virtual void
50         virtual const double
std::vector<Point>& v) const = 0;
51         //virtual const double
double(const std::vector<double>&)>&, const std::vector<std::vector<double>& v) const = 0;
52         virtual const Point
Point(const Point&)>&, const std::vector<Point>& v) const = 0;
53         //virtual const std::vector<double>
std::vector<double>>(const std::vector<double>&)>&, const std::vector<std::vector<double>& v) const =
0;
54         virtual const std::vector<double>
std::vector<double>>(const Point&)>&, const std::vector<Point>&) const = 0;
55         virtual const double
std::vector<Point>& verts, const function_dp& f) const = 0;
56         //virtual const Type
= 0;
57         //virtual const Type
0;
58         //virtual const int
Type& value) = 0;
59         //virtual const int
value) = 0;
60         //virtual CSolution*
61         virtual const int
62         virtual const double
63         //virtual const std::vector<double>
std::vector<double>>(const std::vector<double>&)>&, const std::vector<std::vector<double>&) const = 0;
64         //virtual std::function<const DiffForm(const Point&)> GetShapeFunction(const int) = 0;
65         //virtual const double GetShapeFunction(const int, const Point&) const = 0;
66     };
67     template<class T>
68     class CElement
69     {
70     public:
71         CElement() {}
72         virtual ~CElement() {}
73         virtual const int
74         virtual CElement*
75         virtual const int
76         virtual const int
77         virtual const int
78         virtual void
79     = 0;
80         virtual void
81         virtual const int
82         //virtual void          SetShapeFunction(const unsigned int, const std::function<const
DiffForm(const Point&)>&) = 0;
83         //virtual const DiffForm* GetShapeFunction(const int, const Point&) const = 0;
84         //virtual const double
const std::vector<double>&) const = 0;
85         virtual const double
const = 0;
86         virtual const Point
Point&) const = 0;
87         //virtual const std::vector<double>
const std::vector<double>&) const = 0;
88         virtual const Point
89         //virtual const std::vector<double>
90         virtual void

```

```

91         virtual const int                                     IncreaseOrder() = 0;
92         virtual const double                                 Integrate(const std::function<const
double(const Point&)>&, const std::vector<Point>& v) const = 0;
93         //virtual const double                                Integrate(const std::function<const
double(const std::vector<double>&)>&, const std::vector<std::vector<double>& v) const = 0;
94         virtual const Point                                  Integrate(const std::function<const
Point(const Point&)>&, const std::vector<Point>& v) const = 0;
95         //virtual const std::vector<double>                  Integrate(const std::function<const
std::vector<double>(const std::vector<double>&)>&, const std::vector<std::vector<double>& v) const =
0;
96         virtual const std::vector<double>                   Integrate(const std::function<const
std::vector<double>(const Point&)>&, const std::vector<Point>& v) const = 0;
97         //virtual const Type                                  GetValue(const unsigned number) const
= 0;
98         //virtual const Type                                  GetValue(const Mesh::Point&) const =
0;
99         //virtual const int                                    SetValue(const unsigned int number, const
Type& value) = 0;
100        //virtual const int                                    SetValue(const unsigned int number,
CSolution* value) = 0;
101        //virtual CSolution*                                   GetValue(const unsigned int) = 0;
102        virtual const double                                  GetMeasure() const = 0;
103        virtual const double                                  GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const = 0;
104        //virtual const T                                       GetValue(const int number) const =
0;
105        //virtual const T                                       GetValue(const Point& p) const = 0;
106        //virtual const int                                    SetValue(const int number, const T&
value) = 0;
107        //virtual const std::vector<double>                   Integrate(const std::function<const
std::vector<double>(const std::vector<double>&)>&, const std::vector<std::vector<double>& v) const = 0;
108        //virtual std::function<const DiffForm(const Point&)> GetShapeFunction(const int) = 0;
109        //virtual const double GetShapeFunction(const int, const Point&) const = 0;
110    };
111
112
113
114
115    // Set of nodes
116    // Set of shape function
117    // Set of degrees of freedom ; don't use pls
118    // Type of the weights aligned with the degrees of freedom
119    // The weights should be inside of the set of shape functions and the types should be same
120    template<class Shape, class ShapeFunction, class DoF = bool, class T = bool>
121    class CFiniteElement;
122    template<class Shape, class ShapeFunction, class DoF, class T>
123    class CFiniteElement: public CElement<T>
124    {
125    public:
126        CFiniteElement() {}
127        CFiniteElement(const int* nodes, const Point* points, const int dofs) :
128            m_shape{ nodes },
129            m_shapefunctions{ points },
130            m_dofs{ dofs },
131            m_type{ -1 } {
132            m_neighbours[0] = -1; m_neighbours[1] = -1;
133        };
134        CFiniteElement(const int* nodes, const Point* points) :
135            m_shape{ nodes },
136            m_shapefunctions{ points },
137            m_dofs{ 0 },
138            m_type{ -1 } {
139            m_neighbours[0] = -1; m_neighbours[1] = -1;
140        };
141        CFiniteElement(const Shape& shape, const ShapeFunction& f, const DoF& d) :
142            m_shape{ shape },
143            m_shapefunctions{ f },
144            m_dofs{ d },
145            m_type{ -1 } {
146            m_neighbours[0] = -1; m_neighbours[1] = -1;
147        };
148        CFiniteElement(const Shape& shape, const ShapeFunction& shfunc, const DoF& dofs, const int
type) :
149            m_shape{ shape },
150            m_shapefunctions{ shfunc },
151            m_dofs{ dofs },
152            m_type{ type } {
153            m_neighbours[0] = -1; m_neighbours[1] = -1;
154        };
155        CFiniteElement(const Shape& shape, const ShapeFunction& shfunc, const DoF& dofs,
const int type, const int* neigs) :
156            m_shape{ shape },
157            m_shapefunctions{ shfunc },
158            m_dofs{ dofs },
159            m_type{ type } {
160            m_neighbours[0] = neigs[0]; m_neighbours[1] = neigs[1];
161        };

```

```

162         CFiniteElement(const CFiniteElement<Shape, ShapeFunction, DoF>& e) :
163             m_shape{ e.m_shape },
164             m_shapefunctions{ e.m_shapefunctions },
165             m_dofs{ e.m_dofs },
166             m_type{ e.m_type } {
167                 m_neighbours[0] = e.m_neighbours[0]; m_neighbours[1] = e.m_neighbours[1];
168             };
169         CElement<T>*          Clone() const
170         {
171             return new CFiniteElement<Shape, ShapeFunction, DoF, T>(m_shape,
m_shapefunctions, m_dofs, m_type, m_neighbours);
172         };
173         friend const bool    operator==(const CFiniteElement& e1, const CFiniteElement& e2)
174         {
175             if (e1.m_shape == e2.m_shape)
176                 return true;
177             return false;
178         }
179         ~CFiniteElement() {}
180         const int             GetType() const;
181         const int             GetNode(const int) const;
182         const int             GetNeighbour(const int) const;
183         const Shape           GetShape() const;
184         const ShapeFunction   GetShapeFunctions() const;
185         const DoF             GetDoF() const;
186         const int             GetDofs() const;
187         void                  SetNeighbour(const int k, const int elem);
188         void                  SetType(const int);
189         void                  SetShapeFunction(const int, const ShapeFunction&);
190         void                  SetDoF(const DoF&);
191         void                  SetShape(const Shape&);
192         const int             IncreaseOrder();
193         void                  SetNode(const int, const int);
194         const int             GetNumberOfNodes() const;
195         //const int           SetValue(const int number, CSolution* value);
196         const double          GetMeasure() const;
197         //CSolution*          GetValue(const int);
198         //void                SetShapeFunction(const int, const std::function<const
DiffForm(const Point&)>&);
199         const double          Integrate(const std::function<const double(const Point&)>&, const
std::vector<Point>& v) const;
200         const Point           Integrate(const std::function<const Point(const Point&)>&, const
std::vector<Point>& v) const;
201         const std::vector<double> Integrate(const std::function<const std::vector<double>(const
Point&)>&, const std::vector<Point>&) const;
202         //const std::function<const DiffForm(const Point&)> GetShapeFunction(const int)
const;
203         //const DiffForm*      GetShapeFunction(const int, const Point&);
204         const double          GetShapeFunction(const int, const Point&) const;
205         const Point           GetGradShapeFunction(const int, const Point&) const;
206         const Point           GetNormal() const;
207         void                  ReverseNormal();
208         const double          GetWeight(const int, const std::vector<Point>& verts, const
std::function<const double(const Point&)>& f) const;
209         //const T              GetValue(const int number) const;
210         //const T              GetValue(const Point& p) const;
211         //const int            SetValue(const int number, const T& value);
212         CFiniteElement&       operator=(const CFiniteElement& e)
213         {
214             m_shape = e.m_shape;
215             m_shapefunctions = e.m_shapefunctions;
216             m_dofs = e.m_dofs;
217             m_type = e.m_type;
218             return *this;
219         }
220         friend std::istream&    operator>(std::istream& is, CFiniteElement& k)
221         {
222             is >> k.m_shape;
223             return is;
224         }
225         //const DiffForm        GetDShapeFunction(const int, const Point&);
226     private:
227         Shape                  m_shape;
228         ShapeFunction          m_shapefunctions;
229         DoF                    m_dofs;
230         int                    m_type;
231         int                    m_neighbours[2];
232     };
233
234     template<class Shape, class ShapeFunction, class DoF>
235     class CFiniteElement<Shape, ShapeFunction, DoF, bool> : public CElement<>
236     {
237     public:
238         CFiniteElement() {}
239         CFiniteElement(const int* nodes, const Point* points, const int dofs) :
240             m_shape{ nodes },
241             m_shapefunctions{ points },

```

```

242         m_dofs{ dofs },
243         m_type{ -1 } {
244             m_neighbours[0] = -1; m_neighbours[1] = -1;
245         };
246         CFiniteElement(const int* nodes, const Point* points) :
247             m_shape{ nodes },
248             m_shapefunctions{ points },
249             m_dofs{ 0 },
250             m_type{ -1 } {
251                 m_neighbours[0] = -1; m_neighbours[1] = -1;
252             };
253         CFiniteElement(const Shape& shape, const ShapeFunction& f, const DoF& d) :
254             m_shape{ shape },
255             m_shapefunctions{ f },
256             m_dofs{ d },
257             m_type{ -1 } {
258                 m_neighbours[0] = -1; m_neighbours[1] = -1;
259             };
260         CFiniteElement(const Shape& shape, const ShapeFunction& shfunc, const DoF& dofs, const int
type) :
261             m_shape{ shape },
262             m_shapefunctions{ shfunc },
263             m_dofs{ dofs },
264             m_type{ type } {
265                 m_neighbours[0] = -1; m_neighbours[1] = -1;
266             };
267         CFiniteElement(const Shape& shape, const ShapeFunction& shfunc, const DoF& dofs, const int
type, const int* neigh) :
268             m_shape{ shape },
269             m_shapefunctions{ shfunc },
270             m_dofs{ dofs },
271             m_type{ type } {
272                 m_neighbours[0] = neigh[0]; m_neighbours[1] = neigh[1];
273             };
274         CFiniteElement(const CFiniteElement<Shape, ShapeFunction, DoF>& e) :
275             m_shape{ e.m_shape },
276             m_shapefunctions{ e.m_shapefunctions },
277             m_dofs{ e.m_dofs },
278             m_type{ e.m_type } {
279                 m_neighbours[0] = e.m_neighbours[0]; m_neighbours[1] = e.m_neighbours[1];
280             };
281         friend const bool      operator==(const CFiniteElement& e1, const CFiniteElement& e2)
282         {
283             if (e1.m_shape == e2.m_shape)
284                 return true;
285             return false;
286         }
287         // don't forget to delete after the call
288         CElement<>*      Clone() const
289         {
290             return new CFiniteElement<Shape, ShapeFunction, DoF>(m_shape, m_shapefunctions, m_dofs,
m_type, m_neighbours);
291         };
292         ~CFiniteElement() {}
293         const int          GetType() const;
294         const int          GetNode(const int) const;
295         const int          GetNeighbour(const int) const;
296         const Shape        GetShape() const;
297         const ShapeFunction GetShapeFunctions() const;
298         const DoF          GetDoF() const;
299         const int          GetDoFs() const;
300         void               SetNeighbour(const int k, const int elem);
301         void               SetType(const int);
302         void               SetShapeFunction(const int, const ShapeFunction&);
303         void               SetDoF(const DoF&);
304         void               SetShape(const Shape&);
305         void               SetNode(const int, const int);
306         const int          GetNumberOfNodes() const;
307         //const int         SetValue(const int number, CSolution* value);
308         const int          IncreaseOrder();
309         const double        GetMeasure() const;
310         //CSolution*        GetValue(const int);
311         //void              SetShapeFunction(const int, const std::function<const
DiffForm(const Point&)>&);
312         const double        Integrate(const std::function<const double(const Point&)>&, const
std::vector<Point>& v) const;
313         const Point         Integrate(const std::function<const Point(const Point&)>&, const
std::vector<Point>& v) const;
314         const std::vector<double> Integrate(const std::function<const std::vector<double>(const
Point&)>&, const std::vector<Point>&) const;
315         //const std::function<const DiffForm(const Point&)>      GetShapeFunction(const int)
const;
316         //const DiffForm*   GetShapeFunction(const int, const Point&);
317         const double        GetShapeFunction(const int, const Point&) const;
318         const Point         GetGradShapeFunction(const int, const Point&) const;
319         const Point         GetNormal() const;
320         void               ReverseNormal();

```

```

321         const double          GetWeight(const int, const std::vector<Point>& verts, const
std::function<const double(const Point&)>& f) const;
322         CFiniteElement&      operator=(const CFiniteElement& e)
323         {
324             m_shape = e.m_shape;
325             m_shapefunctions = e.m_shapefunctions;
326             m_dofs = e.m_dofs;
327             m_type = e.m_type;
328             return *this;
329         }
330     friend std::istream&      operator>(std::istream& is, CFiniteElement& k)
331     {
332         is >> k.m_shape;
333         return is;
334     }
335     //const DiffForm          GetDShapeFunction(const int, const Point&);
336 private:
337     Shape                    m_shape;
338     ShapeFunction            m_shapefunctions;
339     DoF                      m_dofs;
340     int                     m_type;
341     int                     m_neighbours[2];
342 };
343
344
345 template<class Shape, class ShapeFunction>
346 class CFiniteElement<Shape, ShapeFunction, bool, bool> : public CElement<>
347 {
348 public:
349     CFiniteElement() {}
350     CFiniteElement(const int* nodes, const Point* points, const int dofs) :
351         m_shape{ nodes },
352         m_shapefunctions{ points, dofs },
353         m_type{ -1 } {
354         m_neighbours[0] = -1; m_neighbours[1] = -1;
355     };
356     CFiniteElement(const int* nodes, const Point* points, const int dofs, const int type) :
357         m_shape{ nodes, dofs },
358         m_shapefunctions{ points, dofs },
359         m_type{ type } {
360         m_neighbours[0] = -1; m_neighbours[1] = -1;
361     };
362     CFiniteElement(const int* nodes, const Point* points) :
363         m_shape{ nodes },
364         m_shapefunctions{ points },
365         m_type{ -1 } {
366         m_neighbours[0] = -1; m_neighbours[1] = -1;
367     };
368     CFiniteElement(const Shape& shape, const ShapeFunction& f) :
369         m_shape{ shape },
370         m_shapefunctions{ f },
371         m_type{ -1 } {
372         m_neighbours[0] = -1; m_neighbours[1] = -1;
373     };
374     CFiniteElement(const Shape& shape, const ShapeFunction& shfunc, const int type) :
375         m_shape{ shape },
376         m_shapefunctions{ shfunc },
377         m_type{ type } {
378         m_neighbours[0] = -1; m_neighbours[1] = -1;
379     };
380     CFiniteElement(const Shape& shape, const ShapeFunction& shfunc, const int type,
const int* neigs) :
381         m_shape{ shape },
382         m_shapefunctions{ shfunc },
383         m_type{ type } {
384         m_neighbours[0] = neigs[0]; m_neighbours[1] = neigs[1];
385     };
386     CFiniteElement(const CFiniteElement<Shape, ShapeFunction>&e) :
387         m_shape{ e.m_shape },
388         m_shapefunctions{ e.m_shapefunctions },
389         m_type{ e.m_type } {
390         m_neighbours[0] = e.m_neighbours[0]; m_neighbours[1] = e.m_neighbours[1];
391     };
392     friend const bool      operator==(const CFiniteElement& e1, const CFiniteElement& e2)
393     {
394         if (e1.m_shape == e2.m_shape)
395             return true;
396         return false;
397     }
398     // don't forget to delete after the call
399     CElement<>*            Clone() const
400     {
401         return new CFiniteElement<Shape, ShapeFunction>(m_shape,
m_shapefunctions, m_type, m_neighbours);
402     };
403     ~CFiniteElement() {}
404     const int              GetType() const;

```

```

405         const int             GetNode(const int) const;
406         const int             GetNeighbour(const int) const;
407         const Shape            GetShape() const;
408         const ShapeFunction    GetShapeFunctions() const;
409         const int             GetDoFs() const;
410         void                   SetNeighbour(const int k, const int elem);
411         void                   SetType(const int);
412         void                   SetShapeFunction(const int, const ShapeFunction&);
413         void                   SetShape(const Shape&);
414         void                   SetNode(const int, const int);
415         const int             GetNumberOfNodes() const;
416         //const int            SetValue(const int number, CSolution* value);
417         const int             IncreaseOrder();
418         const double           GetMeasure() const;
419         //CSolution*           GetValue(const int);
420         //void                  SetShapeFunction(const int, const std::function<const
DiffForm(const Point&)>&);
421         const double           Integrate(const std::function<const double(const Point&)>&, const
std::vector<Point>& v) const;
422         const Point            Integrate(const std::function<const Point(const Point&)>&, const
std::vector<Point>& v) const;
423         const std::vector<double> Integrate(const std::function<const std::vector<double>(const
Point&)>&, const std::vector<Point>&) const;
424         //const std::function<const DiffForm(const Point&)>           GetShapeFunction(const int)
const;
425         //const DiffForm*      GetShapeFunction(const int, const Point&);
426         const double           GetShapeFunction(const int, const Point&) const;
427         const Point            GetGradShapeFunction(const int, const Point&) const;
428         const Point            GetNormal() const;
429         void                   ReverseNormal();
430         const double           GetWeight(const int, const std::vector<Point>& verts, const
std::function<const double(const Point&)>& f) const;
431         CFiniteElement&        operator=(const CFiniteElement& e)
432         {
433             m_shape = e.m_shape;
434             m_shapefunctions = e.m_shapefunctions;
435             m_type = e.m_type;
436             return *this;
437         }
438         friend std::istream&    operator>(std::istream& is, CFiniteElement& k)
439         {
440             is >> k.m_shape;
441             return is;
442         }
443         //const DiffForm        GetDShapeFunction(const int, const Point&);
444     private:
445         Shape                  m_shape;
446         ShapeFunction           m_shapefunctions;
447         int                    m_type;
448         int                    m_neighbours[2];
449     };
450
451
452
453
454
455     // implementation template<class Shape, class ShapeFunction, class DoF>
456     // CFiniteElement<Shape, ShapeFunction, DoF>
457     template<class Shape, class ShapeFunction, class DoF>
458     const int CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetType() const
459     {
460         return m_type;
461     }
462     //template<class Shape, class ShapeFunction, class DoF>
463     //const int CFiniteElement<Shape, ShapeFunction, DoF, bool>::SetValue(const int number,
CSolution* value)
464     //{
465     //    return m_shapefunctions.SetValue(number, value);
466     //}
467     //template<class Shape, class ShapeFunction, class DoF>
468     //CSolution* CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetValue(const int p)
469     //{
470     //    auto&& val = m_shapefunctions.GetValue(p);
471     //    return const_cast<CSolution*>(static_cast<const CSolution*>(&val));
472     //}
473     template<class Shape, class ShapeFunction, class DoF>
474     const int CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetNode(const int k) const
475     {
476         return m_shape.GetNode(k);
477     }
478     template<class Shape, class ShapeFunction, class DoF>
479     const double CFiniteElement<Shape, ShapeFunction, DoF, bool>::Integrate(const
std::function<const double(const Point&)>&f, const std::vector<Point>& v) const
480     {
481         return m_shape.Integrate(f, v);
482     }
483

```

```

484     template<class Shape, class ShapeFunction, class DoF>
485     const Point CFiniteElement<Shape, ShapeFunction, DoF, bool>::Integrate(const std::function<const
Point(const Point&)>&f, const std::vector<Point>& v) const
486     {
487         return m_shape.Integrate(f, v);
488     }
489
490     template<class Shape, class ShapeFunction, class DoF>
491     const std::vector<double> CFiniteElement<Shape, ShapeFunction, DoF, bool>::Integrate(const
std::function<const std::vector<double>(const Point&)>&f, const std::vector<Point>& v) const
492     {
493         return m_shape.Integrate(f, v);
494     }
495     template<class Shape, class ShapeFunction, class DoF>
496     const int CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetNeighbour(const int k) const
497     {
498         return m_neighbours[k];
499     }
500
501     template<class Shape, class ShapeFunction, class DoF>
502     const Point CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetNormal() const
503     {
504         return m_shapefunctions.GetNormal();
505     }
506
507     template<class Shape, class ShapeFunction, class DoF>
508     void CFiniteElement<Shape, ShapeFunction, DoF, bool>::ReverseNormal()
509     {
510         m_shapefunctions.ReverseNormal();
511     }
512
513     template<class Shape, class ShapeFunction, class DoF>
514     const double CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetMeasure() const
515     {
516         return m_shapefunctions.GetMeasure();
517     }
518
519     //template<class Shape, class ShapeFunction, class DoF, class T>
520     //inline const T CFiniteElement<Shape, ShapeFunction, DoF, T>::GetValue(const int number) const
521     //{
522     //    return m_shapefunctions.GetValue(number);
523     //}
524
525     //template<class Shape, class ShapeFunction, class DoF, class T>
526     //inline const T CFiniteElement<Shape, ShapeFunction, DoF, T>::GetValue(const Point & p) const
527     //{
528     //    return m_shapefunctions.GetValue(p);
529     //}
530
531     //template<class Shape, class ShapeFunction, class DoF, class T>
532     //inline const int CFiniteElement<Shape, ShapeFunction, DoF, T>::SetValue(const int number,
const T & value)
533     //{
534     //    return m_shapefunctions.SetValue(number, value);
535     //}
536
537     template<class Shape, class ShapeFunction, class DoF>
538     const Shape CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetShape() const
539     {
540         return m_shape;
541     }
542
543     template<class Shape, class ShapeFunction, class DoF>
544     const ShapeFunction CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetShapeFunctions() const
545     {
546         return m_shapefunctions;
547     }
548
549     template<class Shape, class ShapeFunction, class DoF>
550     const DoF CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetDoF() const
551     {
552         return m_dofs;
553     }
554
555     template<class Shape, class ShapeFunction, class DoF>
556     const int CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetNumberOfNodes() const
557     {
558         return m_shape.GetNumberOfNodes();
559     }
560
561     template<class Shape, class ShapeFunction, class DoF>
562     const int CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetDoFs() const
563     {
564         return m_shapefunctions.GetNumberOfShapeFunctions();
565     }
566
567     template<class Shape, class ShapeFunction, class DoF>

```



```

568     void CFiniteElement<Shape, ShapeFunction, DoF, bool>::SetNeighbour(const int k, const int elem)
569     {
570         m_neighbours[k] = elem;
571     }
572
573     template<class Shape, class ShapeFunction, class DoF>
574     void CFiniteElement<Shape, ShapeFunction, DoF, bool>::SetShapeFunction(const int k, const
ShapeFunction& func)
575     {
576         m_shapefunctions = func;
577     }
578
579     template<class Shape, class ShapeFunction, class DoF>
580     void CFiniteElement<Shape, ShapeFunction, DoF, bool>::SetDoF(const DoF& dof)
581     {
582         m_dofs = dof;
583     }
584
585     template<class Shape, class ShapeFunction, class DoF>
586     void CFiniteElement<Shape, ShapeFunction, DoF, bool>::SetShape(const Shape &shape)
587     {
588         m_shape = shape;
589     }
590
591     template<class Shape, class ShapeFunction, class DoF>
592     void CFiniteElement<Shape, ShapeFunction, DoF, bool>::SetType(const int k)
593     {
594         m_type = k;
595     }
596
597     template<class Shape, class ShapeFunction, class DoF>
598     void CFiniteElement<Shape, ShapeFunction, DoF, bool>::SetNode(const int k, const int node)
599     {
600         m_shape.SetNode(k, node);
601     }
602     template<class Shape, class ShapeFunction, class DoF>
603     const double CFiniteElement<Shape, ShapeFunction, DoF>::GetShapeFunction(const int k, const
Mesh::Point &p) const
604     {
605         return m_shapefunctions.GetShapeFunction(k, p);
606     }
607
608     template<class Shape, class ShapeFunction, class DoF>
609     const Point CFiniteElement<Shape, ShapeFunction, DoF, bool>::GetGradShapeFunction(const int k,
const Mesh::Point &p) const
610     {
611         return m_shapefunctions.GetGradShapeFunction(k, p);
612     }
613
614     template<class Shape, class ShapeFunction, class DoF>
615     const int CFiniteElement<Shape, ShapeFunction, DoF, bool>::IncreaseOrder()
616     {
617         if (m_shape.IncreaseOrder())
618             return 1;
619         if (m_shapefunctions.IncreaseOrder())
620             return 1;
621         return 0;
622     }
623     template<class Shape, class ShapeFunction, class DoF>
624     const double CFiniteElement<Shape, ShapeFunction, DoF>::GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
625     {
626         return 0.0;
627     }
628     // fin.
629
630     // implementation template<class Shape, class ShapeFunction, class DoF, class T>
631     // CFiniteElement<Shape, ShapeFunction, DoF, T>
632     template<class Shape, class ShapeFunction, class DoF, class T>
633     const int CFiniteElement<Shape, ShapeFunction, DoF, T>::GetType() const
634     {
635         return m_type;
636     }
637     template<class Shape, class ShapeFunction, class DoF, class T>
638     const double CFiniteElement<Shape, ShapeFunction, DoF, T>::GetMeasure() const
639     {
640         return m_shapefunctions.GetMeasure();
641     }
642     //template<class Shape, class ShapeFunction, class DoF, class T>
643     //const int CFiniteElement<Shape, ShapeFunction, DoF, T>::SetValue(const int number, CSolution*
value)
644     //{
645     //    return m_shapefunctions.SetValue(number, value);
646     //}
647     //template<class Shape, class ShapeFunction, class DoF, class T>
648     //CSolution* CFiniteElement<Shape, ShapeFunction, DoF, T>::GetValue(const int p)
649     //{

```

```

650         // auto&& val = m_shapefunctions.GetValue(p);
651         // return const_cast<CSolution*>(static_cast<const CSolution*>(&val));
652         //}
653         template<class Shape, class ShapeFunction, class DoF, class T>
654         const int CFiniteElement<Shape, ShapeFunction, DoF, T>::GetNode(const int k) const
655         {
656             return m_shape.GetNode(k);
657         }
658         template<class Shape, class ShapeFunction, class DoF, class T>
659         const double CFiniteElement<Shape, ShapeFunction, DoF, T>::Integrate(const std::function<const
double(const Point&)>&f, const std::vector<Point>& v) const
660         {
661             return m_shape.Integrate(f, v);
662         }
663
664         template<class Shape, class ShapeFunction, class DoF, class T>
665         const Point CFiniteElement<Shape, ShapeFunction, DoF, T>::Integrate(const std::function<const
Point(const Point&)>&f, const std::vector<Point>& v) const
666         {
667             return m_shape.Integrate(f, v);
668         }
669
670         template<class Shape, class ShapeFunction, class DoF, class T>
671         const std::vector<double> CFiniteElement<Shape, ShapeFunction, DoF, T>::Integrate(const
std::function<const std::vector<double>(const Point&)>&f, const std::vector<Point>& v) const
672         {
673             return m_shape.Integrate(f, v);
674         }
675         template<class Shape, class ShapeFunction, class DoF, class T>
676         const int CFiniteElement<Shape, ShapeFunction, DoF, T>::GetNeighbour(const int k) const
677         {
678             return m_neighbours[k];
679         }
680
681         template<class Shape, class ShapeFunction, class DoF, class T>
682         const Point CFiniteElement<Shape, ShapeFunction, DoF, T>::GetNormal() const
683         {
684             return m_shapefunctions.GetNormal();
685         }
686
687         template<class Shape, class ShapeFunction, class DoF, class T>
688         void CFiniteElement<Shape, ShapeFunction, DoF, T>::ReverseNormal()
689         {
690             m_shapefunctions.ReverseNormal();
691         }
692
693         template<class Shape, class ShapeFunction, class DoF, class T>
694         const double CFiniteElement<Shape, ShapeFunction, DoF, T>::GetWeight(const int node,
const std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
695         {
696             return m_shapefunctions.GetWeight(node, verts, f);
697         }
698
699
700         template<class Shape, class ShapeFunction, class DoF, class T>
701         const Shape CFiniteElement<Shape, ShapeFunction, DoF, T>::GetShape() const
702         {
703             return m_shape;
704         }
705
706         template<class Shape, class ShapeFunction, class DoF, class T>
707         const ShapeFunction CFiniteElement<Shape, ShapeFunction, DoF, T>::GetShapeFunctions() const
708         {
709             return m_shapefunctions;
710         }
711
712         template<class Shape, class ShapeFunction, class DoF, class T>
713         const DoF CFiniteElement<Shape, ShapeFunction, DoF, T>::GetDoF() const
714         {
715             return m_dofs;
716         }
717
718         template<class Shape, class ShapeFunction, class DoF, class T>
719         const int CFiniteElement<Shape, ShapeFunction, DoF, T>::GetNumberOfNodes() const
720         {
721             return m_shape.GetNumberOfNodes();
722         }
723
724         template<class Shape, class ShapeFunction, class DoF, class T>
725         const int CFiniteElement<Shape, ShapeFunction, DoF, T>::GetDoFs() const
726         {
727             return m_shapefunctions.GetNumberOfShapeFunctions();
728         }
729
730         template<class Shape, class ShapeFunction, class DoF, class T>
731         void CFiniteElement<Shape, ShapeFunction, DoF, T>::SetNeighbour(const int k, const int elem)
732         {

```

```

733         m_neighbours[k] = elem;
734     }
735
736     template<class Shape, class ShapeFunction, class DoF, class T>
737     void CFiniteElement<Shape, ShapeFunction, DoF, T>::SetShapeFunction(const int k, const
ShapeFunction& func)
738     {
739         m_shapefunctions = func;
740     }
741
742     template<class Shape, class ShapeFunction, class DoF, class T>
743     void CFiniteElement<Shape, ShapeFunction, DoF, T>::SetDoF(const DoF& dof)
744     {
745         m_dofs = dof;
746     }
747
748     template<class Shape, class ShapeFunction, class DoF, class T>
749     void CFiniteElement<Shape, ShapeFunction, DoF, T>::SetShape(const Shape &shape)
750     {
751         m_shape = shape;
752     }
753
754     template<class Shape, class ShapeFunction, class DoF, class T>
755     void CFiniteElement<Shape, ShapeFunction, DoF, T>::SetType(const int k)
756     {
757         m_type = k;
758     }
759
760     template<class Shape, class ShapeFunction, class DoF, class T>
761     void CFiniteElement<Shape, ShapeFunction, DoF, T>::SetNode(const int k, const int node)
762     {
763         m_shape.SetNode(k, node);
764     }
765     template<class Shape, class ShapeFunction, class DoF, class T>
766     const double CFiniteElement<Shape, ShapeFunction, DoF, T>::GetShapeFunction(const int k, const
Mesh::Point &p) const
767     {
768         return m_shapefunctions.GetShapeFunction(k, p);
769     }
770
771     template<class Shape, class ShapeFunction, class DoF, class T>
772     const Point CFiniteElement<Shape, ShapeFunction, DoF, T>::GetGradShapeFunction(const int k,
const Mesh::Point &p) const
773     {
774         return m_shapefunctions.GetGradShapeFunction(k, p);
775     }
776     template<class Shape, class ShapeFunction, class DoF, class T>
777     const int CFiniteElement<Shape, ShapeFunction, DoF, T>::IncreaseOrder()
778     {
779         if (m_shape.IncreaseOrder())
780             return 1;
781         if (m_shapefunctions.IncreaseOrder())
782             return 1;
783         return 0;
784     }
785
786     // implementation template<class Shape, class ShapeFunction>
787     // CFiniteElement<Shape, ShapeFunction>
788
789     template<class Shape, class ShapeFunction>
790     const int CFiniteElement<Shape, ShapeFunction, bool, bool>::GetType() const
791     {
792         return m_type;
793     }
794     template<class Shape, class ShapeFunction>
795     const double CFiniteElement<Shape, ShapeFunction, bool, bool>::GetMeasure() const
796     {
797         return m_shapefunctions.GetMeasure();
798     }
799     //template<class Shape, class ShapeFunction>
800     //const int CFiniteElement<Shape, ShapeFunction, bool, bool>::SetValue(const int number,
CSolution* value)
801     //{
802     //    return m_shapefunctions.SetValue(number, value);
803     //}
804     //template<class Shape, class ShapeFunction>
805     //CSolution* CFiniteElement<Shape, ShapeFunction, bool, bool>::GetValue(const int p)
806     //{
807     //    auto&& val = m_shapefunctions.GetValue(p);
808     //    return const_cast<CSolution*>(static_cast<const CSolution*>(&val));
809     //}
810     template<class Shape, class ShapeFunction>
811     const int CFiniteElement<Shape, ShapeFunction, bool, bool>::GetNode(const int k) const
812     {
813         return m_shape.GetNode(k);
814     }
815     template<class Shape, class ShapeFunction>

```

```

816         const double CFiniteElement<Shape, ShapeFunction, bool, bool>::Integrate(const
std::function<const double(const Point&)>&f, const std::vector<Point>& v) const
817     {
818         return m_shape.Integrate(f, v);
819     }
820
821     template<class Shape, class ShapeFunction>
822     const Point CFiniteElement<Shape, ShapeFunction, bool, bool>::Integrate(const
std::function<const Point(const Point&)>&f, const std::vector<Point>& v) const
823     {
824         return m_shape.Integrate(f, v);
825     }
826
827     template<class Shape, class ShapeFunction>
828     const std::vector<double> CFiniteElement<Shape, ShapeFunction, bool, bool>::Integrate(const
std::function<const std::vector<double>(const Point&)>&f, const std::vector<Point>& v) const
829     {
830         return m_shape.Integrate(f, v);
831     }
832     template<class Shape, class ShapeFunction>
833     const int CFiniteElement<Shape, ShapeFunction, bool, bool>::GetNeighbour(const int k) const
834     {
835         return m_neighbours[k];
836     }
837
838     template<class Shape, class ShapeFunction>
839     const Point CFiniteElement<Shape, ShapeFunction, bool, bool>::GetNormal() const
840     {
841         return m_shapefunctions.GetNormal();
842     }
843
844     template<class Shape, class ShapeFunction>
845     void CFiniteElement<Shape, ShapeFunction, bool, bool>::ReverseNormal()
846     {
847         m_shapefunctions.ReverseNormal();
848     }
849
850
851     template<class Shape, class ShapeFunction>
852     const Shape CFiniteElement<Shape, ShapeFunction, bool, bool>::GetShape() const
853     {
854         return m_shape;
855     }
856
857     template<class Shape, class ShapeFunction>
858     const ShapeFunction CFiniteElement<Shape, ShapeFunction, bool, bool>::GetShapeFunctions() const
859     {
860         return m_shapefunctions;
861     }
862
863
864     template<class Shape, class ShapeFunction>
865     const int CFiniteElement<Shape, ShapeFunction, bool, bool>::GetNumberOfNodes() const
866     {
867         return m_shape.GetNumberOfNodes();
868     }
869
870     template<class Shape, class ShapeFunction>
871     const int CFiniteElement<Shape, ShapeFunction, bool, bool>::GetDoFs() const
872     {
873         return m_shapefunctions.GetNumberOfShapeFunctions();
874     }
875
876     template<class Shape, class ShapeFunction>
877     void CFiniteElement<Shape, ShapeFunction, bool, bool>::SetNeighbour(const int k, const int elem)
878     {
879         m_neighbours[k] = elem;
880     }
881
882     template<class Shape, class ShapeFunction>
883     void CFiniteElement<Shape, ShapeFunction, bool, bool>::SetShapeFunction(const int k, const
ShapeFunction& func)
884     {
885         m_shapefunctions = func;
886     }
887
888     template<class Shape, class ShapeFunction>
889     void CFiniteElement<Shape, ShapeFunction, bool, bool>::SetShape(const Shape &shape)
890     {
891         m_shape = shape;
892     }
893
894     template<class Shape, class ShapeFunction>
895     void CFiniteElement<Shape, ShapeFunction, bool, bool>::SetType(const int k)
896     {
897         m_type = k;
898     }

```

```

899
900     template<class Shape, class ShapeFunction>
901     void CFiniteElement<Shape, ShapeFunction, bool, bool>::SetNode(const int k, const int node)
902     {
903         m_shape.SetNode(k, node);
904     }
905     template<class Shape, class ShapeFunction>
906     const double CFiniteElement<Shape, ShapeFunction, bool, bool>::GetShapeFunction(const int k,
907     const Mesh::Point &p) const
908     {
909         return m_shapefunctions.GetShapeFunction(k, p);
910     }
911     template<class Shape, class ShapeFunction>
912     const Point CFiniteElement<Shape, ShapeFunction, bool, bool>::GetGradShapeFunction(const int k,
913     const Mesh::Point &p) const
914     {
915         return m_shapefunctions.GetGradShapeFunction(k, p);
916     }
917     template<class Shape, class ShapeFunction>
918     const int CFiniteElement<Shape, ShapeFunction, bool, bool>::IncreaseOrder()
919     {
920         if(m_shape.IncreaseOrder())
921             return 1;
922         if (m_shapefunctions.IncreaseOrder())
923             return 1;
924         return 0;
925     }
926     template<class Shape, class ShapeFunction>
927     const double CFiniteElement<Shape, ShapeFunction, bool, bool>::GetWeight(const int node, const
928     std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
929     {
930         return m_shapefunctions.GetWeight(node, verts, f);
931     }
932     // fin.
933 }
934
935 }
936
937
938 #endif /* CORENC_MESH_FINITEELEMENT_H_ */

```

7.24 CoreNCFEM/FiniteElements/FiniteElement2D.h File Reference

```

#include <functional>
#include <iostream>
#include <vector>
#include "../Point.h"
#include "../FESolution.h"

```

Classes

- class [corenc::Mesh::CElement2D< bool >](#)
- class [corenc::Mesh::CElement2D< T >](#)
- class [corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.25 FiniteElement2D.h

[Go to the documentation of this file.](#)

```

1 #ifndef FINITEELEMENT2D_H
2 #define FINITEELEMENT2D_H
3 #include <functional>
4 #include <iostream>
5 #include <vector>
6 #include "../Point.h"
7 #include "../FESolution.h"
8 namespace corenc
9 {
10     namespace Mesh
11     {
12         // 2d element
13
14         template<class T = bool>
15         class CElement2D;
16         using function_dp = std::function<const double(const Point&);>;
17         template<>
18         class CElement2D<bool>
19         {
20         public:
21             CElement2D() {}
22             virtual ~CElement2D() {}
23             virtual const int
24             virtual CElement2D<>*>
25             virtual const int
26             virtual const int
27             virtual const int
28             virtual void
29             = 0;
30             virtual void
31             virtual void
32             virtual const int
33             virtual const double
34             const = 0;
35             virtual const Point
36             Point&) const = 0;
37             virtual const Point
38             virtual void
39             virtual const int
40             virtual const double
41             std::vector<Point>& v) const = 0;
42             virtual const Point
43             Point(const Point&)>&, const std::vector<Point>& v) const = 0;
44             virtual const std::vector<double>
45             std::vector<double>(const Point&)>&, const std::vector<Point>&) const = 0;
46             virtual const double
47             std::vector<Point>& verts, const function_dp& f) const = 0;
48             virtual const int
49             virtual const double
50             };
51
52         template<class T>
53         class CElement2D
54         {
55         public:
56             CElement2D() {}
57             virtual ~CElement2D() {}
58             virtual const int
59             virtual CElement2D*>
60             virtual const int
61             virtual const int
62             virtual const int
63             virtual void
64             = 0;
65             virtual void
66             virtual void
67             virtual const int
68             virtual const double
69             const = 0;
70             virtual const Point
71             Point&) const = 0;
72             virtual const Point
73             virtual void
74             virtual const int
75             virtual const int
76             virtual const double
77             double(const Point&)>&, const std::vector<Point>& v) const = 0;
78             virtual const Point
79             Point(const Point&)>&, const std::vector<Point>& v) const = 0;
80             virtual const std::vector<double>
81             std::vector<double>(const Point&)>&, const std::vector<Point>&) const = 0;
82             virtual const double
83             GetMeasure() const = 0;
84
85             GetType() const = 0;
86             Clone() const = 0;
87             GetDoFs() const = 0;
88             GetNode(const int) const = 0;
89             GetNeighbour(const int) const = 0;
90             SetNeighbour(const int k, const int elem)
91
92             SetType(const int) = 0;
93             SetNode(const int, const int) = 0;
94             GetNumberOfNodes() const = 0;
95             GetShapeFunction(const int, const Point&)
96
97             GetGradShapeFunction(const int, const
98
99             GetNormal() const = 0;
100            ReverseNormal() = 0;
101            SetOrder(const int px, const int py) = 0;
102            Integrate(const function_dp&, const
103
104            Integrate(const std::function<const
105
106            Integrate(const std::function<const
107            GetWeight(const int, const
108
109            IncreaseOrder() = 0;
110            GetMeasure() const = 0;
111
112            GetType() const = 0;
113            Clone() const = 0;
114            GetDoFs() const = 0;
115            GetNode(const int) const = 0;
116            GetNeighbour(const int) const = 0;
117            SetNeighbour(const int k, const int elem)
118
119            SetType(const int) = 0;
120            SetNode(const int, const int) = 0;
121            GetNumberOfNodes() const = 0;
122            GetShapeFunction(const int, const Point&)
123
124            GetGradShapeFunction(const int, const
125
126            GetNormal() const = 0;
127            ReverseNormal() = 0;
128            IncreaseOrder() = 0;
129            SetOrder(const int px, const int py) = 0;
130            Integrate(const std::function<const
131
132            Integrate(const std::function<const
133            Integrate(const std::function<const
134            GetMeasure() const = 0;

```

```

70         virtual const double                               GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const = 0;
71     };
72
73     template<class Shape, class ShapeFunction>
74     class CFiniteElement2D : public CElement2D<>
75     {
76     public:
77         CFiniteElement2D() {}
78         CFiniteElement2D(const int* nodes, const Point* points, const int dofs) :
79             m_shape{ nodes },
80             m_shapefunctions{ points, dofs },
81             m_type{ -1 } {
82             m_neighbours[0] = -1; m_neighbours[1] = -1;
83         };
84         CFiniteElement2D(const int* nodes, const Point* points, const int dofs, const int type) :
85             m_shape{ nodes, dofs },
86             m_shapefunctions{ points, dofs },
87             m_type{ type } {
88             m_neighbours[0] = -1; m_neighbours[1] = -1;
89         };
90         CFiniteElement2D(const int* nodes, const Point* points) :
91             m_shape{ nodes },
92             m_shapefunctions{ points },
93             m_type{ -1 } {
94             m_neighbours[0] = -1; m_neighbours[1] = -1;
95         };
96         CFiniteElement2D(const Shape& shape, const ShapeFunction& f) :
97             m_shape{ shape },
98             m_shapefunctions{ f },
99             m_type{ -1 } {
100             m_neighbours[0] = -1; m_neighbours[1] = -1;
101         };
102         CFiniteElement2D(const Shape& shape, const ShapeFunction& shfunc, const int type) :
103             m_shape{ shape },
104             m_shapefunctions{ shfunc },
105             m_type{ type } {
106             m_neighbours[0] = -1; m_neighbours[1] = -1;
107         };
108         CFiniteElement2D(const Shape& shape, const ShapeFunction& shfunc, const int type, const int*
neigs) :
109             m_shape{ shape },
110             m_shapefunctions{ shfunc },
111             m_type{ type } {
112             m_neighbours[0] = neigs[0]; m_neighbours[1] = neigs[1];
113         };
114         CFiniteElement2D(const CFiniteElement2D&e) :
115             m_shape{ e.m_shape },
116             m_shapefunctions{ e.m_shapefunctions },
117             m_type{ e.m_type } {
118             m_neighbours[0] = e.m_neighbours[0]; m_neighbours[1] = e.m_neighbours[1];
119         };
120         friend const bool operator==(const CFiniteElement2D& e1, const CFiniteElement2D& e2)
121         {
122             if (e1.m_shape == e2.m_shape)
123                 return true;
124             return false;
125         }
126         // don't forget to delete after the call
127         CElement2D<>* Clone() const
128         {
129             return new CFiniteElement2D(m_shape, m_shapefunctions, m_type,
m_neighbours);
130         };
131         ~CFiniteElement2D() {}
132         const int GetType() const;
133         const int GetNode(const int) const;
134         const int GetNeighbour(const int) const;
135         const Shape GetShape() const;
136         const ShapeFunction GetShapeFunctions() const;
137         const int GetDofs() const;
138         void SetNeighbour(const int k, const int elem);
139         void SetType(const int);
140         void SetShapeFunction(const int, const ShapeFunction&);
141         void SetShape(const Shape&);
142         const int SetOrder(const int px, const int py);
143         void SetNode(const int, const int);
144         const int GetNumberOfNodes() const;
145         //const int SetValue(const int number, CSolution* value);
146         const int IncreaseOrder();
147         const double GetMeasure() const;
148         //CSolution* GetValue(const int);
149         //void SetShapeFunction(const int, const std::function<const
DiffForm(const Point&)>&);
150         const double Integrate(const std::function<const double(const Point&)>&, const
std::vector<Point>& v) const;
151         const Point Integrate(const std::function<const Point(const Point&)>&, const

```

```

std::vector<Point>& v) const;
152     const std::vector<double>   Integrate(const std::function<const std::vector<double>(const
Point>&)>&, const std::vector<Point>&) const;
153     //const std::function<const DiffForm(const Point>&)>           GetShapeFunction(const int)
const;
154     //const DiffForm*           GetShapeFunction(const int, const Point>&);
155     const double               GetShapeFunction(const int, const Point>&) const;
156     const Point               GetGradShapeFunction(const int, const Point>&) const;
157     const Point               GetNormal() const;
158     void                       ReverseNormal();
159     const double               GetWeight(const int, const std::vector<Point>& verts, const
std::function<const double(const Point>&)>& f) const;
160     CFiniteElement2D&         operator=(const CFiniteElement2D& e)
161     {
162         m_shape = e.m_shape;
163         m_shapefunctions = e.m_shapefunctions;
164         m_type = e.m_type;
165         return *this;
166     }
167     friend std::istream&       operator>(std::istream& is, CFiniteElement2D& k)
168     {
169         is >> k.m_shape;
170         return is;
171     }
172     //const DiffForm           GetDShapeFunction(const int, const Point>&);
173 private:
174     Shape                     m_shape;
175     ShapeFunction             m_shapefunctions;
176     int                      m_type;
177     int                      m_neighbours[2];
178 };
179
180
181 template<class Shape, class ShapeFunction>
182 const int CFiniteElement2D<Shape, ShapeFunction>::GetType() const
183 {
184     return m_type;
185 }
186 template<class Shape, class ShapeFunction>
187 const double CFiniteElement2D<Shape, ShapeFunction>::GetMeasure() const
188 {
189     return m_shapefunctions.GetMeasure();
190 }
191 //template<class Shape, class ShapeFunction>
192 //const int CFiniteElement2D<Shape, ShapeFunction>::SetValue(const int number, CSolution* value)
193 //{
194 //    return m_shapefunctions.SetValue(number, value);
195 //}
196 //template<class Shape, class ShapeFunction>
197 //CSolution* CFiniteElement2D<Shape, ShapeFunction>::GetValue(const int p)
198 //{
199 //    auto&& val = m_shapefunctions.GetValue(p);
200 //    return const_cast<CSolution*>(static_cast<const CSolution*>(&val));
201 //}
202 template<class Shape, class ShapeFunction>
203 const int CFiniteElement2D<Shape, ShapeFunction>::GetNode(const int k) const
204 {
205     return m_shape.GetNode(k);
206 }
207 template<class Shape, class ShapeFunction>
208 const double CFiniteElement2D<Shape, ShapeFunction>::Integrate(const std::function<const
double(const Point>&)>&f, const std::vector<Point>& v) const
209 {
210     return m_shape.Integrate(f, v);
211 }
212
213 template<class Shape, class ShapeFunction>
214 const Point CFiniteElement2D<Shape, ShapeFunction>::Integrate(const std::function<const
Point(const Point>&)>&f, const std::vector<Point>& v) const
215 {
216     return m_shape.Integrate(f, v);
217 }
218
219 template<class Shape, class ShapeFunction>
220 const std::vector<double> CFiniteElement2D<Shape, ShapeFunction>::Integrate(const
std::function<const std::vector<double>(const Point>&)>&f, const std::vector<Point>& v) const
221 {
222     return m_shape.Integrate(f, v);
223 }
224 template<class Shape, class ShapeFunction>
225 const int CFiniteElement2D<Shape, ShapeFunction>::GetNeighbour(const int k) const
226 {
227     return m_neighbours[k];
228 }
229
230 template<class Shape, class ShapeFunction>
231 const Point CFiniteElement2D<Shape, ShapeFunction>::GetNormal() const

```



```

232     {
233         return m_shapefunctions.GetNormal();
234     }
235
236     template<class Shape, class ShapeFunction>
237     void CFiniteElement2D<Shape, ShapeFunction>::ReverseNormal()
238     {
239         m_shapefunctions.ReverseNormal();
240     }
241
242
243     template<class Shape, class ShapeFunction>
244     const Shape CFiniteElement2D<Shape, ShapeFunction>::GetShape() const
245     {
246         return m_shape;
247     }
248
249     template<class Shape, class ShapeFunction>
250     const ShapeFunction CFiniteElement2D<Shape, ShapeFunction>::GetShapeFunctions() const
251     {
252         return m_shapefunctions;
253     }
254
255
256     template<class Shape, class ShapeFunction>
257     const int CFiniteElement2D<Shape, ShapeFunction>::GetNumberOfNodes() const
258     {
259         return m_shape.GetNumberOfNodes();
260     }
261
262     template<class Shape, class ShapeFunction>
263     const int CFiniteElement2D<Shape, ShapeFunction>::GetDoFs() const
264     {
265         return m_shapefunctions.GetNumberOfShapeFunctions();
266     }
267
268     template<class Shape, class ShapeFunction>
269     void CFiniteElement2D<Shape, ShapeFunction>::SetNeighbour(const int k, const int elem)
270     {
271         m_neighbours[k] = elem;
272     }
273
274     template<class Shape, class ShapeFunction>
275     void CFiniteElement2D<Shape, ShapeFunction>::SetShapeFunction(const int k, const ShapeFunction&
func)
276     {
277         m_shapefunctions = func;
278     }
279
280     template<class Shape, class ShapeFunction>
281     void CFiniteElement2D<Shape, ShapeFunction>::SetShape(const Shape &shape)
282     {
283         m_shape = shape;
284     }
285
286     template<class Shape, class ShapeFunction>
287     void CFiniteElement2D<Shape, ShapeFunction>::SetType(const int k)
288     {
289         m_type = k;
290     }
291
292     template<class Shape, class ShapeFunction>
293     void CFiniteElement2D<Shape, ShapeFunction>::SetNode(const int k, const int node)
294     {
295         m_shape.SetNode(k, node);
296     }
297
298     template<class Shape, class ShapeFunction>
299     const double CFiniteElement2D<Shape, ShapeFunction>::GetShapeFunction(const int k, const
Mesh::Point &p) const
300     {
301         return m_shapefunctions.GetShapeFunction(k, p);
302     }
303
304     template<class Shape, class ShapeFunction>
305     const Point CFiniteElement2D<Shape, ShapeFunction>::GetGradShapeFunction(const int k, const
Mesh::Point &p) const
306     {
307         return m_shapefunctions.GetGradShapeFunction(k, p);
308     }
309
310     template<class Shape, class ShapeFunction>
311     const int CFiniteElement2D<Shape, ShapeFunction>::IncreaseOrder()
312     {
313         if(m_shape.IncreaseOrder())
314             return 1;
315         if (m_shapefunctions.IncreaseOrder())
316             return 1;

```

```

316         return 0;
317     }
318
319     template<class Shape, class ShapeFunction>
320     const int CFiniteElement2D<Shape, ShapeFunction>::SetOrder(const int px, const int py)
321     {
322         if(m_shape.SetOrder(px, py))
323             return 1;
324         if (m_shapefunctions.SetOrder(px, py))
325             return 1;
326         return 0;
327     }
328
329     template<class Shape, class ShapeFunction>
330     const double CFiniteElement2D<Shape, ShapeFunction>::GetWeight(const int node, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const
331     {
332         return m_shapefunctions.GetWeight(node, verts, f);
333     }
334     // fin.
335 }
336 }
337
338 #endif // FINITEELEMENT2D_H

```

7.26 CoreNCFEM/FiniteElements/Node.cpp File Reference

```
#include "Node.h"
```

7.27 CoreNCFEM/FiniteElements/Node.h File Reference

```

#include <stdio.h>
#include "Shape.h"
#include "ShapeFunction.h"
#include <iostream>

```

Classes

- class [corenc::Mesh::CNode](#)
- class [corenc::Mesh::CNodeBasis](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.28 Node.h

[Go to the documentation of this file.](#)

```

1  #ifndef Node_hpp
2  #define Node_hpp
3
4  #include <stdio.h>
5  #include "Shape.h"
6  #include "ShapeFunction.h"
7  #include <iostream>
8
9  namespace corenc
10 {
11     namespace Mesh
12     {
13         class CNode : public CShape
14         {
15         public:
16             CNode();
17             CNode(const CNode&);
18             CNode(const int n);
19             CNode(const int*n);
20             CNode& operator=(const CNode& e)
21             {
22                 m_node = e.m_node;
23                 return *this;
24             }
25             friend const bool operator==(const CNode& e1, const CNode& e2)
26             {
27                 if (e1.m_node == e2.m_node)
28                     return true;
29                 return false;
30             }
31             friend std::istream& operator>(std::istream& is, CNode& e)
32             {
33                 is >> e.m_node;
34                 --e.m_node;
35                 return is;
36             }
37             ~CNode() {};
38             const int
39             const int
40             const int
41             const int
42             void
43             const double
44             const Point
45             std::function<const Point(const Point*)>&, const std::vector<Point*>& v) const;
46             const std::vector<double>
47             std::vector<double>(const Point*)>&, const std::vector<Point*>& v) const;
48             private:
49                 const int
50                 int
51                 m_number = 1;
52                 m_node;
53             };
54             class CNodeBasis : public CShapeFunction<double>
55             {
56             public:
57                 CNodeBasis();
58                 CNodeBasis(const Point*);
59                 CNodeBasis(const CNodeBasis&e)
60                 {
61                     m_p0 = e.m_p0;
62                     m_normal = e.m_normal;
63                 };
64                 CNodeBasis& operator=(const CNodeBasis& e)
65                 {
66                     m_p0 = e.m_p0;
67                     m_normal = e.m_normal;
68                     return *this;
69                 }
70                 ~CNodeBasis() {};
71                 const int
72                 //const DForm<0>*
73                 const;
74                 const double
75                 Point*)> const;
76                 const Point
77                 const Point
78                 const Point
79                 void
80                 const double
81                 std::vector<Point*> verts, const std::function<const double(const Point*)>& f) const
82                 {
83                     GetNode(const int) const;
84                     GetNode(const NODES&) const;
85                     IncreaseOrder() { return 1; };
86                     GetNumberOfNodes() const;
87                     SetNode(const int k, const int node);
88                     Integrate(const std::function<const
89                                     Integrate(const
90                                     Integrate(const std::function<const
91                                     m_number = 1;
92                                     m_node;
93                                     GetNumberOfShapeFunctions() const;
94                                     GetShapeFunction(const int)
95                                     GetShapeFunction(const int, const
96                                     GetGradShapeFunction(const int,
97                                     GetNormal() const;
98                                     ReverseNormal();
99                                     GetWeight(const int node, const
100                                     }

```

```

76         return f(verts[node]);
77     };
78     //const int                               SetValue(const int, CSolution*
value);
79     //const int                               SetValue(const int, const
CFESolution& value);
80     const int                               IncreaseOrder() { return 1; };
81     //const CFESolution                       GetValue(const Point&) const;
82     const double                             GetMeasure() const { return 0.; };
83     //const CFESolution                       GetValue(const int) const;
84     //const std::function<const DForm<0>*(const Point&)>   GetShapeFunction(const int)
const;
85     private:
86         static const int                     m_number = 1;
87         Point                               m_p0;
88         Point                               m_normal;
89         //CFESolution                       m_w;
90         //const std::function<const double(const Point&p)>   m_psi[2];
91     };
92 }
93
94 }
95 #endif /* Node_hpp */

```

7.29 CoreNCFEM/FiniteElements/Rectangle.cpp File Reference

```
#include "Rectangle.h"
```

7.30 CoreNCFEM/FiniteElements/Rectangle.h File Reference

```

#include <stdio.h>
#include "Shape.h"
#include "ShapeFunction.h"
#include <iostream>

```

Classes

- class [corenc::Mesh::CRectangle](#)
- class [corenc::Mesh::CRectangleBasis](#)
- class [corenc::Mesh::CRectangleHBasis](#)
- class [corenc::Mesh::CRectangleBasis2x](#)
- class [corenc::Mesh::CRectangleBasis2y](#)
- class [corenc::Mesh::CRectangleBasis2](#)
- class [corenc::Mesh::CRectangleConstantBasis](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.31 Rectangle.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_MESH_RECTANGLE_H_
2 #define CORENC_MESH_RECTANGLE_H_
3
4 #include <stdio.h>
5 #include "Shape.h"
6 #include "ShapeFunction.h"
7 #include <iostream>
8 namespace corenc
9 {
10     namespace Mesh
11     {
12         class CRectangle : public CShape
13         {
14         public:
15             CRectangle();
16             CRectangle(const int n1, const int n2, const int n3, const int n4, const int order);
17             CRectangle(const int n1, const int n2, const int n3, const int n4, const int e1, const int
e2, const int e3, const int e4, const int order);
18             CRectangle(const int*, const int order);
19             CRectangle(const int*, const int*, const int order);
20             CRectangle(const CRectangle&);
21             CRectangle& operator=(const CRectangle& t)
22             {
23                 m_nodes = t.m_nodes;
24                 m_edges[0] = t.m_edges[0];
25                 m_edges[1] = t.m_edges[1];
26                 m_edges[2] = t.m_edges[2];
27                 m_edges[3] = t.m_edges[3];
28                 m_number = t.m_number;
29                 m_order = t.m_order;
30                 m_px = t.m_px;
31                 m_py = t.m_py;
32                 return *this;
33             }
34             const bool operator==(const CRectangle& t)
35             {
36                 for (unsigned int i = 0; i < 4; ++i)
37                     if (m_nodes[i] == t.m_nodes[0])
38                         for (unsigned int j = 0; j < 4; ++j)
39                             if (m_nodes[j] == t.m_nodes[1])
40                                 for (unsigned int k = 0; k < 4; ++k)
41                                     if (m_nodes[k] == t.m_nodes[2])
42                                         for (unsigned int l = 0; l < 4; ++l)
43                                             if (m_nodes[l] == t.m_nodes[3])
44                                                 return true;
45                 return false;
46             }
47             std::istream& operator>>(std::istream& is)
48             {
49                 is >> m_nodes[0] >> m_nodes[1] >> m_nodes[2] >> m_nodes[3];
50                 return is;
51             }
52             ~CRectangle() {}
53             const int GetNode(const int) const;
54             const int GetNode(const NODES&) const;
55             const int GetEdge(const int) const;
56             const int GetFacet(const int) const;
57             const int GetNumberOfNodes() const;
58             const int GetNumberOfEdges() const;
59             const int GetNumberOfFacets() const;
60             const double Integrate(const std::function<const
double(const Point&)>&, const std::vector<Point>& v) const;
61             const Point Integrate(const std::function<const
Point(const Point&)>&, const std::vector<Point>& v) const;
62             const std::vector<double> Integrate(const std::function<const
std::vector<double>(const Point&)>&, const std::vector<Point>& v) const;
63             void SetNode(const int k, const int node);
64             void IncreaseOrder();
65             void SetOrder(const int px, const int py);
66             void SetEdge(const int k, const int edge);
67             void SetFacet(const int k, const int facet);
68         private:
69             std::vector<int> m_nodes;
70             int m_edges[4];
71             int m_order;
72             int m_number;
73             int m_px, m_py;
74         };
75
76         class CRectangleBasis : public CShapeFunction<double>
77         {
78         public:

```

```

79         CRectangleBasis();
80         CRectangleBasis(const Point&, const Point&, const Point&, const Point&, const int order);
81         CRectangleBasis(const Point*, const int order);
82         CRectangleBasis(const CRectangleBasis&);
83         CRectangleBasis& operator=(const CRectangleBasis& t)
84         {
85             m_normal = t.m_normal;
86             m_det = t.m_det;
87             m_order = t.m_order;
88             m_ldorder = t.m_ldorder;
89             m_number = t.m_number;
90             m_s = t.m_s;
91             m_sp = t.m_sp;
92             m_points = t.m_points;
93             m_hx = t.m_hx;
94             m_hy = t.m_hy;
95             return *this;
96         }
97         ~CRectangleBasis() {}
98         const int                                     GetNumberOfShapeFunctions() const;
99         //const DForm<0>*                               GetShapeFunction(const int, const
Point&) const;
100         const double                                   GetShapeFunction(const int, const
Point&) const;
101         const Point                                     GetGradShapeFunction(const int, const
Point&) const;
102         const Point                                     GetNormal() const;
103         void                                             ReverseNormal();
104         const double                                   GetValue(const Point&) const;
105         const int                                       IncreaseOrder();
106         //const int                                     SetValue(const int, CSolution* value);
107         //CSolution*                                   GetValue(const unsigned int);
108         //const CFESolution                             GetValue(const int) const;
109         const double                                   GetMeasure() const { return m_det; };
110         const double                                   GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
111         //const unsigned int                             GetOrder() const;
112         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
113     private:
114         int                                             m_number;
115         int                                             m_order;
116         int                                             m_ldorder;
117         Point                                           m_normal;
118         std::vector<Mesh::Point>                       m_points;
119         double                                           m_det;
120         double                                           m_hx, m_hy;
121         const double                                   m_x1(const double) const;
122         const double                                   m_x2(const double) const;
123         const double                                   m_y1(const double) const;
124         const double                                   m_y2(const double) const;
125         void                                             compD(const Point&, const Point&, const
Point&);
126         void                                             compNormal(const Point&, const Point&,
const Point&);
127         const int                                       createS();
128         //std::vector<double>                           m_w[m_number];
129         //std::vector<CFESolution>                     m_w{ 4 };
130         int                                             m_s;
131         int                                             m_sp;
132     };
133
134     class CRectangleHBasis : public CShapeFunction<double>
135     {
136     public:
137         CRectangleHBasis();
138         CRectangleHBasis(const Point&, const Point&, const Point&, const Point&, const int order);
139         CRectangleHBasis(const Point&, const Point&, const Point&, const Point&, const int px, const
int py);
140         CRectangleHBasis(const Point*, const int order);
141         CRectangleHBasis(const Point*, const int px, const int py);
142         CRectangleHBasis(const CRectangleHBasis&);
143         CRectangleHBasis& operator=(const CRectangleHBasis& t)
144         {
145             m_normal = t.m_normal;
146             m_det = t.m_det;
147             m_order = t.m_order;
148             m_ldorder = t.m_ldorder;
149             m_number = t.m_number;
150             m_s = t.m_s;
151             m_sp = t.m_sp;
152             m_points = t.m_points;
153             m_hx = t.m_hx;
154             m_hy = t.m_hy;
155             m_px = t.m_px;
156             m_py = t.m_py;
157             return *this;
158         }

```

```

159         }
160         ~CRectangleHBasis() {};
161         const int                                     GetNumberOfShapeFunctions() const;
162         //const DForm<0>*                               GetShapeFunction(const int, const
Point&) const;
163         const double                                   GetShapeFunction(const int, const
Point&) const;
164         const Point                                   GetGradShapeFunction(const int, const
Point&) const;
165         const Point                                   GetNormal() const;
166         void                                           ReverseNormal();
167         const double                                   GetValue(const Point&) const;
168         const int                                     IncreaseOrder();
169         const int                                     SetOrder(const int px, const int py);
170         //const int                                     SetValue(const int, CSolution* value);
171         //CSolution*                                   GetValue(const unsigned int);
172         //const CFESolution                             GetValue(const int) const;
173         const double                                   GetMeasure() const { return m_det; };
174         const double                                   GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
175         //const unsigned int                             GetOrder() const;
176         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
177     private:
178         int                                           m_number;
179         int                                           m_order;
180         int                                           m_ldorder;
181         Point                                         m_normal;
182         std::vector<Mesh::Point>                     m_points;
183         double                                        m_det;
184         double                                        m_hx, m_hy;
185         int                                           m_px, m_py;
186         const double                                  m_x1(const double) const;
187         const double                                  m_x2(const double) const;
188         const double                                  m_y1(const double) const;
189         const double                                  m_y2(const double) const;
190         const double                                  m_xi(const double, const int n) const;
191         const double                                  m_dxi(const double, const int n) const;
192         void                                           compD(const Point&, const Point&, const
Point&);
193         void                                           compNormal(const Point&, const Point&,
const Point&);
194         const int                                     createS();
195         //std::vector<double>                           m_w[m_number];
196         //std::vector<CFESolution>                     m_w{ 4 };
197         int                                           m_s;
198         int                                           m_sp;
199     };
200
201     class CRectangleBasis2x : public CShapeFunction<double>
202     {
203     public:
204         CRectangleBasis2x();
205         CRectangleBasis2x(const Point&, const Point&, const Point&, const Point&, const int order);
206         CRectangleBasis2x(const Point*, const int order);
207         CRectangleBasis2x(const CRectangleBasis2x&);
208         CRectangleBasis2x& operator=(const CRectangleBasis2x& t)
209         {
210             m_normal = t.m_normal;
211             m_det = t.m_det;
212             m_order = t.m_order;
213             m_ldorder = t.m_ldorder;
214             m_number = t.m_number;
215             m_s = t.m_s;
216             m_sp = t.m_sp;
217             m_points = t.m_points;
218             m_hx = t.m_hx;
219             m_hy = t.m_hy;
220             return *this;
221         }
222         ~CRectangleBasis2x() {};
223         const int                                     GetNumberOfShapeFunctions() const;
224         //const DForm<0>*                               GetShapeFunction(const int, const
Point&) const;
225         const double                                   GetShapeFunction(const int, const
Point&) const;
226         const Point                                   GetGradShapeFunction(const int, const
Point&) const;
227         const Point                                   GetNormal() const;
228         void                                           ReverseNormal();
229         const double                                   GetValue(const Point&) const;
230         const int                                     IncreaseOrder();
231         //const int                                     SetValue(const int, CSolution* value);
232         //CSolution*                                   GetValue(const unsigned int);
233         //const CFESolution                             GetValue(const int) const;
234         const double                                   GetMeasure() const { return m_det; };
235         const double                                   GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;

```

```

236         //const unsigned int                                     GetOrder() const;
237         //const std::function<const DForm<0>*(const Point&)>   GetShapeFunction(const int) const;
238     private:
239         int                                                     m_number;
240         int                                                     m_order;
241         int                                                     m_ldorder;
242         Point                                                   m_normal;
243         std::vector<Mesh::Point>                                m_points;
244         double                                                  m_det;
245         double                                                  m_hx, m_hy;
246         const double                                            m_x1(const double) const;
247         const double                                            m_x2(const double) const;
248         const double                                            m_x3(const double) const;
249         const double                                            m_dx3(const double) const;
250         const double                                            m_y1(const double) const;
251         const double                                            m_y2(const double) const;
252         void                                                    compD(const Point&, const Point&, const
Point&);
253         void                                                    compNormal(const Point&, const Point&,
const Point&);
254         const int                                              createS();
255         //std::vector<double>                                    m_w[m_number];
256         //std::vector<CFESolution>                             m_w{ 4 };
257         int                                                    m_s;
258         int                                                    m_sp;
259     };
260
261     class CRectangleBasis2y : public CShapeFunction<double>
262     {
263     public:
264         CRectangleBasis2y();
265         CRectangleBasis2y(const Point&, const Point&, const Point&, const Point&, const int order);
266         CRectangleBasis2y(const Point*, const int order);
267         CRectangleBasis2y(const CRectangleBasis2y&);
268         CRectangleBasis2y& operator=(const CRectangleBasis2y& t)
269         {
270             m_normal = t.m_normal;
271             m_det = t.m_det;
272             m_order = t.m_order;
273             m_ldorder = t.m_ldorder;
274             m_number = t.m_number;
275             m_s = t.m_s;
276             m_sp = t.m_sp;
277             m_points = t.m_points;
278             m_hx = t.m_hx;
279             m_hy = t.m_hy;
280             return *this;
281         }
282         ~CRectangleBasis2y() {};
283         const int
284         //const DForm<0>*
Point&) const;
285         const double
Point&) const;
286         const Point
Point&) const;
287         const Point
288         void
289         const double
290         const int
291         //const int
292         //CSolution*
293         //const CFESolution
294         const double
295         const double
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
296         //const unsigned int                                     GetOrder() const;
297         //const std::function<const DForm<0>*(const Point&)>   GetShapeFunction(const int) const;
298     private:
299         int                                                     m_number;
300         int                                                     m_order;
301         int                                                     m_ldorder;
302         Point                                                   m_normal;
303         std::vector<Mesh::Point>                                m_points;
304         double                                                  m_det;
305         double                                                  m_hx, m_hy;
306         const double                                            m_x1(const double) const;
307         const double                                            m_x2(const double) const;
308         const double                                            m_y3(const double) const;
309         const double                                            m_dy3(const double) const;
310         const double                                            m_y1(const double) const;
311         const double                                            m_y2(const double) const;
312         void                                                    compD(const Point&, const Point&, const
Point&);
313         void                                                    compNormal(const Point&, const Point&,
const Point&);
314         const int                                              createS();

```



```

315         //std::vector<double>                                m_w[m_number];
316         //std::vector<CFESolution>                            m_w{ 4 };
317         int                                                    m_s;
318         int                                                    m_sp;
319     };
320
321     class CRectangleBasis2 : public CShapeFunction<double>
322     {
323     public:
324         CRectangleBasis2();
325         CRectangleBasis2(const Point&, const Point&, const Point&, const Point&, const int order);
326         CRectangleBasis2(const Point*, const int order);
327         CRectangleBasis2(const CRectangleBasis2&);
328         CRectangleBasis2& operator=(const CRectangleBasis2& t)
329         {
330             m_normal = t.m_normal;
331             m_det = t.m_det;
332             m_order = t.m_order;
333             m_ldorder = t.m_ldorder;
334             m_number = t.m_number;
335             m_s = t.m_s;
336             m_sp = t.m_sp;
337             m_points = t.m_points;
338             m_hx = t.m_hx;
339             m_hy = t.m_hy;
340             return *this;
341         }
342         ~CRectangleBasis2() {}
343         const int GetNumberOfShapeFunctions() const;
344         //const DForm<0>* GetShapeFunction(const int, const
345         Point&) const; const double GetShapeFunction(const int, const
346         Point&) const; const Point GetGradShapeFunction(const int, const
347         Point&) const; const Point GetNormal() const;
348         void ReverseNormal();
349         const double GetValue(const Point&) const;
350         const int IncreaseOrder();
351         //const int SetValue(const int, CSolution* value);
352         //CSolution* GetValue(const unsigned int);
353         //const CFESolution GetValue(const int) const;
354         const double GetMeasure() const { return m_det; };
355         const double GetWeight(const int, const
356         std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
357         //const unsigned int GetOrder() const;
358         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
359     private:
360         int m_number;
361         int m_order;
362         int m_ldorder;
363         Point m_normal;
364         std::vector<Mesh::Point> m_points;
365         double m_det;
366         double m_hx, m_hy;
367         const double m_x1(const double) const;
368         const double m_x2(const double) const;
369         const double m_x3(const double) const;
370         const double m_dx3(const double) const;
371         const double m_y1(const double) const;
372         const double m_y2(const double) const;
373         const double m_y3(const double) const;
374         const double m_dy3(const double) const;
375         void compD(const Point&, const Point&, const
376         Point&); void compNormal(const Point&, const Point&,
377         const Point&);
378         const int createS();
379         //std::vector<double> m_w[m_number];
380         //std::vector<CFESolution> m_w{ 4 };
381         int m_s;
382         int m_sp;
383     };
384
385     class CRectangleConstantBasis : public CShapeFunction<double>
386     {
387     public:
388         CRectangleConstantBasis();
389         CRectangleConstantBasis(const Point&, const Point&, const Point&, const Point&, const int
390         order);
391         CRectangleConstantBasis(const Point*, const int order);
392         CRectangleConstantBasis(const CRectangleConstantBasis&);
393         CRectangleConstantBasis& operator=(const CRectangleConstantBasis& t)
394         {
395             m_normal = t.m_normal;
396             m_det = t.m_det;
397             m_order = t.m_order;

```

```

395         m_ldorder = t.m_ldorder;
396         m_number = t.m_number;
397         m_s = t.m_s;
398         m_sp = t.m_sp;
399         m_points = t.m_points;
400         m_hx = t.m_hx;
401         m_hy = t.m_hy;
402         return *this;
403     }
404     ~CRectangleConstantBasis() {}
405     const int                                     GetNumberOfShapeFunctions() const;
406     //const DForm<0>*                               GetShapeFunction(const int, const
Point&) const;
407     const double                                   GetShapeFunction(const int, const
Point&) const;
408     const Point                                     GetGradShapeFunction(const int, const
Point&) const;
409     const Point                                     GetNormal() const;
410     void                                             ReverseNormal();
411     const double                                   GetValue(const Point&) const;
412     const int                                       IncreaseOrder();
413     //const int                                     SetValue(const int, CSolution* value);
414     //CSolution*                                   GetValue(const unsigned int);
415     //const CFESolution                             GetValue(const int) const;
416     const double                                   GetMeasure() const { return m_det; };
417     //const unsigned int                             GetOrder() const;
418     //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
419     private:
420         int                                         m_number;
421         int                                         m_order;
422         int                                         m_ldorder;
423         Point                                       m_normal;
424         std::vector<Mesh::Point>                  m_points;
425         double                                      m_det;
426         double                                      m_hx, m_hy;
427         const double                               m_x1(const double) const;
428         const double                               m_x2(const double) const;
429         const double                               m_y1(const double) const;
430         const double                               m_y2(const double) const;
431         void                                        compD(const Point&, const Point&, const
Point&);
432         void                                        compNormal(const Point&, const Point&,
const Point&);
433         const int                                   createS();
434         //std::vector<double>                       m_w[m_number];
435         //std::vector<CFESolution>                  m_w{ 1 };
436         int                                         m_s;
437         int                                         m_sp;
438     };
439 }
440 }
441 #endif // CORENC_MESH_RECTANGLE_H_

```

7.32 CoreNCFEM/FiniteElements/RectangleBasis2.cpp File Reference

```

#include "Rectangle.h"
#include <iostream>

```

7.33 CoreNCFEM/FiniteElements/RectangleBasis2y.cpp File Reference

```

#include "Rectangle.h"
#include <iostream>

```

7.34 CoreNCFEM/FiniteElements/RectangleHBasis.cpp File Reference

```

#include "Rectangle.h"
#include <random>

```

```
#include <iostream>
#include "../CoreNCA/Matrix.h"
#include "../CoreNCA/MatrixSkyline.h"
```

7.35 CoreNCFEM/FiniteElements/Shape.h File Reference

```
#include <functional>
#include <vector>
#include "../Point.h"
```

Classes

- class [corenc::Mesh::CShape](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

Typedefs

- using [corenc::scalar_func](#) = std::function< const double(const Mesh::Point &)>
- using [corenc::vector_func](#) = std::function< const Mesh::Point(const Mesh::Point &)>

Enumerations

- enum class [corenc::Mesh::NODES](#) { [corenc::Mesh::FIRST](#) , [corenc::Mesh::LAST](#) }

7.36 Shape.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_MESH_Shape_h
2 #define CORENC_MESH_Shape_h
3 #include <functional>
4 #include <vector>
5 #include "../Point.h"
6 namespace corenc
7 {
8     using scalar_func = std::function<const double(const Mesh::Point&);>;
9     using vector_func = std::function<const Mesh::Point(const Mesh::Point&);>;
10     namespace Mesh
11     {
12         //class Point;
13         enum class NODES
14         {
15             FIRST,
16             LAST
17         };
18         class CShape
19         {
20         public:
21             CShape() {}
22             CShape(const int*) {}
```

```

23         virtual ~CShape() {}
24         virtual const int
25         virtual const int
26         virtual const int
27         virtual const int
28         virtual const int
29         virtual const int
30         virtual const int
31         virtual const double
32         const = 0;
33         virtual const Point
34         const = 0;
35         virtual const std::vector<double>
36         std::vector<double>(const Point&)>&, const
37         virtual void
38         virtual void
39         virtual void
40         //virtual std::istream&
41     };
42 #endif /* CORENC_MESH_Shape_h */

```

```

GetNumberOfNodes() const { return 0; };
GetNumberOfEdges() const { return 0; };
GetNumberOfFacets() const { return 0; };
GetNode(const int) const { return 1; };
GetNode(const NODES&) const { return 1; };
GetEdge(const int) const { return -1; };
GetFacet(const int) const { return -1; };
Integrate(const scalar_func&, const std::vector<Point>&)
Integrate(const vector_func&, const std::vector<Point>&)
Integrate(const std::function<const
std::vector<Point>&) const = 0;
SetNode(const int, const int) = 0;
SetEdge(const int, const int) {};
SetFacet(const int, const int) {};
operator>(std::istream&) = 0;

```

7.37 CoreNCFEM/FiniteElements/ShapeFunction.h File Reference

```

#include "../Point.h"
#include <functional>
#include "../FESolution.h"

```

Classes

- class `corenc::Mesh::CShapeFunction< Type >`

Namespaces

- namespace `corenc`
- namespace `corenc::Mesh`

7.38 ShapeFunction.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_MESH_ShapeFunction_h
2 #define CORENC_MESH_ShapeFunction_h
3 #include "../Point.h"
4 #include <functional>
5 #include "../FESolution.h"
6 namespace corenc
7 {
8     namespace Mesh
9     {
10         template<class Type>
11         class CShapeFunction
12         {
13         public:
14             CShapeFunction() {}
15             CShapeFunction(const Point*) {}
16             virtual ~CShapeFunction() {}
17             virtual const int
18             //virtual const std::function<const DiffForm*(const Point*)> GetShapeFunction(const int)
19             const = 0;
20             //virtual const DiffForm* GetShapeFunction(const int) const = 0;
21             virtual const double
22             GetShapeFunction(const int, const Point&) const = 0;
23             virtual const Point
24             GetGradShapeFunction(const int, const Point&) const = 0;

```

```

22         virtual const Point          GetNormal() const = 0;
23         virtual void                  ReverseNormal() = 0;
24         virtual const double          GetMeasure() const = 0;
25         //virtual const Type          GetValue(const Point&) const = 0;
26         //virtual const int            SetValue(const unsigned int, const Type& value) = 0;
27         //virtual const Type          GetValue(const unsigned int) const = 0;
28         //virtual const int            SetValue(const )
29         //virtual CSolution*           GetValue(const unsigned int) = 0;
30         //virtual const int            SetValue(const int, CSolution*) = 0;
31     };
32 }
33 }
34 #endif /* CORENC_MESH_ShapeFunction_h */

```

7.39 CoreNCFEM/FiniteElements/Triangle.cpp File Reference

```

#include "Triangle.h"
#include <iostream>
#include <algorithm>
#include <random>
#include "../CoreNCA/Matrix.h"
#include "../CoreNCA/MatrixSkyline.h"

```

Functions

- const [Point](#) mid_point (const [Point](#) &p1, const [Point](#) &p2)
- const [Point](#) s_point (const [Point](#) &p1, const [Point](#) &p2, const double s)
- const [Point](#) center_point (const [Point](#) &p1, const [Point](#) &p2, const [Point](#) &p3)

7.39.1 Function Documentation

7.39.1.1 center_point()

```

const Point center_point (
    const Point & p1,
    const Point & p2,
    const Point & p3 )

```

7.39.1.2 mid_point()

```

const Point mid_point (
    const Point & p1,
    const Point & p2 )

```

7.39.1.3 s_point()

```
const Point s_point (
    const Point & p1,
    const Point & p2,
    const double s )
```

7.40 CoreNCFEM/FiniteElements/Triangle.h File Reference

```
#include <stdio.h>
#include "Shape.h"
#include "ShapeFunction.h"
#include <iostream>
```

Classes

- class [corenc::Mesh::CTriangle](#)
- class [corenc::Mesh::CTriangleBasis](#)
- class [corenc::Mesh::CTriangleLagrangeBasis](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.41 Triangle.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_MESH_TRIANGLE_H_
2 #define CORENC_MESH_TRIANGLE_H_
3
4 #include <stdio.h>
5 #include "Shape.h"
6 #include "ShapeFunction.h"
7 #include <iostream>
8 namespace corenc
9 {
10     namespace Mesh
11     {
12         class CTriangle : public CShape
13         {
14         public:
15             CTriangle();
16             CTriangle(const int n1, const int n2, const int n3, const int order);
17             CTriangle(const int n1, const int n2, const int n3, const int e1, const int e2, const int e3,
18                 const int order);
19             CTriangle(const int*, const int order);
20             CTriangle(const int*, const int*, const int order);
21             CTriangle(const CTriangle&);
22             CTriangle& operator=(const CTriangle& t)
23             {
24                 m_nodes = t.m_nodes;
25                 m_edges[0] = t.m_edges[0];
26                 m_edges[1] = t.m_edges[1];
27                 m_edges[2] = t.m_edges[2];
28                 m_number = t.m_number;
29                 m_order = t.m_order;
30                 return *this;
31             }
32         }
```

```

31     const bool    operator==(const CTriangle& t)
32     {
33         for (unsigned int i = 0; i < 3; ++i)
34             if (m_nodes[i] == t.m_nodes[0])
35                 for (unsigned int j = 0; j < 3; ++j)
36                     if (m_nodes[j] == t.m_nodes[1])
37                         for (unsigned int k = 0; k < 3; ++k)
38                             if (m_nodes[k] == t.m_nodes[2])
39                                 return true;
40         return false;
41     }
42     std::istream& operator>>(std::istream& is)
43     {
44         is >> m_nodes[0] >> m_nodes[1] >> m_nodes[2];
45         return is;
46     }
47     ~CTriangle() {}
48     const int      GetNode(const int) const;
49     const int      GetNode(const NODES&) const;
50     const int      GetEdge(const int) const;
51     const int      GetFacet(const int) const;
52     const int      GetNumberOfNodes() const;
53     const int      GetNumberOfEdges() const;
54     const int      GetNumberOfFacets() const;
55     const double   Integrate(const std::function<const
double(const Point&)>&, const std::vector<Point>& v) const;
56     const Point    Integrate(const std::function<const
Point(const Point&)>&, const std::vector<Point>& v) const;
57     const std::vector<double> Integrate(const std::function<const
std::vector<double>(const Point&)>&, const std::vector<Point>>) const;
58     void          SetNode(const int k, const int node);
59     const int      IncreaseOrder();
60     void          SetEdge(const int k, const int edge);
61     void          SetFacet(const int k, const int facet);
62     private:
63     std::vector<int> m_nodes;
64     int              m_edges[3];
65     int              m_order;
66     int              m_number;
67     void            SetOrder();
68 };
69
70 class CTriangleBasis : public CShapeFunction<double>
71 {
72 public:
73     CTriangleBasis();
74     CTriangleBasis(const Point&, const Point&, const Point&, const int order);
75     CTriangleBasis(const Point*, const int order);
76     CTriangleBasis(const CTriangleBasis&);
77     CTriangleBasis& operator=(const CTriangleBasis& t)
78     {
79         m_normal = t.m_normal;
80         m_det = t.m_det;
81         m_order = t.m_order;
82         m_ldorder = t.m_ldorder;
83         m_number = t.m_number;
84         m_alpha[0][0] = t.m_alpha[0][0];
85         m_alpha[0][1] = t.m_alpha[0][1];
86         m_alpha[0][2] = t.m_alpha[0][2];
87
88         m_alpha[1][0] = t.m_alpha[1][0];
89         m_alpha[1][1] = t.m_alpha[1][1];
90         m_alpha[1][2] = t.m_alpha[1][2];
91
92         m_alpha[2][0] = t.m_alpha[2][0];
93         m_alpha[2][1] = t.m_alpha[2][1];
94         m_alpha[2][2] = t.m_alpha[2][2];
95         m_s = t.m_s;
96         m_sp = t.m_sp;
97         m_all = t.m_all;
98         return *this;
99     }
100     ~CTriangleBasis() {}
101     const int      GetNumberOfShapeFunctions() const;
102     //const DForm<0>* GetShapeFunction(const int, const
Point&) const;
103     const double   GetShapeFunction(const int, const
Point&) const;
104     const Point    GetGradShapeFunction(const int, const
Point&) const;
105     const Point    GetNormal() const;
106     void          ReverseNormal();
107     const double   GetValue(const Point&) const;
108     const int      IncreaseOrder();
109     //const int     SetValue(const int, CSolution* value);
110     //CSolution*    GetValue(const unsigned int);
111     //const CFESolution GetValue(const int) const;

```

```

112         const double                                     GetMeasure() const { return fabs(m_det);
};
113         const double                                     GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
114         //const unsigned int                             GetOrder() const;
115         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
116     private:
117         int                                               m_number;
118         int                                               m_order;
119         int                                               m_ldorder;
120         double                                             m_alpha[3][3];
121         Point                                              m_normal;
122         double                                             m_det;
123         const double                                       m_L(const int, const Point&) const;
124         const double                                       m_xi(const int, const Point&) const;
125         void                                               compD(const Point&, const Point&, const
Point&);
126         void                                               compAlpha(const Point&, const Point&,
const Point&);
127         void                                               compNormal(const Point&, const Point&,
const Point&);
128         const int                                         createS();
129         //std::vector<double>                             m_w[m_number];
130         //std::vector<CFESolution>                       m_w{ 3 };
131         int                                               m_s;
132         int                                               m_sp;
133         std::vector<int>                                   m_all;
134     };
135
136     class CTriangleLagrangeBasis : public CShapeFunction<double>
137     {
138     public:
139         CTriangleLagrangeBasis();
140         CTriangleLagrangeBasis(const Point&, const Point&, const Point&, const int order);
141         CTriangleLagrangeBasis(const Point*, const int order);
142         CTriangleLagrangeBasis(const CTriangleLagrangeBasis&);
143         CTriangleLagrangeBasis& operator=(const CTriangleLagrangeBasis& t)
144         {
145             m_normal = t.m_normal;
146             m_det = t.m_det;
147             m_order = t.m_order;
148             m_ldorder = t.m_ldorder;
149             m_number = t.m_number;
150             m_alpha[0][0] = t.m_alpha[0][0];
151             m_alpha[0][1] = t.m_alpha[0][1];
152             m_alpha[0][2] = t.m_alpha[0][2];
153
154             m_alpha[1][0] = t.m_alpha[1][0];
155             m_alpha[1][1] = t.m_alpha[1][1];
156             m_alpha[1][2] = t.m_alpha[1][2];
157
158             m_alpha[2][0] = t.m_alpha[2][0];
159             m_alpha[2][1] = t.m_alpha[2][1];
160             m_alpha[2][2] = t.m_alpha[2][2];
161             m_s = t.m_s;
162             m_sp = t.m_sp;
163             m_all = t.m_all;
164             return *this;
165         }
166         ~CTriangleLagrangeBasis() {};
167         const int                                         GetNumberOfShapeFunctions() const;
168         //const DForm<0>*                                  GetShapeFunction(const int, const
Point&) const;
169         const double                                       GetShapeFunction(const int, const
Point&) const;
170         const Point                                        GetGradShapeFunction(const int, const
Point&) const;
171         const Point                                        GetNormal() const;
172         void                                               ReverseNormal();
173         const double                                       GetValue(const Point&) const;
174         const int                                           IncreaseOrder();
175         //const int                                         SetValue(const int, CSolution* value);
176         //CSolution*                                       GetValue(const unsigned int);
177         //const CFESolution                                GetValue(const int) const;
178         const double                                       GetAlpha(const int i, const int j) const
{ return m_alpha[i][j]; }
179         const double                                       GetMeasure() const { return fabs(m_det);
};
180         const double                                       GetWeight(const int, const
std::vector<Point>& verts, const std::function<const double(const Point&)>& f) const;
181         //const unsigned int                             GetOrder() const;
182         //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
183     private:
184         int                                               m_number;
185         int                                               m_order;
186         int                                               m_ldorder;
187         double                                             m_alpha[3][3];

```



```

188         Point
189         double
190         const double
191         const double
192         void
193     Point&);
194     void
195     const Point&);
196     void
197     const Point&);
198     const int
199     //std::vector<double>
200     //std::vector<CFESolution>
201     int
202     int
203     std::vector<int>
204     };
205 }
206 #endif // CORENC_MESH_TRIANGLE_H_

```

```

m_normal;
m_det;
m_L(const int, const Point&) const;
m_xi(const int, const Point&) const;
compD(const Point&, const Point&, const
compAlpha(const Point&, const Point&,
compNormal(const Point&, const Point&,
createS();
    m_w[m_number];
m_w{ 3 };
m_s;
m_sp;
m_all;

```

7.42 CoreNCFEM/FiniteElements/TriangleLagrange.cpp File Reference

```

#include "Triangle.h"
#include <iostream>
#include <algorithm>
#include <random>
#include ".../CoreNCA/Matrix.h"
#include ".../CoreNCA/MatrixSkyline.h"

```

Namespaces

- namespace [wtf](#)

Functions

- const [Point](#) [wtf::mid_point](#) (const [Point](#) &p1, const [Point](#) &p2)
- const [Point](#) [wtf::s_point](#) (const [Point](#) &p1, const [Point](#) &p2, const double s)
- const [Point](#) [wtf::center_point](#) (const [Point](#) &p1, const [Point](#) &p2, const [Point](#) &p3)

7.43 CoreNCFEM/FiniteElements/TriangleLinear.cpp File Reference

```

#include "TriangleLinear.h"
#include <iostream>

```

7.44 CoreNCFEM/FiniteElements/TriangleLinear.h File Reference

```

#include <stdio.h>
#include "Shape.h"
#include "ShapeFunction.h"
#include <iostream>

```

Classes

- class [corenc::Mesh::CTriangleLinear](#)
- class [corenc::Mesh::CTriangleLinearBasis](#)
- class [corenc::Mesh::CTriangleBasis](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.45 TriangleLinear.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_MESH_TRIANGLELINEAR_H_
2 #define CORENC_MESH_TRIANGLELINEAR_H_
3
4 #include <stdio.h>
5 #include "Shape.h"
6 #include "ShapeFunction.h"
7 #include <iostream>
8 namespace corenc
9 {
10     namespace Mesh
11     {
12         class CTriangleLinear : public CShape
13         {
14         public:
15             CTriangleLinear();
16             CTriangleLinear(const int n1, const int n2, const int n3);
17             CTriangleLinear(const int n1, const int n2, const int n3, const int e1, const int e2, const
int e3);
18             CTriangleLinear(const int*);
19             CTriangleLinear(const int*, const int*);
20             CTriangleLinear(const CTriangleLinear&);
21             CTriangleLinear& operator=(const CTriangleLinear& t)
22             {
23                 m_nodes[0] = t.m_nodes[0];
24                 m_nodes[1] = t.m_nodes[1];
25                 m_nodes[2] = t.m_nodes[2];
26                 m_edges[0] = t.m_edges[0];
27                 m_edges[1] = t.m_edges[1];
28                 m_edges[2] = t.m_edges[2];
29                 return *this;
30             }
31             const bool operator==(const CTriangleLinear& t)
32             {
33                 for (unsigned int i = 0; i < 3; ++i)
34                     if (m_nodes[i] != t.m_nodes[i])
35                         for (unsigned int j = 0; j < 3; ++j)
36                             if (m_nodes[j] != t.m_nodes[j])
37                                 for (unsigned int k = 0; k < 3; ++k)
38                                     if (m_nodes[k] != t.m_nodes[k])
39                                         return true;
40                 return false;
41             }
42             std::istream& operator>>(std::istream& is)
43             {
44                 is >> m_nodes[0] >> m_nodes[1] >> m_nodes[2];
45                 return is;
46             }
47             ~CTriangleLinear() {}
48             const int
49             const int
50             const int
51             const int
52             const int
53             const int
54             const int
55             const double
56             double(const Point&)>>, const std::vector<Point>& v) const;
57             const Point
58             Point(const Point&)>>, const std::vector<Point>& v) const;
59
60             GetNode(const int) const;
61             GetNode(const NODES&) const;
62             GetEdge(const int) const;
63             GetFacet(const int) const;
64             GetNumberOfNodes() const;
65             GetNumberOfEdges() const;
66             GetNumberOfFacets() const;
67             Integrate(const std::function<const
68             Integrate(const std::function<const

```

```

57         const std::vector<double>                                Integrate(const std::function<const
std::vector<double>>(const Point&)>&, const std::vector<Point>>&) const;
58         void                                                    SetNode(const int k, const int node);
59         const int                                                IncreaseOrder() { return 1; };
60         void                                                    SetEdge(const int k, const int edge);
61         void                                                    SetFacet(const int k, const int facet);
62     private:
63         int                                                       m_nodes[3];
64         int                                                       m_edges[3];
65     };
66
67     class CTriangleLinearBasis : public CShapeFunction<double>
68     {
69     public:
70         CTriangleLinearBasis();
71         CTriangleLinearBasis(const Point&, const Point&, const Point&);
72         CTriangleLinearBasis(const Point*);
73         CTriangleLinearBasis(const CTriangleLinearBasis&);
74         CTriangleLinearBasis& operator=(const CTriangleLinearBasis& t)
75         {
76             m_normal = t.m_normal;
77             m_det = t.m_det;
78             m_alpha[0][0] = t.m_alpha[0][0];
79             m_alpha[0][1] = t.m_alpha[0][1];
80             m_alpha[0][2] = t.m_alpha[0][2];
81
82             m_alpha[1][0] = t.m_alpha[1][0];
83             m_alpha[1][1] = t.m_alpha[1][1];
84             m_alpha[1][2] = t.m_alpha[1][2];
85
86             m_alpha[2][0] = t.m_alpha[2][0];
87             m_alpha[2][1] = t.m_alpha[2][1];
88             m_alpha[2][2] = t.m_alpha[2][2];
89             return *this;
90         }
91         ~CTriangleLinearBasis() {};
92         const int                                                GetNumberOfShapeFunctions() const;
93         //const DForm<0>*                                         GetShapeFunction(const int, const
Point&) const;
94         const double                                             GetShapeFunction(const int, const Point&)
const;
95         const Point                                              GetGradShapeFunction(const int, const
Point&) const;
96         const Point                                              GetNormal() const;
97         void                                                    ReverseNormal();
98         const double                                             GetValue(const Point&) const;
99         const int                                                IncreaseOrder() { return 1; };
100        //const int                                              SetValue(const int, CSolution* value);
101        //CSolution*                                             GetValue(const unsigned int);
102        //const CFESolution                                       GetValue(const int) const;
103        const double                                             GetMeasure() const { return m_det; };
104        //const std::function<const DForm<0>*(const Point&)>      GetShapeFunction(const int) const;
105    private:
106        static const int                                         m_number = 3;
107        double                                                    m_alpha[3][3];
108        Point                                                     m_normal;
109        double                                                    m_det;
110        const double                                             L(const int, const Point&) const;
111        void                                                      compD(const Point&, const Point&, const
Point&);
112        void                                                      compAlpha(const Point&, const Point&,
const Point&);
113        void                                                      compNormal(const Point&, const Point&,
const Point&);
114        //std::vector<double>                                     m_w[m_number];
115        //std::vector<CFESolution>                                m_w[3];
116    };
117
118     class CTriangleBasis : public CShapeFunction<double>
119     {
120     public:
121         CTriangleBasis();
122         CTriangleBasis(const Point&, const Point&, const Point&, const int order);
123         CTriangleBasis(const Point*, const int order);
124         CTriangleBasis(const CTriangleBasis&);
125         CTriangleBasis& operator=(const CTriangleBasis& t)
126         {
127             m_normal = t.m_normal;
128             m_det = t.m_det;
129             m_order = t.m_order;
130             m_number = t.m_number;
131             m_alpha[0][0] = t.m_alpha[0][0];
132             m_alpha[0][1] = t.m_alpha[0][1];
133             m_alpha[0][2] = t.m_alpha[0][2];
134
135             m_alpha[1][0] = t.m_alpha[1][0];
136             m_alpha[1][1] = t.m_alpha[1][1];

```

```

137         m_alpha[1][2] = t.m_alpha[1][2];
138
139         m_alpha[2][0] = t.m_alpha[2][0];
140         m_alpha[2][1] = t.m_alpha[2][1];
141         m_alpha[2][2] = t.m_alpha[2][2];
142         return *this;
143     }
144     ~CTriangleBasis() {};
145     const int                                     GetNumberOfShapeFunctions() const;
146     //const DForm<0>*                               GetShapeFunction(const int, const
Point&) const;
147     const double                                   GetShapeFunction(const int, const
Point&) const;
148     const Point                                     GetGradShapeFunction(const int, const
Point&) const;
149     const Point                                     GetNormal() const;
150     void                                             ReverseNormal();
151     const double                                   GetValue(const Point&) const;
152     //const int                                     SetValue(const int, CSolution* value);
153     //CSolution*                                   GetValue(const unsigned int);
154     //const CFESolution                             GetValue(const int) const;
155     //const std::function<const DForm<0>*(const Point&)> GetShapeFunction(const int) const;
156     private:
157         int                                         m_number;
158         int                                         m_order;
159         double                                     m_alpha[3][3];
160         Point                                       m_normal;
161         double                                     m_det;
162         const double                               L(const int, const Point&) const;
163         void                                       compD(const Point&, const Point&, const
Point&);
164         void                                       compAlpha(const Point&, const Point&,
const Point&);
165         void                                       compNormal(const Point&, const Point&,
const Point&);
166         //std::vector<double>                       m_w[m_number];
167         //std::vector<CFESolution>                 m_w{ 3 };
168     };
169 }
170 }
171 #endif // CORENC_MESH_TRIANGLELINEAR_H_

```

7.46 CoreNCFEM/FiniteSolver.h File Reference

Classes

- class [corenc::CFiniteSolver< Method, Mesh, Solver >](#)

Namespaces

- namespace [corenc](#)

7.47 FiniteSolver.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_FINITESOLVER_H
2 #define CORENC_FINITESOLVER_H
3
4 namespace corenc
5 {
6     template<class Method, class Mesh, class Solver>
7     class CFiniteSolver
8     {
9     public:
10         CFiniteSolver() {};
11         ~CFiniteSolver() {};
12         void                                     Solve();
13     private:
14         Method*                                 m_method;
15         Mesh*                                   m_mesh;

```

```

16         Solver*                m_solver;
17     };
18
19     template<class Method, class Mesh, class Solver>
20     void CFiniteSolver<Method, Mesh, Solver>::Solve()
21     {
22         m_method->Assemble();
23         return;
24     }
25 }
26
27 #endif // !CORENC_FFINITESOLVER_H
28

```

7.48 CoreNCFEM/GaussianField.h File Reference

```

#include <algorithm>
#include <vector>
#include <cmath>
#include "Point.h"

```

Classes

- struct [corenc::GaussianProcess](#)
- struct [corenc::GaussianKernel](#)

Namespaces

- namespace [corenc](#)

7.49 GaussianField.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_GAUSSIANPROCESS_H_
2 #define CORENC_GAUSSIANPROCESS_H_
3 #include <algorithm>
4 #include <vector>
5 #include <cmath>
6 #include "Point.h"
7 namespace corenc
8 {
9     struct GaussianProcess
10     {
11         double sigma2;
12         double l;
13         double a;
14         double b;
15         double c;
16         double A;
17         double B;
18         size_t K = 1;
19         std::vector<double> lambda;
20         GaussianProcess(const double L, const size_t num)
21         {
22             K = num;
23             sigma2 = L;
24             l = 2 * L;
25             a = 1. / (4 * sigma2);
26             b = 1. / (2 * l * l);
27             c = sqrt(a * a + 2 * a * b);
28             A = a + b + c;
29             B = b / A;
30             lambda.resize(K);

```

```

31         for (size_t i = 0; i < K; ++i)
32             lambda[i] = std::pow(B, i) * sqrt(2 * a / A);
33     }
34     const double He(const int i, const double x) const
35     {
36         switch (i)
37         {
38             case 0:
39                 return 1.;
40             case 1:
41                 return x;
42             case 2:
43                 return x * x - 1.;
44             case 3:
45                 return x * x * x - 3. * x;
46             case 4:
47                 return x * x * x * x - 6. * x * x + 3.;
48             case 5:
49                 return x * x * x * x * x - 10. * x * x * x + 15. * x;
50             case 6:
51                 return x * x * x * x * x * x - 15. * x * x * x * x + 45. * x * x - 25.;
52             case 7:
53                 return x * x * x * x * x * x * x - 21. * x * x * x * x * x + 105. * x * x * x - 105. * x;
54             case 8:
55                 return x * x * x * x * x * x * x * x - 28. * x * x * x * x * x * x + 210. * x * x * x * x * x
- 420. * x * x * x + 105.;
56             case 9:
57                 return x * x * x * x * x * x * x * x * x - 36. * x * x * x * x * x * x * x + 378. * x * x * x
* x * x * x - 1260. * x * x * x * x + 945. * x;
58             default:
59                 return x * x * x * x * x * x * x * x * x * x - 45. * x * x * x * x * x * x * x * x + 630.
* x * x * x * x * x * x * x - 3150. * x * x * x * x * x + 4725. * x * x * x - 945.;
60             break;
61         }
62     };
63     const double phi(const int i, const double x) const
64     {
65         return exp(-(c - a) * x * x) * He(i, x * sqrt(2 * c));
66     };
67 };
68 /*enum class gkernels
69 {
70     gexponent,
71     gker1,
72     gker2,
73     gker3
74 };*/
75 struct GaussianKernel
76 {
77     int N;
78     const double gpexp(const Mesh::Point& a) const
79     {
80         return exp(-12.5 * (a.x * a.x + a.y * a.y));
81     }
82
83     const double gpstep(const Mesh::Point& a) const
84     {
85         if (fabs(a.x) < 0.5 && fabs(a.y) < 0.5)
86             return 1.;
87         return 0.;
88     }
89
90     std::vector<Mesh::Point> _centrs;
91     GaussianKernel(const int _n, const std::vector<Mesh::Point>& centers) :
92         N(_n), _centrs{ centers } {}
93     const double get_gp(const std::vector<double>& a, const Mesh::Point& p) const
94     {
95         double sum = 0;
96         for (auto i = 0; i < N; ++i)
97             sum += a[i] * gpexp(p - _centrs[i]);
98         return sum;
99     }
100 };
101 }
102 #endif // CORENC_GAUSSIANPROCESS_H_

```

7.50 CoreNCFEM/Grids/Mesh1D.cpp File Reference

```

#include "Mesh1D.h"
#include "../FiniteElements/Node.h"

```

```
#include "../FiniteElements/Edge.h"
#include <iostream>
```

7.51 CoreNCFEM/Grids/Mesh1D.h File Reference

```
#include <stdio.h>
#include "../Mesh.h"
#include "../Point.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
#include <unordered_map>
#include <typeinfo>
```

Classes

- class [corenc::Mesh::CMesh1D](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.52 Mesh1D.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_Mesh1D_hpp
2 #define CORENC_Mesh1D_hpp
3
4 #include <stdio.h>
5 #include "../Mesh.h"
6
7 #include "../Point.h"
8 #include <iostream>
9 #include <fstream>
10 #include <vector>
11 #include <string>
12 #include <map>
13 #include <unordered_map>
14 #include <typeinfo>
15 namespace corenc
16 {
17     namespace Mesh
18     {
19
20         class CMesh1D : public CMesh<CFESolution>
21         {
22         public:
23             CMesh1D();
24             CMesh1D(const std::string& domain_name);
25             CMesh1D(const std::string& domain_file, const std::string& init_file);
26             CMesh1D(const double x0, const double x1, const unsigned n, const int order, const
27                 std::function<const double(const Point&)>& init_func);
28             CMesh1D(const double x0, const double x1, const unsigned n, const int order, const
29                 std::function<const double(const Point&)>& init_func, const std::function<const double(const
30                     Point&)>& init_derivative);
```

```

28         CMesh1D(const CMesh1D&);
29         CMesh1D& operator=(const CMesh1D& m)
30         {
31             auto sz = m.m_elems.size();
32             m_elems.resize(sz);
33             for (int i = 0; i < sz; ++i)
34                 m_elems[i] = m.m_elems[i]->Clone();
35         }
36         const unsigned int          GetNumberOfElements() const;
37         const unsigned int          GetNumberOfNodes() const;
38         const unsigned int          GetNumberOfBoundaries() const;
39         const int                   FindElement(const Point&) const;
40         const Point                 GetNode(const unsigned int) const;
41         const CElement<CFESolution>* GetElement(const unsigned int) const;
42         const CElement<CFESolution>* GetBoundary(const unsigned int) const;
43         const double                getSolution(const unsigned int element, const
unsigned int node) const;
44         const double                getParameter(Parameters, const unsigned int,
const Point& p) const;
45         const double                getParameter(Parameters, const unsigned int,
const int) const;
46         const std::vector<double>    getSolution() const { return m_solution; };
47         const int                   updateSolution(const std::vector<double>&
new_solution);
48         const int                   updateSolution(const unsigned int element, const
unsigned int node, const double value);
49         const int                   updateSolution(const unsigned int element, const
unsigned int node, CSolution* value);
50         const int                   updateSolution(const unsigned int node, const
double value);
51         const int                   setParameter(Parameters, const double, const
unsigned int);
52         const double                getMinSize() const { return m_minsize; };
53         ~CMesh1D();
54     private:
55         std::vector<CElement<CFESolution>*> m_elems;
56         std::vector<CElement<CFESolution>*> m_bnds;
57         std::vector<Point> m_points;
58         std::vector<double> m_solution;
59         std::vector<int> m_nums;
60         std::vector<double> m_params;
61         double m_minsize{0.};
62     public:
63         auto GetElements() -> decltype(m_elems) { return
m_elems; };
64         auto GetBoundary() -> decltype(m_bnds) { return
m_bnds; };
65     };
66 }
67 }
68 #endif /* CORENC_Mesh1D_hpp */

```

7.53 CoreNCFEM/Grids/RegularMesh.cpp File Reference

```

#include <stdio.h>
#include "RegularMesh.h"
#include "../FiniteElements/Rectangle.h"
#include "../FiniteElements/Edge.h"
#include <iostream>
#include <algorithm>
#include <numeric>

```

Functions

- `template<class T >`
`vector< size_t > sort_indexes (const vector< T > &v)`

7.53.1 Function Documentation

7.53.1.1 sort_indexes()

```
template<class T >
vector< size_t > sort_indexes (
    const vector< T > & v )
```

7.54 CoreNCFEM/Grids/RegularMesh.h File Reference

```
#include "../Mesh.h"
#include "../FiniteElements/FiniteElement2D.h"
#include "../Point.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
#include "../Parameter.h"
```

Classes

- class [corenc::Mesh::CRegularMesh](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.55 RegularMesh.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_MESH_RegularMesh_h
2 #define CORENC_MESH_RegularMesh_h
3 #include "../Mesh.h"
4 #include "../FiniteElements/FiniteElement2D.h"
5 #include "../Point.h"
6 #include <iostream>
7 #include <fstream>
8 #include <vector>
9 #include <string>
10 #include <map>
11 #include "../Parameter.h"
12 namespace corenc
13 {
14     namespace Mesh
15     {
16         class CRegularMesh// : public CMesh<>
17         {
18         public:
19             CRegularMesh();
20             CRegularMesh(const std::string& file_name);
21             CRegularMesh(const CRegularMesh&);
22             // nx ny number of elements on x y
23             CRegularMesh(const Point& p1, const Point& p2, const int nx, const int ny);
24             CRegularMesh(const Point& p1, const Point& p2, const int nx, const int ny, const int px,
25                 const int py);
26             CRegularMesh(const double x1, const double y1, const double x2, const double y2, const int
27                 nx, const int ny);
28             CRegularMesh& operator=(const CRegularMesh& tr)
```

```

27     {
28         const int sz_el = (int)tr.m_elems.size();
29         const int sz_pt = (int)tr.m_points.size();
30         const int sz_bpt = (int)tr.m_basepoints.size();
31         const int sz_bel = (int)tr.m_elemsbase.size();
32         const int sz_ed = (int)tr.m_edges.size();
33         const int sz_bed = (int)tr.m_edgesbase.size();
34         m_elems.resize(sz_el);
35         m_edges.resize(sz_ed);
36         m_points.resize(sz_pt);
37         m_basepoints.resize(sz_bpt);
38         m_elemsbase.resize(sz_bel);
39         m_edgesbase.resize(sz_bed);
40         int i = 0;
41         for (i = 0; i < sz_el; ++i)
42             m_elems[i] = tr.m_elems[i]->Clone();
43         for (i = 0; i < sz_ed; ++i)
44             m_edges[i] = tr.m_edges[i]->Clone();
45         for (i = 0; i < sz_pt; ++i)
46             m_points[i] = tr.m_points[i];
47         for (i = 0; i < sz_bpt; ++i)
48             m_basepoints[i] = tr.m_basepoints[i];
49         for (i = 0; i < sz_bel; ++i)
50             m_elemsbase[i] = tr.m_elemsbase[i]->Clone();
51         for (i = 0; i < sz_bed; ++i)
52             m_edgesbase[i] = tr.m_edgesbase[i]->Clone();
53         m_bnds.resize(tr.m_bnds.size());
54         for (i = 0; i < m_bnds.size(); ++i)
55             m_bnds[i] = tr.m_bnds[i];
56         m_offsets = tr.m_offsets;
57         m_params = tr.m_params;
58         m_order = tr.m_order;
59         m_inodes = tr.m_inodes;
60         return *this;
61     }
62     CRegularMesh*          Clone() const
63     {
64         return new CRegularMesh(*this);
65     };
66     const unsigned int      GetNumberOfElements() const;
67     const unsigned int      GetNumberOfNodes() const;
68     const int               GetNumberOfINodes() const;
69     const unsigned int      GetNumberOfBoundaries() const;
70     const int               FindElement(const Point&) const;
71     const Point             GetNode(const unsigned int) const;
72     const CElement2D<>*&    GetElement(const unsigned int) const;
73     const CElement<>*&      GetBoundary(const unsigned int) const;
74     const double            GetMinSize() const { return 0.; };
75     const double            getSolution(const unsigned int element, const unsigned
int node) const;
76     const int               updateSolution(const unsigned int element, const unsigned
int node, const double value);
77     const std::vector<double> getSolution() const;
78     const int               updateSolution(const std::vector<double>&);
79     const int               updateSolution(const unsigned int element, const unsigned
int node, CSolution* value);
80     const double            getParameter(Parameters, const unsigned int, const
Point&) const;
81     const double            getParameter(Parameters, const unsigned int, const int)
const;
82     const int               setParameter(Parameters, const double, const unsigned
int);
83     const int               setParameter(const CParameter&, const unsigned int type);
84     const int               updateSolution(const unsigned int node, const double
value);
85     const int               refine_hx();
86     const int               refine_hy();
87     const int               refine_h();
88     const int               refine_p();
89     const int               refine_hp();
90     const int               interpolate(const int node) const;
91     ~CRegularMesh();
92     private:
93         std::vector<CElement2D<>*> m_elems;
94         std::vector<CElement<>*> m_edges;
95         std::vector<Point> m_points;
96         std::vector<Point> m_basepoints;
97         std::vector<CElement2D<>*> m_elemsbase;
98         std::vector<CElement<>*> m_edgesbase;
99         std::map<int, CParameter> m_params;
100         std::vector<int> m_offsets;
101         std::vector<int> m_bnds;
102         const double      CompSquare(const Point& p1, const Point& p2, const
Point& p3) const;
103         int m_order;
104         int m_inodes;
105     public:

```

```

106         auto                                     GetElements() -> decltype(m_elems) { return m_elems; };
107         auto                                     GetBoundary() -> decltype(m_edges) { return m_edges; };
108     };
109 }
110 }
111 #endif /* CORENC_MESH_RegularMesh_h */
112

```

7.56 CoreNCFEM/Grids/RegularMesh3D.cpp File Reference

```

#include <stdio.h>
#include "RegularMesh3D.h"
#include "../FiniteElements/Cube.h"
#include "../FiniteElements/Edge.h"
#include <iostream>
#include <algorithm>
#include <numeric>

```

Functions

- `template<class T >`
`vector< size_t > sort_indexes (const vector< T > &v)`

7.56.1 Function Documentation

7.56.1.1 `sort_indexes()`

```

template<class T >
vector< size_t > sort_indexes (
    const vector< T > & v )

```

7.57 CoreNCFEM/Grids/RegularMesh3D.h File Reference

```

#include "../Mesh.h"
#include "../FiniteElements/FiniteElement2D.h"
#include "../Point.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
#include "../Parameter.h"

```

Classes

- class `corenc::Mesh::CRegularMesh3D`

Namespaces

- namespace `corenc`
- namespace `corenc::Mesh`

7.58 RegularMesh3D.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_MESH_RegularMesh3D_h
2 #define CORENC_MESH_RegularMesh3D_h
3 #include "../Mesh.h"
4 #include "../FiniteElements/FiniteElement2D.h"
5 #include "../Point.h"
6 #include <iostream>
7 #include <fstream>
8 #include <vector>
9 #include <string>
10 #include <map>
11 #include "../Parameter.h"
12 namespace corenc
13 {
14     namespace Mesh
15     {
16         class CRegularMesh3D// : public CMesh<
17         {
18         public:
19             CRegularMesh3D();
20             CRegularMesh3D(const std::string& file_name);
21             CRegularMesh3D(const CRegularMesh3D&);
22             // nx ny number of elements on x y
23             CRegularMesh3D(const Point& p1, const Point& p2, const int nx, const int ny);
24             CRegularMesh3D(const Point& p1, const Point& p2, const int nx, const int ny, const int px,
25             const int py);
26             CRegularMesh3D(const double x1, const double y1, const double x2, const double y2, const int
27             nx, const int ny);
28             CRegularMesh3D& operator=(const CRegularMesh3D& tr)
29             {
30                 const int sz_el = (int)tr.m_elems.size();
31                 const int sz_pt = (int)tr.m_points.size();
32                 const int sz_bpt = (int)tr.m_basepoints.size();
33                 const int sz_bel = (int)tr.m_elemsbase.size();
34                 const int sz_ed = (int)tr.m_edges.size();
35                 const int sz_bed = (int)tr.m_edgesbase.size();
36                 m_elems.resize(sz_el);
37                 m_edges.resize(sz_ed);
38                 m_points.resize(sz_pt);
39                 m_basepoints.resize(sz_bpt);
40                 m_elemsbase.resize(sz_bel);
41                 m_edgesbase.resize(sz_bed);
42                 int i = 0;
43                 for (i = 0; i < sz_el; ++i)
44                     m_elems[i] = tr.m_elems[i]->Clone();
45                 for (i = 0; i < sz_ed; ++i)
46                     m_edges[i] = tr.m_edges[i]->Clone();
47                 for (i = 0; i < sz_pt; ++i)
48                     m_points[i] = tr.m_points[i];
49                 for (i = 0; i < sz_bpt; ++i)
50                     m_basepoints[i] = tr.m_basepoints[i];
51                 for (i = 0; i < sz_bel; ++i)
52                     m_elemsbase[i] = tr.m_elemsbase[i]->Clone();
53                 for (i = 0; i < sz_bed; ++i)
54                     m_edgesbase[i] = tr.m_edgesbase[i]->Clone();
55                 m_bnds.resize(tr.m_bnds.size());
56                 for (i = 0; i < m_bnds.size(); ++i)
57                     m_bnds[i] = tr.m_bnds[i];
58                 m_offsets = tr.m_offsets;
59                 m_params = tr.m_params;
60                 m_order = tr.m_order;
61                 m_inodes = tr.m_inodes;
62                 return *this;
63             }
64             CRegularMesh3D* Clone() const
65             {
66                 return new CRegularMesh3D(*this);
67             };
68             const unsigned int GetNumberOfElements() const;
69             const unsigned int GetNumberOfNodes() const;
70             const int GetNumberOfINodes() const;
71             const unsigned int GetNumberOfBoundaries() const;
72             const int FindElement(const Point&) const;

```

```

71         const Point
72         const CElement<>*
73         const CElement<>*
74         const double
75         const double
76     int node) const;
77     const int
78     int node, const double value);
79     const std::vector<double>
80     const int
81     const int
82     int node, CSolution* value);
83     const double
84     Point&) const;
85     const double
86     const;
87     const int
88     int);
89     const int
90     const int
91     value);
92     const int
93     const int
94     const int
95     const int
96     const int
97     const int
98     const int
99     ~CRegularMesh3D();
100 private:
101     std::vector<CElement<>*>
102     std::vector<CElement<>*>
103     std::vector<Point>
104     std::vector<Point>
105     std::vector<CElement<>*>
106     std::vector<CElement<>*>
107     std::vector<CElement<>*>
108     std::map<int, CParameter>
109     std::vector<int>
110     std::vector<int>
111     const double
112     Point& p3) const;
113     int
114     int
115     public:
116     auto
117     auto
118     };
119 }
120 }
121 #endif /* CORENC_MESH_RegularMesh3D_h */
122
GetNode(const unsigned int) const;
GetElement(const unsigned int) const;
GetBoundary(const unsigned int) const;
getMinSize() const { return 0.; };
getSolution(const unsigned int element, const unsigned
updateSolution(const unsigned int element, const unsigned
getSolution() const;
updateSolution(const std::vector<double>&);
updateSolution(const unsigned int element, const unsigned
getParameter(Parameters, const unsigned int, const
getParameter(Parameters, const unsigned int, const int)
setParameter(Parameters, const double, const unsigned
setParameter(const CParameter&, const unsigned int type);
updateSolution(const unsigned int node, const double
refine_hx();
refine_hy();
refine_h();
refine_p();
refine_hp();
interpolate(const int node) const;
m_elems;
m_edges;
m_points;
m_basepoints;
m_elemsbase;
m_edgesbase;
m_params;
m_offsets;
m_bnds;
CompSquare(const Point& p1, const Point& p2, const
m_order;
m_inodes;
GetElements() -> decltype(m_elems) { return m_elems; };
GetBoundary() -> decltype(m_edges) { return m_edges; };

```

7.59 CoreNCFEM/Grids/TriangularMesh.cpp File Reference

```

#include <stdio.h>
#include "TriangularMesh.h"
#include "../FiniteElements/Triangle.h"
#include "../FiniteElements/Edge.h"
#include <iostream>
#include <algorithm>
#include <numeric>
#include <random>

```

Functions

- `template<class T >`
`vector< size_t > sort_indexes (const vector< T > &v)`

7.59.1 Function Documentation

7.59.1.1 sort_indexes()

```
template<class T >
vector< size_t > sort_indexes (
    const vector< T > & v )
```

7.60 CoreNCFEM/Grids/TriangularMesh.h File Reference

```
#include "../Mesh.h"
#include "../Point.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
#include "../Parameter.h"
```

Classes

- class [corenc::Mesh::CTriangularMesh](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.61 TriangularMesh.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_MESH_TriangularMesh_h
2 #define CORENC_MESH_TriangularMesh_h
3 #include "../Mesh.h"
4
5 #include "../Point.h"
6 #include <iostream>
7 #include <fstream>
8 #include <vector>
9 #include <string>
10 #include <map>
11 #include "../Parameter.h"
12 namespace corenc
13 {
14     namespace Mesh
15     {
16         class CTriangularMesh : public CMesh<>
17         {
18         public:
19             CTriangularMesh();
20             CTriangularMesh(const std::string& file_name);
21             CTriangularMesh(const CTriangularMesh&);
22             CTriangularMesh(const Point& p1, const Point& p2, const int nx, const int ny);
23             CTriangularMesh& operator=(const CTriangularMesh& tr)
24             {
25                 const int sz_el = tr.m_elems.size();
26                 const int sz_pt = tr.m_points.size();
27                 const int sz_bpt = tr.m_basepoints.size();
28                 const int sz_bel = tr.m_elemsbase.size();
29                 const int sz_ed = tr.m_edges.size();
30                 const int sz_bed = tr.m_edgesbase.size();
```

```

31         m_elems.resize(sz_el);
32         m_edges.resize(sz_ed);
33         m_points.resize(sz_pt);
34         m_basepoints.resize(sz_bpt);
35         m_elemsbase.resize(sz_bel);
36         m_edgesbase.resize(sz_bed);
37         int i = 0;
38         for (i = 0; i < sz_el; ++i)
39             m_elems[i] = tr.m_elems[i]->Clone();
40         for (i = 0; i < sz_ed; ++i)
41             m_edges[i] = tr.m_edges[i]->Clone();
42         for (i = 0; i < sz_pt; ++i)
43             m_points[i] = tr.m_points[i];
44         for (i = 0; i < sz_bpt; ++i)
45             m_basepoints[i] = tr.m_basepoints[i];
46         for (i = 0; i < sz_bel; ++i)
47             m_elemsbase[i] = tr.m_elemsbase[i]->Clone();
48         for (i = 0; i < sz_bed; ++i)
49             m_edgesbase[i] = tr.m_edgesbase[i]->Clone();
50         m_params = tr.m_params;
51         m_bnds.resize(tr.m_bnds.size());
52         for (i = 0; i < m_bnds.size(); ++i)
53             m_bnds[i] = tr.m_bnds[i];
54         m_order = tr.m_order;
55         m_offsets = tr.m_offsets;
56         return *this;
57     }
58     CTriangularMesh*      Clone() const
59     {
60         return new CTriangularMesh(*this);
61     };
62     //const bool operator<(const CTriangularMesh& mesh) const
63     //{
64     //    if(m_points.size() < mesh.m_points.size())
65     //        return true;
66     //    return false;
67     //}
68     const unsigned int      GetNumberOfElements() const;
69     const unsigned int      GetNumberOfNodes() const;
70     const unsigned int      GetNumberOfBoundaries() const;
71     const int               FindElement(const Point&) const;
72     const Point             GetNode(const unsigned int) const;
73     const CElement<>*       GetElement(const unsigned int) const;
74     const CElement<>*       GetBoundary(const unsigned int) const;
75     const double            GetMinSize() const { return 0.; };
76     const double            GetSolution(const unsigned int element, const unsigned
77     int node) const;
78     const int               updateSolution(const unsigned int element, const unsigned
79     int node, const double value);
80     const std::vector<double> getSolution() const;
81     const int               updateSolution(const std::vector<double>&);
82     const int               updateSolution(const unsigned int element, const unsigned
83     int node, CSolution* value);
84     const double            getParameter(Parameters, const unsigned int, const
85     Point&) const;
86     const double            getParameter(Parameters, const unsigned int, const int)
87     const;
88     const int               setParameter(Parameters, const double, const unsigned
89     int);
90     const int               setParameter(const CParameter&, const unsigned int type);
91     const int               updateSolution(const unsigned int node, const double
92     value);
93     const int               refine_h();
94     const int               refine_p();
95     const int               refine_hp();
96     const int               set4thOrder();
97     const int               set2ndOrder();
98     const int               set3rdOrder();
99     const int               interpolate(const int node) const;
100    const int               GetNumberOfINodes() const;
101    ~CTriangularMesh();
102    private:
103    std::vector<CElement<>*> m_elems;
104    std::vector<CElement<>*> m_edges;
105    std::vector<Point> m_points;
106    std::vector<Point> m_basepoints;
107    std::vector<CElement<>*> m_elemsbase;
108    std::vector<CElement<>*> m_edgesbase;
109    std::map<int, CParameter> m_params;
110    std::vector<int> m_offsets;
111    std::vector<int> m_bnds;
112    const double      CompSquare(const Point& p1, const Point& p2, const
113    Point& p3) const;
114    void              set3rdNodes();
115    void              set4thNodes_1();
116    void              set4thNodes_2();
117    int               m_order;

```

```

110         public:
111             auto GetElements() -> decltype(m_elems) { return m_elems; };
112             auto GetBoundary() -> decltype(m_edges) { return m_edges; };
113         };
114     };
115 }
116 #endif /* CORENC_MESH_TriangularMesh_h */
117

```

7.62 CoreNCFEM/Grids/TriangularMeshLinear.cpp File Reference

```

#include <stdio.h>
#include "TriangularMeshLinear.h"
#include "../FiniteElements/TriangleLinear.h"
#include "../FiniteElements/Edge.h"
#include <iostream>
#include <algorithm>

```

7.63 CoreNCFEM/Grids/TriangularMeshLinear.h File Reference

```

#include "../Mesh.h"
#include "../Point.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <map>
#include "../Parameter.h"

```

Classes

- class [corenc::Mesh::CTriangularMeshLinear](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

7.64 TriangularMeshLinear.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_MESH_TriangularMesh_h
2 #define CORENC_MESH_TriangularMesh_h
3 #include "../Mesh.h"
4
5 #include "../Point.h"
6 #include <iostream>
7 #include <fstream>
8 #include <vector>
9 #include <string>
10 #include <map>

```



```

11 #include "../Parameter.h"
12 namespace corenc
13 {
14     namespace Mesh
15     {
16         class CTriangularMeshLinear : public CMesh<>
17         {
18         public:
19             CTriangularMeshLinear();
20             CTriangularMeshLinear(const std::string& file_name);
21             CTriangularMeshLinear(const CTriangularMeshLinear&);
22             const unsigned int      GetNumberOfElements() const;
23             const unsigned int      GetNumberOfNodes() const;
24             const unsigned int      GetNumberOfBoundaries() const;
25             const int               FindElement(const Point&) const;
26             const Point             GetNode(const unsigned int) const;
27             const CElement<>*       GetElement(const unsigned int) const;
28             const CElement<>*       GetBoundary(const unsigned int) const;
29             const double            getMinSize() const { return 0.; };
30             const double            getSolution(const unsigned int element, const unsigned
int node) const;
31             const int              updateSolution(const unsigned int element, const unsigned
int node, const double value);
32             const std::vector<double> getSolution() const;
33             const int              updateSolution(const std::vector<double>&);
34             const int              updateSolution(const unsigned int element, const unsigned
int node, CSolution* value);
35             const double           getParameter(Parameters, const unsigned int, const
Point&) const;
36             const double           getParameter(Parameters, const unsigned int, const int)
const;
37             const int              setParameter(Parameters, const double, const unsigned
int);
38             const int              setParameter(const CParameter&, const unsigned int type);
39             const int              updateSolution(const unsigned int node, const double
value);
40             const int              refine_h();
41             ~CTriangularMeshLinear();
42         private:
43             std::vector<CElement<>*> m_elems;
44             std::vector<CElement<>*> m_edges;
45             std::vector<Point> m_points;
46             std::map<int, CParameter> m_params;
47         public:
48             auto GetElements() -> decltype(m_elems) { return m_elems; };
49             auto GetBoundary() -> decltype(m_edges) { return m_edges; };
50         };
51     }
52 }
53 #endif /* CORENC_MESH_TriangularMesh_h */
54

```

7.65 CoreNCFEM/Mesh.h File Reference

```
#include "FiniteElements/FiniteElement.h"
```

Classes

- class `corenc::Mesh::CMesh< T >`
- class `corenc::Mesh::CMesh< bool >`

Namespaces

- namespace `corenc`
- namespace `corenc::Mesh`

Enumerations

- enum `corenc::Mesh::Meshes` { `corenc::Mesh::Mesh1D` = 0 , `corenc::Mesh::TriangularMesh` = 1 , `corenc::Mesh::TetrahedralMesh` = 2 }

7.66 Mesh.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_MESH_Mesh_h
2 #define CORENC_MESH_Mesh_h
3 #include "FiniteElements/FiniteElement.h"
4 namespace corenc
5 {
6     namespace Mesh
7     {
8         enum Meshes
9         {
10             Mesh1D = 0,
11             TriangularMesh = 1,
12             TetrahedralMesh = 2
13         };
14         template<class T = bool>
15         class CMesh;
16         template<class T>
17         class CMesh
18         {
19         public:
20             CMesh() {}
21             virtual ~CMesh() {}
22             virtual const unsigned int
23             virtual const unsigned int
24             virtual const int
25             virtual const unsigned int
26             virtual const CElement<T>*
27             virtual const CElement<T>*
28             virtual const Point
29             virtual const double
30             unsigned int node) const = 0;
31             virtual const int
32             unsigned int node, const double value) = 0;
33             virtual const std::vector<double>
34             virtual const int
35             //virtual const int
36             virtual const int
37             unsigned int node, CSolution* value) = 0;
38             virtual const double
39             virtual const double
40             Point&) const = 0;
41             virtual const double
42             int) const = 0;
43             virtual const int
44             int) = 0;
45             virtual const double
46             virtual const int
47             value) = 0;
48             unsigned int node, const double value) = 0;
49             virtual const std::vector<double>
50             virtual const int
51             //virtual const int
52             virtual const int
53             unsigned int node, CSolution* value) = 0;
54             virtual const double
55             Point& p) const = 0;
56             GetNumberOfNodes() const = 0;
57             GetNumberOfElements() const = 0;
58             FindElement(const Point&) const = 0;
59             GetNumberOfBoundaries() const = 0;
60             GetElement(const unsigned int) const = 0;
61             GetBoundary(const unsigned int) const = 0;
62             GetNode(const unsigned int) const = 0;
63             GetSolution(const unsigned int element, const
64             updateSolution(const unsigned int element, const
65             getSolution() const = 0;
66             updateSolution(const std::vector<double>&) = 0;
67             updateSolution(const std::vector<CSolution*>&) = 0;
68             updateSolution(const unsigned int element, const
69             getParameter(Parameters, const unsigned int, const
70             getParameter(Parameters, const unsigned int, const
71             setParameter(Parameters, const double, const unsigned
72             getMinSize() const = 0;
73             updateSolution(const unsigned int node, const double
74             GetNumberOfNodes() const = 0;
75             GetNumberOfElements() const = 0;
76             FindElement(const Point&) const = 0;
77             GetNumberOfBoundaries() const = 0;
78             GetElement(const unsigned int) const = 0;
79             GetBoundary(const unsigned int) const = 0;
80             GetNode(const unsigned int) const = 0;
81             GetSolution(const unsigned int element, const
82             updateSolution(const unsigned int element, const
83             getSolution() const = 0;
84             updateSolution(const std::vector<double>&) = 0;
85             updateSolution(const std::vector<CSolution*>&) = 0;
86             updateSolution(const unsigned int element, const
87             getParameter(Parameters, const unsigned int, const

```

```

61         virtual const double           getParameter(Parameters, const unsigned int, const
        int) const = 0;
62         virtual const int             setParameter(Parameters, const double, const unsigned
        int) = 0;
63         virtual const double           getMinSize() const = 0;
64     };
65 }
66 }
67
68 #endif /* CORENC_MESH_Mesh_h */

```

7.67 CoreNCFEM/Methods/CSMethod.h File Reference

Classes

- class [Methods::CSMethod](#)

Namespaces

- namespace [Methods](#)

7.68 CSMethod.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENCFEM_METHODS_CSMethod_h
2 #define CORENCFEM_METHODS_CSMethod_h
3
4 namespace Methods
5 {
6     class CSMethod
7     {
8     public:
9         CSMethod() {}
10        virtual ~CSMethod() {}
11    };
12 }
13
14 #endif /* CORENCFEM_METHODS_CSMethod_h */

```

7.69 CoreNCFEM/Methods/dg_flux.h File Reference

Namespaces

- namespace [corenc](#)
- namespace [corenc::method](#)

Enumerations

- enum class [corenc::method::DGFlux](#) {
[corenc::method::EIP](#), [corenc::method::EBaumannOden](#), [corenc::method::EBaumannOdenIP](#), [corenc::method::ENIPG](#),
[corenc::method::EUpwind](#), [corenc::method::ECentral](#), [corenc::method::ELaxFriedrichs](#), [corenc::method::IIP](#),
[corenc::method::IBaumannOden](#), [corenc::method::IBaumannOdenIP](#), [corenc::method::INIPG](#), [corenc::method::IUpwind](#),
[corenc::method::ICentral](#), [corenc::method::ILaxFriedrichs](#), [corenc::method::CUSTOM](#), [corenc::method::NOFLUX](#)
}

7.70 dg_flux.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_METHOD_DG_FLUX_H_
2 #define CORENC_METHOD_DG_FLUX_H_
3 namespace corenc
4 {
5     namespace method
6     {
7         enum class DGFlux
8         {
9             EIP,
10            EBaumannOden,
11            EBaumannOdenIP,
12            ENIPG,
13            EUpwind,
14            ECentral,
15            ELaxFriedrichs,
16            IIP,
17            IBaumannOden,
18            IBaumannOdenIP,
19            INIPG,
20            IUpwind,
21            ICentral,
22            ILaxFriedrichs,
23            CUSTOM,
24            NOFLUX,
25        };
26    }
27 }
28 #endif // !CORENC_METHOD_DG_FLUX_H_

```

7.71 CoreNCFEM/Methods/DGMethod.h File Reference

```

#include <functional>
#include <set>
#include "../Point.h"
#include "../Parameter.h"
#include "CSMethod.h"
#include <memory>
#include <cmath>
#include <map>
#include <algorithm>
#include <vector>
#include <iostream>
#include <fstream>
#include <string>

```

Classes

- class [corenc::method::CDGMethod< Type >](#)
- class [corenc::method::DGMethod< Problem, Grid, Matrix >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)
- namespace [corenc::method](#)

7.72 DGMethod.h

[Go to the documentation of this file.](#)

```

1  #ifndef DGMethod_H
2  #define DGMethod_H
3
4  // DGMethod.h describes an abstract interface and functions for a DG method with zero Dirichlet boundaries
5  #include <functional>
6  #include <set>
7  #include "../Point.h"
8  #include "../Parameter.h"
9  #include "CSMethod.h"
10 #include <memory>
11 #include <cmath>
12 #include <map>
13 #include <algorithm>
14 #include <vector>
15 #include <iostream>
16 #include <fstream>
17 #include <string>
18 namespace corenc
19 {
20     namespace Mesh
21     {
22         class Point;
23     }
24     namespace method
25     {
26         // class Type = Type of the solution, for ex vector or double, or even more specific
27
28         template<class Type>
29         class CDGMethod
30         {
31         public:
32             CDGMethod() {};
33             virtual ~CDGMethod() {};
34             virtual const int Assemble() = 0;
35             virtual const Type GetSolution(const std::vector<double>& point)
36             const = 0;
37             virtual const std::vector<Type> GetSolution() const = 0;
38             virtual const Type GetMaxSolution() const = 0;
39             virtual const Type GetMinSolution() const = 0;
40         };
41
42         template<class Problem, class Grid, class Matrix>
43         class DGMethod
44         {
45         public:
46             DGMethod() :
47                 m_problem{nullptr},
48                 m_Grid{nullptr},
49                 m_GlobalMatrix{nullptr},
50                 m_RightMatrix{nullptr},
51                 m_rhsvector{nullptr}
52             {}
53             DGMethod(
54                 Problem* p,
55                 Grid* g,
56                 Matrix* m,
57                 std::vector<double>* rhs):
58                 m_problem{ p },
59                 m_Grid{ g->Clone() },
60                 m_GlobalMatrix{ m },
61                 m_N( g->GetNumberOfElements() ),
62                 m_Ns( g->GetNumberOfBoundaries() ),
63                 m_rhsvector{ rhs }{
64                 //GeneratePortrait();
65             }
66             DGMethod(
67                 Problem* p,
68                 Grid* g,
69                 Matrix* m,
70                 Matrix* rm,
71                 std::vector<double>* rhs):
72                 m_problem{ p },
73                 m_Grid{ g->Clone() },
74                 m_GlobalMatrix{ m },
75                 m_RightMatrix{ rm },
76                 m_N( g->GetNumberOfElements() ),
77                 m_Ns( g->GetNumberOfBoundaries() ),
78                 m_rhsvector{ rhs }{
79                 //GeneratePortrait();
80             }
81             DGMethod(const std::shared_ptr<Grid>& grid) :m_Grid{ grid->Clone() } {}

```

```

82         DGMMethod(Grid* grid) :m_Grid{ grid->Clone() } {}
83         DGMMethod(const DGMMethod& meth) :
84             m_Grid{ meth.m_Grid->Clone() },
85             //m_GlobalMatrix{ meth.m_GlobalMatrix->Clone() },
86             //m_rhsvector{ meth.m_rhsvector },
87             //m_problem{ meth.m_problem },
88             m_time{ meth.m_time },
89             //m_solution{ meth.m_solution },
90             m_size{ meth.m_size },
91             m_N{ meth.m_N },
92             m_Ns{ meth.m_Ns },
93             m_nums{ meth.m_nums }
94         {};
95         void Discretization();
96         const double GetValue(const Mesh::Point&) const;
97         const double GetValue(const Mesh::Point&, const std::vector<double>& vec)
98     const;
99         const double GetValue(const Mesh::Point&, const std::vector<double>& vec,
100     const int num) const;
101         //const Mesh::Point GetGradValue(const Mesh::Point&, const std::vector<double>& vec)
102     const;
103         //const Mesh::Point GetLambdaGrad(const Mesh::Point&, const std::vector<double>&
104     vec) const;
105         const double GetEffective(const std::vector<double>& vec) const;
106         void ProjectSolution(std::vector<double>&, std::function<const
107     double(const Mesh::Point&, const std::vector<double>&, const int)> GetValue, std::vector<double>&
108     sol);
109         void ProjectSolution(std::vector<double>&, std::function<const
110     double(const Mesh::Point&, const std::vector<double>&)> GetValue, std::vector<double>& sol, const
111     int);
112         void LoadSolution(const std::vector<double>& vec);
113         const std::vector<double> SetSolution(const int sol, const int liq, const double, const
114     double, const double);
115         void GetSolution(std::vector<double>& vec);
116         void Rediscretization(const std::shared_ptr<Grid>&);
117         void Rediscretization();
118         void SetTimeStep(const double& step) { m_step = step; m_time = step;
119     }
120     Matrix* GetGlobalMatrix() const;
121     Grid* GetMesh() { return m_Grid; }
122     const std::vector<double> GetRightVector() const;
123     void OutDatFormat(const Mesh::Point& min, const Mesh::Point& max,
124     const std::string& file_name, const std::vector<double>& vec) const;
125     void OutMeshFormat(const std::string& file_name, const
126     std::vector<double>& vec);
127     void OutMeshTimeFormat(const std::string& file_name, const
128     std::vector<double>& vec);
129     static const double GetSolution(const Grid& g, const std::vector<double> &weights,
130     const Mesh::Point& p);
131     static const double GetSolution(const Grid& g, const std::vector<double> &weights,
132     const Mesh::Point& p, const int nfem);
133     static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
134     const Mesh::Point& p);
135     static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
136     const Mesh::Point& p, const int n);
137     ~DGMMethod();
138     private:
139     void GeneratePortrait();
140     void AssemblGlobal();
141     void MainConditions();
142     void SecondConditions();
143     void ThirdConditions();
144     void StefanConditions();
145     void ApplySources();
146     const int AssembleLocalMatrix(const int);
147     const int AssembleIDUDVMMatrix(const int);
148     const int AssembleIDUVMMatrix(const int);
149     const int AssembleIUDVMMatrix(const int);
150     const int AssembleRUVMatrix(const int);
151     const int AssembleSUPGMatrix(const int);
152     const int AssembleLocalMatrix(const int, const int);
153     const int AssembleInter();
154     Grid* m_Grid = nullptr;
155     Matrix* m_GlobalMatrix = nullptr;
156     Matrix* m_RightMatrix = nullptr;
157     Problem* m_problem = nullptr;
158     std::vector<double> m_solution;
159     std::vector<double>* m_rhsvector;
160     unsigned int m_size;
161     double m_step{ 0.1 };
162     double m_time{ 0.1 };
163     unsigned int m_N;
164     unsigned int m_Ns;
165     std::vector<unsigned int> m_nums;
166     // interpolation nodes
167     std::vector<std::vector<int>> m_inums;

```

```

152     };
153
154     template<class Problem, class Grid, class Matrix>
155     void DGMethod<Problem, Grid, Matrix>::Discretization()
156     {
157         GeneratePortrait();
158         AssemblGlobal();
159         AssembleInter();
160         //ApplySources();
161         //SecondConditions();
162         //ThirdConditions();
163         MainConditions();
164         //StefanConditions();
165     }
166     template<class Problem, class Grid, class Matrix>
167     void DGMethod<Problem, Grid, Matrix>::GeneratePortrait()
168     {
169         const auto& el = m_Grid->GetElement(0);
170         int order = m_Grid->GetElement(0)->GetDoFs();
171         std::vector<std::set<unsigned int> temp;
172         //m_Ns = m_Grid->GetNumberOfNodes();
173         m_Ns = m_Grid->GetNumberOfBoundaries();
174         m_N = m_Grid->GetNumberOfElements();
175         //temp.resize(m_Grid->GetNumberOfNodes());
176         unsigned i, j, k;
177         m_nums.resize(m_N);
178         m_inums.resize(m_N);
179         int size;
180         m_size = 0;
181         std::cout << "nums" << std::endl;
182         for (k = 0; k < m_N; ++k)
183         {
184             const auto& elem{ m_Grid->GetElement(k) };
185             size = 0;
186             m_inums[k].resize(order);
187             for (i = 0; i < order; ++i)
188             {
189                 {
190                     m_inums[k][i] = size;
191                     ++size;
192                 }
193             }
194             m_nums[k] = m_size;
195             m_size += size;
196             std::cout << k << "\t" << m_nums[k] << std::endl;
197         }
198         int sz = m_Ns;
199         int nk, ne;
200         int sizej = 0;
201         int sizei = 0;
202         temp.resize(m_size);
203         for (k = 0; k < sz; ++k)
204         {
205             auto bound = m_Grid->GetBoundary(k);
206             nk = bound->GetNeighbour(0);
207             ne = bound->GetNeighbour(1);
208             std::cout << nk << ne << std::endl;
209             sizei = 0;
210             sizej = 0;
211             if (ne != -1)
212             {
213                 auto elemk = m_Grid->GetElement(nk);
214                 auto eleme = m_Grid->GetElement(ne);
215                 size = 0;
216                 for (i = 0; i < order; ++i)
217                 {
218                     for (j = i + 1; j < order; ++j)
219                     {
220                         {
221                             temp[m_nums[nk] + m_inums[nk][j]].insert(m_nums[nk] +
222                             m_inums[nk][i]);
223                         }
224                     }
225                 }
226                 for (i = 0; i < order; ++i)
227                 {
228                     for (j = 0; j < order; ++j)
229                     {
230                         int jnode = m_Grid->interpolate(eleme->GetNode(j));
231                         temp[m_nums[ne] + m_inums[ne][j]].insert(m_nums[nk] +
232                         m_inums[nk][i]);
233                     }
234                 }
235                 else
236                 {
237                     sizei = 0;

```

```

237         sizej = 0;
238         auto elemk = m_Grid->GetElement(nk);
239         size = 0;
240         for (i = 0; i < order; ++i)
241         {
242             for (j = i + 1; j < order; ++j)
243             {
244                 temp[m_nums[nk] + m_inums[nk][j]].insert(m_nums[nk] +
m_inums[nk][i]);
//temp[m_nums[nk] + sizej].insert(m_nums[nk] + sizei);
            }
        }
247     }
248 }
249 }
250 if(m_problem->findTerm(Terms::RUV))
251     m_RightMatrix->Create(temp.size(), temp);
252
253 // for (auto & it : temp)
254 // {
255 //     for (auto& it2 : it)
256 //         std::cout << it2 << "\t";
257 //     std::cout << std::endl;
258 // }
259 //m_GlobalMatrix = std::shared_ptr<Matrix>(new Matrix(m_Grid->GetNumberOfNodes(), temp));
260 //m_rhsvector.resize(m_Grid->GetNumberOfNodes());
261 //std::cout << temp.size() << std::endl;
262 m_GlobalMatrix->Create(temp.size(), temp);
263 m_rhsvector->resize(temp.size());
264 //m_solution.resize(m_Grid->GetNumberOfNodes());
265 for (int l = 0; l < m_Grid->GetNumberOfNodes(); ++l)
266     m_solution[l] = 20;
267 }
268 template<class Problem, class Grid, class Matrix>
269 void DGMethod<Problem, Grid, Matrix>::AssemblGlobal()
270 {
271     int l;
272     //std::vector<std::future<int>> futures;
273     int i, j, k, nodes;
274     double mij;
275     const int terms{ (int)m_problem->getNumberOfTerms() };
276     for (k = 0; k < terms; ++k)
277     {
278         switch (m_problem->getTerm(k))
279         {
280             case Terms::IDUDV:
281                 for (l = 0; l < m_N; ++l)
282                 {
283                     AssembleIDUDVMatrix(l);
284                 }
285                 break;
286             case Terms::IDUV:
287                 for (l = 0; l < m_N; ++l)
288                     AssembleIDUVMatrix(l);
289                 break;
290             case Terms::IUDV:
291                 for (l = 0; l < m_N; ++l)
292                     AssembleIUDVMatrix(l);
293                 break;
294             case Terms::SUPG:
295                 for (l = 0; l < m_N; ++l)
296                     AssembleSUPGMatrix(l);
297                 break;
298             case Terms::RUV:
299                 for (l = 0; l < m_N; ++l)
300                     AssembleRUVMatrix(l);
301                 break;
302             default:
303                 break;
304         }
305     }
306     //for (l = 0; l < m_N; ++l)
307         //futures.push_back(async(&DGMethod<Problem, Grid, Matrix>::AssembleLocalMatrix, this,
1));
308     // AssembleLocalMatrix(l, 0);
309     //for (auto &it : futures)
310         //it.get();
311 }
312
313 template<class Problem, class Grid, class Matrix>
314 const int DGMethod<Problem, Grid, Matrix>::AssembleIDUDVMatrix(const int l)
315 {
316     int i, j, k, nodes;
317     double mij;
318     const auto& elem{ m_Grid->GetElement(l) };
319     const int dofs{ (int)elem->GetDoFs() };
320     const int terms{ (int)m_problem->getNumberOfTerms() };
321     nodes = elem->GetNumberOfNodes();

```



```

322         std::vector<Mesh::Point> points(nodes);
323         for (i = 0; i < nodes; ++i)
324             points[i] = m_Grid->GetNode(elem->GetNode(i));
325         int sizei = 0, sizej = 0;
326         for (i = 0; i < (int)dofs; ++i)
327         {
328             for (j = 0; j < (int)dofs; ++j)
329             {
330                 auto M = [&](const Mesh::Point& p)
331                 {
332                     //auto m = elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
333                     return m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(), p) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
334                 };
335                 //mij = m_Grid->getParameter(Parameters::DIFFUSION, 1, j) * elem->Integrate(M,
points);
336                 mij = elem->Integrate(M, points);
337                 //m_GlobalMatrix->AddElement(inode, jnode, mij);
338                 m_GlobalMatrix->AddElement(m_nums[1] + i, m_nums[1] + j, mij);
339             }
340         }
341         return 0;
342     }
343
344     template<class Problem, class Grid, class Matrix>
345     const int DGMethod<Problem, Grid, Matrix>::AssembleIDUVMMatrix(const int l)
346     {
347         int i, j, k, nodes;
348         double mij;
349         const auto& elem{ m_Grid->GetElement(l) };
350         const int dofs{ (int)elem->GetDoFs() };
351         const int terms{ (int)m_problem->getNumberOfTerms() };
352         nodes = elem->GetNumberOfNodes();
353         std::vector<Mesh::Point> points(nodes);
354         for (i = 0; i < nodes; ++i)
355             points[i] = m_Grid->GetNode(elem->GetNode(i));
356         int sizei = 0, sizej = 0;
357         for (i = 0; i < (int)dofs; ++i)
358         {
359             for (j = 0; j < (int)dofs; ++j)
360             {
361                 auto M = [&](const Mesh::Point& p)
362                 {
363                     return m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
364                 };
365                 auto _mij = elem->Integrate(M, points);
366                 //m_GlobalMatrix->AddElement(inode, jnode, _mij);
367                 m_GlobalMatrix->AddElement(m_nums[1] + i, m_nums[1] + j, _mij);
368             }
369         }
370         return 0;
371     }
372
373     template<class Problem, class Grid, class Matrix>
374     const int DGMethod<Problem, Grid, Matrix>::AssembleIUDVMMatrix(const int l)
375     {
376         int i, j, k, nodes;
377         double mij;
378         const auto& elem{ m_Grid->GetElement(l) };
379         const int dofs{ (int)elem->GetDoFs() };
380         const int terms{ (int)m_problem->getNumberOfTerms() };
381         nodes = elem->GetNumberOfNodes();
382         std::vector<Mesh::Point> points(nodes);
383         for (i = 0; i < nodes; ++i)
384             points[i] = m_Grid->GetNode(elem->GetNode(i));
385         int sizei = 0, sizej = 0;
386         for (i = 0; i < dofs; ++i)
387         {
388             for (j = 0; j < dofs; ++j)
389             {
390                 auto M = [&](const Mesh::Point& p)
391                 {
392                     return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
393                 };
394                 //mij = m_CoarseGrid->getParameter(Parameters::ADVECTION, 1, j) *
m_flux(m_CoarseGrid->getSolution(1, j)) * elem->Integrate(M, points).x;
395                 mij = elem->Integrate(M, points).x;
396                 //m_GlobalMatrix->AddElement(inode, jnode, mij);
397                 m_GlobalMatrix->AddElement(m_nums[1] + i, m_nums[1] + j, mij);
398             }
399         }
400         return 0;
401     }
402
403     template<class Problem, class Grid, class Matrix>

```

```

405     const int DGMethod<Problem, Grid, Matrix>::AssembleRUVMatrix(const int l)
406     {
407         int i, j, k, nodes;
408         double mij;
409         const auto& elem{ m_Grid->GetElement(l) };
410         const int dofs{ (int)elem->GetDoFs() };
411         const int terms{ (int)m_problem->getNumberOfTerms() };
412         nodes = elem->GetNumberOfNodes();
413         std::vector<Mesh::Point> points(nodes);
414         for (i = 0; i < nodes; ++i)
415             points[i] = m_Grid->GetNode(elem->GetNode(i));
416         int sizei = 0, sizej = 0;
417         for (i = 0; i < (int)dofs; ++i)
418         {
419             for (j = 0; j < (int)dofs; ++j)
420             {
421                 auto M = [&](const Mesh::Point& p)
422                 {
423                     double vel = sqrt(m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p,
0) * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0));
424                     double h = elem->GetMeasure();
425                     double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, l,
elem->GetType(), p);
426                     double tau = 0.;
427                     //double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, l,
elem->GetType(), p);
428
429                     if (Pe >= 1)
430                         tau = h / 2. / vel;
431                     else
432                         tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, l,
elem->GetType(), p);
433                     auto supg = tau * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p,
0) * elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p) * elem->GetShapeFunction(i, p);
434                     return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p); // + supg;
435                 };
436                 mij = elem->Integrate(M, points);
437
438                 m_RightMatrix->AddElement(m_nums[l] + i, m_nums[l] + j, mij);
439             }
440         }
441         return 0;
442     }
443
444     template<class Problem, class Grid, class Matrix>
445     const int DGMethod<Problem, Grid, Matrix>::AssembleSUPGMatrix(const int l)
446     {
447         int i, j, k, nodes;
448         double mij;
449         const auto& elem{ m_Grid->GetElement(l) };
450         const int dofs{ (int)elem->GetDoFs() };
451         const int terms{ (int)m_problem->getNumberOfTerms() };
452         nodes = elem->GetNumberOfNodes();
453         std::vector<Mesh::Point> points(nodes);
454         for (i = 0; i < nodes; ++i)
455             points[i] = m_Grid->GetNode(elem->GetNode(i));
456         for (i = 0; i < (int)dofs; ++i)
457         {
458             for (j = 0; j < (int)dofs; ++j)
459             {
460                 auto M = [&](const Mesh::Point& p)
461                 {
462                     double vel = sqrt(m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p,
0) * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0));
463                     double h = elem->GetMeasure();
464                     //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, l,
elem->GetType(), p);
465                     double tau = 0.;
466                     double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, l,
elem->GetType(), p);
467                     //double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. /
Pe);
468                     //double beta = h / std::sqrt(3.) * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) -
1. / Pe);
469                     //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. / Pe);
470                     //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. / Pe);
471                     //beta = 0.;
472                     //for (int ii = 0; ii < (int)dofs; ++ii)
473                         //beta += m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(ii, p);
474                     //return beta * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0)
* m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0) *
475                         elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
476                     if (Pe >= 1)
477                         tau = h / 2. / vel;
478                     else
479                         tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, l,

```

```

elem->GetType(), p);
480 //return 0.;
481 return tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
482 elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
483 };
484
485 //double tau =
486 auto _mij = elem->Integrate(M, points);
487 m_GlobalMatrix->AddElement(m_nums[l] + i, m_nums[l] + j, _mij);
488 }
489 }
490 return 0;
491 }
492
493
494 template<class Problem, class Grid, class Matrix>
495 const int DGMethod<Problem, Grid, Matrix>::AssembleInter()
496 {
497     const auto mu = 1e6;
498     for (int l = 0; l < m_Ns; ++l)
499     {
500         const auto& bound{ m_Grid->GetBoundary(l) };
501         const auto& nk{ bound->GetNeighbour(0) };
502         const auto& ne{ bound->GetNeighbour(1) };
503         const auto& elemk{ m_Grid->GetElement(nk) };
504         const auto& dofs{ bound->GetDoFs() };
505         const auto& dofsk{ elemk->GetDoFs() };
506         std::vector<Mesh::Point> points(dofs);
507         for (int i = 0; i < dofs; ++i)
508         {
509             points[i] = m_Grid->GetNode(bound->GetNode(i));
510         }
511         if (ne < 0)
512             continue;
513         const auto& eleme{ m_Grid->GetElement(ne) };
514         for (int i = 0; i < dofsk; ++i)
515         {
516             for (int j = 0; j < dofsk; ++j)
517             {
518                 auto Tkk = [&](const Mesh::Point& p)
519                 {
520                     auto kappa = m_problem->get_parameter(Terms::IDUDV, 1, elemk->GetType(), p);
521                     auto val1 = bound->GetNormal() * elemk->GetShapeFunction(j, p) *
elemk->GetGradShapeFunction(i, p);
522                     auto val2 = bound->GetNormal() * elemk->GetShapeFunction(i, p) *
elemk->GetGradShapeFunction(j, p);
523
524                     auto ip = bound->GetNormal() * bound->GetNormal() *
elemk->GetShapeFunction(j, p) * elemk->GetShapeFunction(i, p);
525                     return 0.5 * kappa * (val2 - val1) + mu * ip;
526                 };
527                 auto mj = bound->Integrate(Tkk, points);
528                 std::cout << mj << std::endl;
529                 m_GlobalMatrix->AddElement(m_nums[nk] + i, m_nums[nk] + j, mj);
530             }
531         }
532
533         for (int i = 0; i < dofsk; ++i)
534         {
535             for (int j = 0; j < dofsk; ++j)
536             {
537                 auto Tkk = [&](const Mesh::Point& p)
538                 {
539                     auto kappa = m_problem->get_parameter(Terms::IDUDV, 1, eleme->GetType(), p);
540                     auto val1 = bound->GetNormal() * eleme->GetShapeFunction(j, p) *
elemk->GetGradShapeFunction(i, p);
541                     auto val2 = bound->GetNormal() * elemk->GetShapeFunction(i, p) *
eleme->GetGradShapeFunction(j, p);
542
543                     auto ip = bound->GetNormal() * bound->GetNormal() *
eleme->GetShapeFunction(j, p) * elemk->GetShapeFunction(i, p);
544                     return 0.5 * kappa * (val2 + val1) + mu * ip;
545                 };
546                 auto mj = bound->Integrate(Tkk, points);
547                 m_GlobalMatrix->AddElement(m_nums[nk] + i, m_nums[ne] + j, mj);
548             }
549         }
550
551
552         for (int i = 0; i < dofsk; ++i)
553         {
554             for (int j = 0; j < dofsk; ++j)
555             {
556                 auto Tkk = [&](const Mesh::Point& p)
557                 {
558                     auto kappa = m_problem->get_parameter(Terms::IDUDV, 1, eleme->GetType(), p);

```

```

559         auto val1 = bound->GetNormal() * eleme->GetShapeFunction(j, p) *
eleme->GetGradShapeFunction(i, p);
560         auto val2 = bound->GetNormal() * eleme->GetShapeFunction(i, p) *
eleme->GetGradShapeFunction(j, p);
561
562         auto ip = bound->GetNormal() * bound->GetNormal() *
eleme->GetShapeFunction(j, p) * eleme->GetShapeFunction(i, p);
563         return 0.5 * kappa * (val2 - val1) + mu * ip;
564     };
565     auto mj = bound->Integrate(Tkk, points);
566     m_GlobalMatrix->AddElement(m_nums[ne] + i, m_nums[ne] + j, mj);
567     }
568 }
569
570 for (int i = 0; i < dofsk; ++i)
571 {
572     for (int j = 0; j < dofsk; ++j)
573     {
574         auto Tkk = [&](const Mesh::Point& p)
575         {
576             auto kappa = m_problem->get_parameter(Terms::IDUDV, 1, elemk->GetType(), p);
577             auto val1 = bound->GetNormal() * elemk->GetShapeFunction(j, p) *
eleme->GetGradShapeFunction(i, p);
578             auto val2 = bound->GetNormal() * eleme->GetShapeFunction(i, p) *
elemk->GetGradShapeFunction(j, p);
579
580             auto ip = bound->GetNormal() * bound->GetNormal() *
elemk->GetShapeFunction(j, p) * eleme->GetShapeFunction(i, p);
581             return 0.5 * kappa * (val2 + val1) + mu * ip;
582         };
583         auto mj = bound->Integrate(Tkk, points);
584         m_GlobalMatrix->AddElement(m_nums[ne] + i, m_nums[nk] + j, mj);
585     }
586 }
587 }
588 return 0;
589 }
590
591 template<class Problem, class Grid, class Matrix>
592 const int DGMethod<Problem, Grid, Matrix>::AssembleLocalMatrix(const int l, const int old)
593 {
594     int i, j, k, nodes;
595     double mij;
596     const auto& elem{ m_Grid->GetElement(l) };
597     const int dofs{ (int)elem->GetDoFs() };
598     const int terms{ (int)m_problem->getNumberOfTerms() };
599     nodes = elem->GetNumberOfNodes();
600     std::vector<Mesh::Point> points(nodes);
601     for (i = 0; i < nodes; ++i)
602         points[i] = m_Grid->GetNode(elem->GetNode(i));
603     for (k = 0; k < terms; ++k)
604     {
605         switch (m_problem->getTerm(k))
606         {
607             case Terms::IUV:
608                 for (i = 0; i < (int)dofs; ++i)
609                 {
610                     for (j = 0; j < (int)dofs; ++j)
611                     {
612                         auto M = [&](const Mesh::Point& p)
613                         {
614                             return m_problem->get_parameter(Terms::IUV, 1, elem->GetType(), p) *
elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
615                         };
616                         mij = elem->Integrate(M, points);
617                         auto inode = m_Grid->interpolate(elem->GetNode(i));
618                         auto jnode = m_Grid->interpolate(elem->GetNode(j));
619                         if (inode > -1 && jnode > -1)
620                             m_GlobalMatrix->AddElement(inode, jnode, mij);
621                     }
622                 }
623                 break;
624             case Terms::IDUDV:
625                 for (i = 0; i < (int)dofs; ++i)
626                 {
627                     for (j = 0; j < (int)dofs; ++j)
628                     {
629                         auto inode = m_Grid->interpolate(elem->GetNode(i));
630                         auto jnode = m_Grid->interpolate(elem->GetNode(j));
631                         if (inode == -1 || jnode == -1)
632                             continue;
633                         auto M = [&](const Mesh::Point& p)
634                         {
635                             //auto m = elem->GetGradShapeFunction(i, p) *
elem->GetGradShapeFunction(j, p);
636                             return m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(), p) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);

```

```

637         };
638         //mij = m_Grid->getParameter(Parameters::DIFFUSION, 1, j) *
elem->Integrate(M, points);
639         mij = elem->Integrate(M, points);
640         m_GlobalMatrix->AddElement(inode, jnode, mij);
641     }
642     }
643     break;
644     case Terms::IDUV:
645         for (i = 0; i < (int)dofs; ++i)
646         {
647             for (j = 0; j < (int)dofs; ++j)
648             {
649                 auto inode = m_Grid->interpolate(elem->GetNode(i));
650                 auto jnode = m_Grid->interpolate(elem->GetNode(j));
651                 if (inode == -1 || jnode == -1)
652                     continue;
653                 auto M = [&](const Mesh::Point& p)
654                 {
655                     return m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
656                 };
657                 auto _mij = elem->Integrate(M, points);
658                 m_GlobalMatrix->AddElement(inode, jnode, _mij);
659             }
660         }
661     break;
662     case Terms::IUDV:
663         for (i = 0; i < dofs; ++i)
664         {
665             for (j = 0; j < dofs; ++j)
666             {
667                 auto inode = m_Grid->interpolate(elem->GetNode(i));
668                 auto jnode = m_Grid->interpolate(elem->GetNode(j));
669                 if (inode == -1 || jnode == -1)
670                     continue;
671                 auto M = [&](const Mesh::Point& p)
672                 {
673                     return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
674                 };
675                 //mij = m_CoarseGrid->getParameter(Parameters::ADVECTION, 1, j) *
m_flux(m_CoarseGrid->getSolution(1, j)) * elem->Integrate(M, points).x;
676                 mij = elem->Integrate(M, points).x;
677                 m_GlobalMatrix->AddElement(inode, jnode, mij);
678             }
679         }
680     break;
681     case Terms::EUV:
682         for (i = 0; i < dofs; ++i)
683         {
684             for (j = 0; j < dofs; ++j)
685             {
686                 auto M = [&](const Mesh::Point& p)
687                 {
688                     return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
689                 };
690                 mij = elem->Integrate(M, points);
691                 m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::MASS, 1, j) * m_Grid->getSolution(1, j) * mij;
692                 //m_rhsvector->operator[] (m_nums[1] + i) +=
m_CoarseGrid->getParameter(Parameters::MASS, 1, points[j]) * elem->GetValue(j) * mij;
693             }
694         }
695     break;
696     case Terms::EDUDV:
697         for (i = 0; i < dofs; ++i)
698         {
699             for (j = 0; j < dofs; ++j)
700             {
701                 auto M = [&](const Mesh::Point& p)
702                 {
703                     return elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
704                 };
705                 mij = elem->Integrate(M, points);
706                 m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::DIFFUSION, 1, j) * m_Grid->getSolution(1, j) * mij;
707             }
708         }
709     break;
710     case Terms::EDUV:
711         for (i = 0; i < dofs; ++i)
712         {
713             for (j = 0; j < dofs; ++j)
714             {
715                 auto M = [&](const Mesh::Point& p)
716                 {

```

```

717         return elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
718     };
719     mij = elem->Integrate(M, points).x;
720     m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, 1, j) * mij;
721     }
722     }
723     break;
724     case Terms::EUDV:
725     for (i = 0; i < dofs; ++i)
726     {
727         for (j = 0; j < dofs; ++j)
728         {
729             auto M = [&](const Mesh::Point& p)
730             {
731                 return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
732             };
733             mij = elem->Integrate(M, points).x;
734             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, 1, j) * mij; // *mij;
735         }
736     }
737     break;
738     case Terms::EFV:
739     for (i = 0; i < dofs; ++i)
740     {
741         /*for (j = 0; j < dofs; ++j)
742         {
743             auto M = [&](const Mesh::Point& p)
744             {
745                 return m_problem->get_parameter(Terms::EFV, elem->GetType(), 1, j, p) *
elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
746             };
747             mij = elem->Integrate(M, points);
748             m_rhsvector->operator[] (elem->GetNode(i)) += mij;
749             */
750             auto M = [&](const Mesh::Point& p)
751             {
752                 return m_problem->get_parameter(Terms::EFV, elem->GetType(), 1, i, p) *
elem->GetShapeFunction(i, p);
753             };
754             mij = elem->Integrate(M, points);
755             m_rhsvector->operator[] (elem->GetNode(i)) += mij;
756         }
757     }
758     break;
759     case Terms::RUV:
760     for (i = 0; i < (int)dofs; ++i)
761     {
762         for (j = 0; j < (int)dofs; ++j)
763         {
764             auto M = [&](const Mesh::Point& p)
765             {
766                 return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
767             };
768             mij = elem->Integrate(M, points);
769             auto inode = m_Grid->interpolate(elem->GetNode(i));
770             auto jnode = m_Grid->interpolate(elem->GetNode(j));
771             if (inode > -1 && jnode > -1)
772                 m_RightMatrix->AddElement(inode, jnode, mij);
773         }
774     }
775     break;
776     case Terms::SUPG:
777     for (i = 0; i < (int)dofs; ++i)
778     {
779         for (j = 0; j < (int)dofs; ++j)
780         {
781             auto inode = m_Grid->interpolate(elem->GetNode(i));
782             auto jnode = m_Grid->interpolate(elem->GetNode(j));
783             if (inode == -1 || jnode == -1)
784                 continue;
785             auto M = [&](const Mesh::Point& p)
786             {
787                 double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
788                 double h = elem->GetMeasure();
789                 //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
790                 double tau = 0.;
791                 double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
792                 //double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) -
1.) - 1. / Pe);
793                 double beta = h / std::sqrt(3.) * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) -
1.) - 1. / Pe);
794                 //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1.
/ Pe);

```

```

794                                     //beta = 0.;
795                                     //for (int ii = 0; ii < (int)dofs; ++ii)
796                                     //beta += m_problem->get_parameter(Terms::IDUV, l, elem->GetType(),
p, 0) * elem->GetGradShapeFunction(ii, p);
797                                     return beta * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0) *
798                                     elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
799                                     if (Pe >= 1)
800                                         tau = h / 2. / vel;
801                                     else
802                                         tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, l,
elem->GetType(), p);
803                                     //return 0.;
804                                     return tau * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0) *
805                                     elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
806                                     };
807
808                                     //double tau =
809                                     auto _mij = elem->Integrate(M, points);
810                                     m_GlobalMatrix->AddElement(inode, jnode, _mij);
811                                 }
812                             }
813                             break;
814                         default:
815                             break;
816                     }
817                 }
818                 return 0;
819             }
820             template<class Problem, class Grid, class Matrix>
821             void DGMethod<Problem, Grid, Matrix>::MainConditions()
822             {
823                 double mu{ 1e8 };
824                 const auto n = m_problem->get_number_of_boundaries();
825                 const auto m = m_Grid->GetNumberOfBoundaries();
826                 for (int i = 0; i < n; ++i)
827                 {
828                     const auto& type = m_problem->get_boundary_type(i);
829                     for (int j = 0; j < m; ++j)
830                     {
831                         const auto& row = m_Grid->GetBoundary(j);
832                         if (row->GetType() == type)
833                         {
834                             const int dofs = (int)row->GetDoFs();
835                             const int dofs2 = 2;
836                             const auto& elem_num = row->GetNeighbour(0);
837                             const auto& elem = m_Grid->GetElement(elem_num);
838                             const int dofs_elem = elem->GetDoFs();
839                             std::vector<Mesh::Point> points(dofs_elem);
840                             std::vector<Mesh::Point> bpoints(dofs);
841                             for (int k = 0; k < dofs_elem; ++k)
842                                 points[k] = m_Grid->GetNode(elem->GetNode(k));
843                             for (int k = 0; k < dofs; ++k)
844                                 bpoints[k] = m_Grid->GetNode(row->GetNode(k));
845                             for (int ii = 0; ii < dofs_elem; ++ii)
846                             {
847                                 for (int jj = 0; jj < dofs_elem; ++jj)
848                                 {
849                                     auto M = [&](const Mesh::Point& p)
850                                     {
851                                         return elem->GetShapeFunction(ii, p) * elem->GetShapeFunction(jj,
p); // + supg;
852                                     };
853                                     auto mij = mu * row->Integrate(M, bpoints);
854                                     m_GlobalMatrix->AddElement(m_nums[elem_num] + ii, m_nums[elem_num] + jj,
mij);
855                                 }
856                                 auto MM = [&](const Mesh::Point& p)
857                                 {
858                                     return elem->GetWeight(elem_num, points, [=](const Mesh::Point& p) {
return m_problem->get_boundary_parameter(0, type, p); }) * elem->GetShapeFunction(ii, p);
859                                 };
860                                 auto mij = row->Integrate(MM, points);
861                                 std::cout << mij << std::endl;
862                                 m_rhsvector->operator[](m_nums[elem_num] + ii) += mij;
863                             }
864                             /*for (int k = 0; k < dofs; ++k)
865                             {
866                                 int l = 0;
867                                 for (; l < dofs_elem; ++l)
868                                 {
869                                     if (elem->GetNode(l) == row->GetNode(k))
870                                         break;
871                                 }

```

```

872
873         m_GlobalMatrix->NullRow(row->GetNode(k));
874         //m_GlobalMatrix->operator()(row->GetNode(k), row->GetNode(k)) *= mu;
875         //m_rhsvector->operator[](row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, m_Grid->GetNode(row->GetNode(k)));
876         //m_rhsvector->operator[](row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, elem_num, 1, m_Grid->GetNode(row->GetNode(k)));
877         m_rhsvector->operator[](row->GetNode(k)) = elem->GetWeight(1, points,
[=](const Mesh::Point& p) { return m_problem->get_boundary_parameter(0, type, p); });
878         if(m_problem->findTerm(Terms::RUV))
879         {
880             m_RightMatrix->NullRow(row->GetNode(k));
881             //m_RightMatrix->operator()(row->GetNode(k), row->GetNode(k)) *= mu;
882         }
883     }*/
884     /*for (int k = dofs2; k < dofs; ++k)
885     {
886         m_GlobalMatrix->NullRow(row->GetNode(k));
887         m_rhsvector->operator[](row->GetNode(k)) = 0;
888     }*/
889     }
890 }
891 }
892 /*for (auto bnd : m_Grid->GetBoundaryConditions())
893 {
894     if (get<0>(bnd.second) == 1)
895         for (auto row : m_Grid->GetBoundary())
896         {
897             if (bnd.first == row->GetType())
898             {
899                 for (int i = 0; i < row->GetDoF(); ++i)
900                 {
901                     m_GlobalMatrix->NullRow(row->GetNodes(i));
902                     m_rhsvector[row->GetNodes(i)] =
get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
903                 }
904             }
905         }
906     }*/
907 }
908 template<class Problem, class Grid, class Matrix>
909 void DGMMethod<Problem, Grid, Matrix>::SecondConditions()
910 {
911     double theta = 0;
912     int nfem;
913     Mesh::Point temp[3];
914     std::vector<int> local;
915     for (auto bnd : m_Grid->GetBoundaryConditions())
916     {
917         //if (get<0>(bnd.second) == 2)
918         {
919             for (auto row : m_Grid->GetBoundary())
920             {
921                 if (bnd.first == row->GetType())
922                 {
923                     local.resize(0);
924                     int dofs = row->GetDoF();
925                     nfem = row->GetNumberOfElement(0);
926                     auto elem = m_Grid->GetElements()[nfem];
927                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
928                     for (int j = 0; j < dofs; ++j)
929                     {
930                         temp[j] = m_Grid->GetNodes() [row->GetNodes(j)];
931                         for (int i = 0; i < elem->GetDoF(); ++i)
932                         {
933                             if (row->GetNodes(j) == elem->GetNodes() [i])
934                             {
935                                 local.push_back(i);
936                                 break;
937                             }
938                         }
939                     }
940                     for (int i = 0; i < dofs; ++i)
941                     {
942                         for (int j = 0; j < dofs; ++j)
943                         {
944                             //theta = get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
945                             theta = 0;
946                             auto GetMass = [&](const Mesh::Point& p) {return
elem->GetBasis(local[j], p) * elem->GetBasis(local[i], p); };
947                             auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
948                             //if (i < 2 || j < 2)
949                             m_rhsvector[row->GetNodes(i)] += theta * row->Integrate(GetMass,
temp);
950
951                             //if (i < 3 || j < 3)

```



```

952                                     // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
953                                     }
954                                     }
955                                     }
956                                     }
957                                     }
958                                     }
959                                     }
960 template<class Problem, class Grid, class Matrix>
961 void DGMethod<Problem, Grid, Matrix>::StefanConditions()
962 {
963     double dest{ 0. }, lat{ 0 };
964     int nfem;
965     Mesh::Point temp[3];
966     std::vector<int> local;
967     for (auto bnd : m_Grid->GetBoundaryConditions())
968     {
969         //if (get<0>(bnd.second) == 4)
970         {
971             lat = 0;
972             //lat = get<2>(bnd.second);
973             for (auto row : m_Grid->GetBoundary())
974             {
975                 if (bnd.first == row->GetType())
976                 {
977                     local.resize(0);
978                     int dofs = row->GetDoF();
979                     nfem = row->GetNumberOfElement(0);
980                     auto elem = m_Grid->GetElements()[nfem];
981                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
982                     for (int j = 0; j < dofs; ++j)
983                     {
984                         temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
985                         for (int i = 0; i < elem->GetDoF(); ++i)
986                         {
987                             if (row->GetNodes(j) == elem->GetNodes()[i])
988                             {
989                                 local.push_back(i);
990                                 break;
991                             }
992                         }
993                     }
994                     for (int i = 0; i < dofs; ++i)
995                     {
996                         for (int j = 0; j < dofs; ++j)
997                         {
998                             dest = 0;
999                             //dest = get<1>(bnd.second) (m_Grid->GetNodes()[row->GetNodes(i)]);
1000                             auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
1001                             //if (i < 2 || j < 2)
1002                             m_rhsvector[row->GetNodes(i)] += dest * lat *
row->Integrate(GetBBasis, temp);
1003                             //if (i < 3 || j < 3)
1004                             // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
1005                             }
1006                         }
1007                     }
1008                 }
1009             }
1010         }
1011     }
1012 }
1013 template<class Problem, class Grid, class Matrix>
1014 void DGMethod<Problem, Grid, Matrix>::ThirdConditions()
1015 {
1016     double param{ 0 }, beta{ 0 };
1017     int nfem;
1018     Mesh::Point temp[6];
1019     std::vector<int> local;
1020     auto fxy = [&](const Mesh::Point& p) {return (10 * p.y*m_time + m_time) / 10; };
1021     //auto fxy = [&](const Point& p){return 10 * p.y + 10 * m_time; };
1022     for (auto bnd : m_Grid->GetBoundaryConditions())
1023     {
1024         //if (get<0>(bnd.second) == 3)
1025         {
1026             for (auto row : m_Grid->GetBoundary())
1027             {
1028                 if (bnd.first == row->GetType())
1029                 {
1030                     local.resize(0);
1031                     int dofs = row->GetDoF();
1032                     nfem = row->GetNumberOfElement(0);
1033                     auto elem = m_Grid->GetElements()[nfem];

```

```

1035         //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
1036         auto order = elem->GetDoF();
1037         for (int j = 0; j < dofs; ++j)
1038         {
1039             temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
1040             for (int i = 0; i < order; ++i)
1041             {
1042                 if (row->GetNodes(j) == elem->GetNodes()[i])
1043                 {
1044                     local.push_back(i);
1045                     break;
1046                 }
1047             }
1048         }
1049         double val{ 0 };
1050         for (int i = 0; i < dofs; ++i)
1051         {
1052             for (int j = 0; j < dofs; ++j)
1053             {
1054                 param = 0;
1055                 beta = 0;
1056                 //beta = get<2>(bnd.second);
1057                 //param = get<1>(bnd.second)(m_Grid->GetNodes()[row->GetNodes(i)]);
1058                 //param = fxy(temp[j]);
1059                 auto GetBBasis = [&](const Mesh::Point& p) {return
elem->GetBasis(local[j], p)*elem->GetBasis(local[i], p); };
1060                 //val = row->GetElement(GetBBasis, temp);
1061                 val = row->Integrate(GetBBasis, temp);
1062                 m_GlobalMatrix->operator()(row->GetNodes(i), row->GetNodes(j)) +=
beta * val;
1063                 m_rhsvector[row->GetNodes(i)] += beta * param * val;
1064             }
1065         }
1066     }
1067 }
1068 }
1069 }
1070 }
1071 template<class Problem, class Grid, class Matrix>
1072 Matrix* DGMMethod<Problem, Grid, Matrix>::GetGlobalMatrix() const
1073 {
1074     return m_GlobalMatrix;
1075 }
1076 template<class Problem, class Grid, class Matrix>
1077 const double DGMMethod<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p) const
1078 {
1079     if (!m_solution.size())
1080         return -1;
1081     double val = 0;
1082     int nfem = -1;
1083     nfem = m_Grid->FindElement(p);
1084     if (nfem == -1)
1085         return -1;
1086     auto elem = m_Grid->GetElements()[nfem];
1087     for (int i = 0; i < elem->GetDoF(); ++i)
1088         val += m_solution[elem->GetNodes()[i]] * elem->GetBasis(i, p);
1089     return val;
1090 }
1091 template<class Problem, class Grid, class Matrix>
1092 const double DGMMethod<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
std::vector<double>& vec) const
1093 {
1094     if (!vec.size())
1095         return -1;
1096     double val{ 0 };
1097     int nfem{ -1 };
1098     nfem = m_Grid->FindElement(p);
1099     if (nfem == -1)
1100         return -1;
1101     auto elem = m_Grid->GetElements()[nfem];
1102     for (int i = 0; i < elem->GetDoFs(); ++i)
1103         val += vec[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1104     return val;
1105 }
1106 template<class Problem, class Grid, class Matrix>
1107 const double DGMMethod<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
std::vector<double>& vec, const int num) const
1108 {
1109     if (!vec.size() || num < 0)
1110         return -1;
1111     double val{ 0 };
1112     auto elem = m_Grid->GetElements()[num];
1113     for (int i = 0; i < elem->GetDoF(); ++i)
1114         val += vec[elem->GetNodes()[i]] * elem->GetBasis(i, p);
1115     return val;
1116 }
1117 //template<class Problem, class Grid, class Matrix>

```

```

1118         //const Mesh::Point DGMethod<Problem, Grid, Matrix>::GetGradValue(const Mesh::Point& p, const
1119         std::vector<double>& vec) const
1120         //{
1121         //    Mesh::Point val{ 0, 0 };
1122         //    int nfem{ -1 };
1123         //    nfem = m_Grid->FindElement(p);
1124         //    if (nfem == -1)
1125         //        return val;
1126         //    auto elem = m_Grid->GetElements()[nfem];
1127         //    for (int i = 0; i < elem->GetDoF(); ++i)
1128         //    {
1129         //        val.x += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).x;
1130         //        val.y += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).y;
1131         //        val.z += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).z;
1132         //    }
1133         //    return val;
1134         //}
1135         template<class Problem, class Grid, class Matrix>
1136         const double DGMethod<Problem, Grid, Matrix>::GetEffective(const std::vector<double>& vec)
1137         const
1138         {
1139             double sum = 0;
1140             //std::vector<int> dofs;
1141             //Mesh::Point points[10];
1142             //for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1143             //{
1144                 //auto mb = [&](const Mesh::Point& b) {return GetGradValue(b, vec)*GetGradValue(b,
1145                 vec); };
1146                 //dofs.resize(0);
1147                 //auto elem = m_Grid->GetElements()[i];
1148                 //int order = elem->GetDoF();
1149                 //double diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
1150                 //for (int j = 0; j < order; ++j)
1151                 //{
1152                     //dofs.push_back(elem->GetNodes()[j]);
1153                     //points[j] = m_Grid->GetNodes()[dofs[j]];
1154                 //}
1155                 //sum += diff * elem->Integrate(mb, points);
1156             //}
1157             //std::cout << "Effect (local): " << sum << std::endl;
1158             //std::cout << "Effect (local) sqrt: " << sqrt(sum) << std::endl;
1159             return sum;
1160         }
1161         //template<class Problem, class Grid, class Matrix>
1162         //const Mesh::Point DGMethod<Problem, Grid, Matrix>::GetLambdaGrad(const Mesh::Point& p, const
1163         std::vector<double>& vec) const
1164         //{
1165         //    Mesh::Point val{ 0, 0, 0 };
1166         //    //double val{ 0 };
1167         //    double diff{ 0 };
1168         //    Mesh::Point temp{ 0, 0, 0 };
1169         //    int nfem{ -1 };
1170         //    nfem = m_Grid->FindElement(p);
1171         //    if (nfem == -1)
1172         //        return val;
1173         //    auto elem = m_Grid->GetElements()[nfem];
1174         //    diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
1175         //    for (int i = 0; i < elem->GetDoF(); ++i)
1176         //    {
1177         //        //val += elem->GetGradBasis(i, p) * elem->GetGradBasis(i, p) * vec[elem->GetNodes()[i]]
1178         //        * vec[elem->GetNodes()[i]] * diff;
1179         //        //val += elem->GetBasis(i, p) * vec[elem->GetNodes()[i]] * diff;
1180         //        temp = elem->GetGradBasis(i, p);
1181         //        val.x += temp.x * vec[elem->GetNodes()[i]] * (diff);
1182         //        val.y += temp.y * vec[elem->GetNodes()[i]] * (diff);
1183         //        val.z += temp.z * vec[elem->GetNodes()[i]] * (diff);
1184         //    }
1185         //    return val;
1186         //}
1187         template<class Problem, class Grid, class Matrix>
1188         const std::vector<double> DGMethod<Problem, Grid, Matrix>::GetRightVector() const
1189         {
1190             return *m_rhsvector;
1191         }
1192         template<class Problem, class Grid, class Matrix>
1193         void DGMethod<Problem, Grid, Matrix>::OutDatFormat(const Mesh::Point& mn, const Mesh::Point&
1194         mx, const std::string& file_name, const std::vector<double>& vec) const
1195         {
1196             std::ofstream of(file_name + ".dat");
1197             std::streambuf *buf = std::cout.rdbuf();
1198             std::cout.rdbuf(of.rdbuf());
1199             std::cout << "TITLE = FE-METHOD\n";
1200             std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1201             std::cout << "ZONE i=51, j=51, F=POINT\n";
1202             double stepx = (mx.x - mn.x) / 51;
1203             double stepy = (mx.y - mn.y) / 51;
1204             for (int i = 0; i < 51; ++i)

```

```

1199         for (int j = 0; j < 51; ++j)
1200             std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.x + j * stepx, mn.y + i * stepy, mn.z), vec) << std::endl;
1201         std::cout.rdbuf(buf);
1202         of.close();
1203         of.open(file_name + "x.dat");
1204         buf = std::cout.rdbuf();
1205         std::cout.rdbuf(of.rdbuf());
1206         std::cout << "TITLE = FE-METHOD\n";
1207         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1208         std::cout << "ZONE i=51, j=51, F=POINT\n";
1209         for (int i = 0; i < 51; ++i)
1210             for (int j = 0; j < 51; ++j)
1211                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.z, mn.x + j * stepx, mn.y + i * stepy), vec) << std::endl;
1212         std::cout.rdbuf(buf);
1213         of.close();
1214         of.open(file_name + "y.dat");
1215         buf = std::cout.rdbuf();
1216         std::cout.rdbuf(of.rdbuf());
1217         std::cout << "TITLE = FE-METHOD\n";
1218         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1219         std::cout << "ZONE i=51, j=51, F=POINT\n";
1220         for (int i = 0; i < 51; ++i)
1221             for (int j = 0; j < 51; ++j)
1222                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.x + j * stepx, mn.z, mn.y + i * stepy), vec) << std::endl;
1223         std::cout.rdbuf(buf);
1224         of.close();
1225     }
1226     template<class Problem, class Grid, class Matrix>
1227     void DGMMethod<Problem, Grid, Matrix>::ApplySources()
1228     {
1229         int nfem = -1;
1230         auto total = m_problem->get_total_sources();
1231         for (int i = 0; i < total; ++i)
1232         {
1233             auto src = m_problem->get_point_source(i);
1234             auto point = src.get_point();
1235             nfem = m_Grid->FindElement(point);
1236             if (nfem != -1)
1237             {
1238                 auto val = src.get_value();
1239                 auto elem = m_Grid->GetElement(nfem);
1240                 for (int j = 0; j < 3; ++j)
1241                     m_rhsvector->operator[] (elem->GetNode(j)) += val * elem->GetShapeFunction(j,
point);
1242             }
1243             nfem = -1;
1244         }
1245         /*for (auto srd : m_Grid->GetDottedSources())
1246         {
1247             nfem = m_Grid->FindElement(srd.first);
1248             if (nfem != -1)
1249             {
1250                 auto elem = m_Grid->GetElements() [nfem];
1251                 for (int i = 0; i < elem->GetDoF(); ++i)
1252                 {
1253                     m_rhsvector[elem->GetNodes() [i]] += srd.second * elem->GetBasis(i, srd.first);
1254                 }
1255             }
1256             nfem = -1;
1257         }*/
1258     }
1259     template<class Problem, class Grid, class Matrix>
1260     void DGMMethod<Problem, Grid, Matrix>::Rediscretization(const std::shared_ptr<Grid>& grid)
1261     {
1262         m_GlobalMatrix->NullMatrix();
1263         for (unsigned int i = 0; i < m_rhsvector->size(); ++i)
1264             (*m_rhsvector) [i] = 0;
1265         AssemblGlobal();
1266         //SecondConditions();
1267         //ApplySources();
1268         //StefanConditions();
1269         MainConditions();
1270     }
1271     template<class Problem, class Grid, class Matrix>
1272     void DGMMethod<Problem, Grid, Matrix>::Rediscretization()
1273     {
1274         m_time += m_step;
1275         m_GlobalMatrix->NullMatrix();
1276         for (unsigned int i = 0; i < m_rhsvector->size(); ++i)
1277             (*m_rhsvector) [i] = 0;
1278         AssemblGlobal();
1279         SecondConditions();
1280         ThirdConditions();
1281         StefanConditions();

```

```

1282         //ApplySources();
1283         MainConditions();
1284     }
1285     template<class Problem, class Grid, class Matrix>
1286     void DGMethod<Problem, Grid, Matrix>::GetSolution(std::vector<double>& vec)
1287     {
1288         int size = vec.size();
1289         //Translation(vec);
1290         for (int i = 0; i < size; ++i)
1291             vec[i] = m_solution[i];
1292     }
1293     template<class Problem, class Grid, class Matrix>
1294     const double DGMethod<Problem, Grid, Matrix>::GetSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
1295     {
1296         double sum{ 0 };
1297         auto nfem{ g.FindElement(p) };
1298         if (nfem < 0)
1299             return 0.;
1300         auto elem{ g.GetElement(nfem) };
1301         auto dofs{ elem->GetDoFs() };
1302         for (auto i{ 0 }; i < dofs; ++i)
1303             sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1304         return sum;
1305     }
1306     template<class Problem, class Grid, class Matrix>
1307     const double DGMethod<Problem, Grid, Matrix>::GetSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p, const int nfem)
1308     {
1309         double sum{ 0 };
1310         //if (nfem < 0)
1311         //    return 0.;
1312         auto elem{ g.GetElement(nfem) };
1313         auto dofs{ elem->GetDoFs() };
1314         //std::cout << nfem << std::endl;
1315         for (auto i{ 0 }; i < dofs; ++i)
1316             sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1317         return sum;
1318     }
1319     template<class Problem, class Grid, class Matrix>
1320     const Mesh::Point DGMethod<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
1321     {
1322         Mesh::Point sum{ 0, 0, 0 };
1323         auto nfem{ g.FindElement(p) };
1324         auto elem{ g.GetElement(nfem) };
1325         auto dofs{ elem->GetDoFs() };
1326         for (auto i{ 0 }; i < dofs; ++i)
1327             sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
1328         return sum;
1329     }
1330     template<class Problem, class Grid, class Matrix>
1331     const Mesh::Point DGMethod<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p, const int nfem)
1332     {
1333         Mesh::Point sum{ 0, 0, 0 };
1334         auto elem{ g.GetElement(nfem) };
1335         auto dofs{ elem->GetDoFs() };
1336         for (auto i{ 0 }; i < dofs; ++i)
1337             sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
1338         return sum;
1339     }
1340     template<class Problem, class Grid, class Matrix>
1341     void DGMethod<Problem, Grid, Matrix>::LoadSolution(const std::vector<double>& vec)
1342     {
1343         m_solution.resize(vec.size());
1344         for (unsigned int i = 0; i < vec.size(); ++i)
1345             m_solution[i] = vec[i];
1346     }
1347     template<class Problem, class Grid, class Matrix>
1348     void DGMethod<Problem, Grid, Matrix>::OutMeshFormat(const std::string& file_name, const
std::vector<double>& vec)
1349     {
1350         const int size{ (int)m_Grid->GetNodes().size() };
1351         const int number{ (int)m_Grid->GetElements().size() };
1352         //const int size{ number * 4 };
1353         std::ofstream ofs(file_name + ".dat", std::ios::out);
1354         std::string title("TITLE = \"Mesh data\"\\n Variables = \"X\\\", \"Y\\\", \"Z\\\", \"U\\\"\\n Zone N
= \" + std::to_string(size) + \", E = \" + std::to_string(number) + \", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\\n");
1355         ofs << title;
1356         Mesh::Point p;
1357         for (int i = 0; i < size; ++i)
1358         {
1359             p = m_Grid->GetNodes()[i];
1360             ofs << p.x << "\\t" << p.y << "\\t" << p.z << "\\t" << GetValue(p, vec, 1) << std::endl;
1361         }

```

```

1362         for (int i = 0; i < number; ++i)
1363         {
1364             auto elem = m_Grid->GetElements()[i];
1365             for (int k = 0; k < 4; ++k)
1366             {
1367                 ofs « elem->GetNodes()[k] + 1 « "\t";
1368             }
1369             ofs « std::endl;
1370         }
1371         ofs.close();
1372     }
1373     template<class Problem, class Grid, class Matrix>
1374     void DGMMethod<Problem, Grid, Matrix>::OutMeshTimeFormat(const std::string& file_name, const
std::vector<double>& vec)
1375     {
1376         const int size{ (int)m_Grid->GetNodes().size() };
1377         const int number{ (int)m_Grid->GetElements().size() };
1378         //const int size{ number * 4 };
1379         std::ofstream ofs(file_name + ".dat", std::ios::out | std::ios::app);
1380         std::string title("TITLE = \"Mesh data\"\n Variables = \"X\", \"Y\", \"Z\", \"U\"\n Zone N
= \" + std::to_string(size) + \", E = \" + std::to_string(number) + \", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\n");
1381         ofs « title;
1382         Mesh::Point p;
1383         for (int i = 0; i < size; ++i)
1384         {
1385             p = m_Grid->GetNodes()[i];
1386             ofs « p.x « "\t" « p.y « "\t" « p.z « "\t" « GetValue(p, vec, 1) « std::endl;
1387         }
1388         for (int i = 0; i < number; ++i)
1389         {
1390             auto elem = m_Grid->GetElements()[i];
1391             for (int k = 0; k < 4; ++k)
1392             {
1393                 ofs « elem->GetNodes()[k] + 1 « "\t";
1394             }
1395             ofs « std::endl;
1396         }
1397         ofs.close();
1398     }
1399     template<class Problem, class Grid, class Matrix>
1400     void DGMMethod<Problem, Grid, Matrix>::ProjectSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&, const int)> GetVal,
std::vector<double>& vec)
1401     {
1402         for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1403         {
1404             auto elem = m_Grid->GetElements()[i];
1405             int order = elem->GetDoF();
1406             for (int j = 0; j < order; ++j)
1407                 sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes()[j]], vec, i);
1408         }
1409     }
1410     template<class Problem, class Grid, class Matrix>
1411     void DGMMethod<Problem, Grid, Matrix>::ProjectSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&)> GetVal,
std::vector<double>& vec, const int)
1412     {
1413         for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1414         {
1415             auto elem = m_Grid->GetElements()[i];
1416             int order = elem->GetDoF();
1417             for (int j = 0; j < order; ++j)
1418                 sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes()[j]], vec);
1419         }
1420     }
1421     template<class Problem, class Grid, class Matrix>
1422     const std::vector<double> DGMMethod<Problem, Grid, Matrix>::SetSolution(const int sol, const int
liq, const double s, const double l, const double m)
1423     {
1424         int i;
1425         m_solution.resize(m_Grid->GetNodes().size());
1426         for (i = 0; i < m_Grid->GetElements().size(); ++i)
1427         {
1428             auto elem = m_Grid->GetElements()[i];
1429             int order = elem->GetDoF();
1430             if (m_Grid->GetElements()[i]->GetType() == liq)
1431                 for (int j = 0; j < order; ++j)
1432                     m_solution[elem->GetNodes()[j]] = 1;
1433             else
1434                 for (int j = 0; j < order; ++j)
1435                     m_solution[elem->GetNodes()[j]] = s;
1436         }
1437         for (auto bnd : m_Grid->GetBoundaryConditions())
1438         {
1439             //if (get<0>(bnd.second) == 4)
1440

```

```

1441         {
1442             for (auto row : m_Grid->GetBoundary())
1443             {
1444                 if (bnd.first == row->GetType())
1445                 {
1446                     int dofs = row->GetDoF();
1447                     for (int i = 0; i < dofs; ++i)
1448                     {
1449                         m_solution[row->GetNodes(i)] = m;
1450                     }
1451                 }
1452             }
1453         }
1454     }
1455     return m_solution;
1456 }
1457 template<class Problem, class Grid, class Matrix>
1458 DGMethod<Problem, Grid, Matrix>::~DGMethod()
1459 {
1460     delete m_Grid;
1461 }
1462 }
1463 }
1464
1465 #endif // !CORENC_METHODS_DGMethod_h

```

7.73 CoreNCFEM/Methods/DGMethodZero.h File Reference

```

#include <functional>
#include <set>
#include "../Point.h"
#include "../Parameter.h"
#include "CSMethod.h"
#include <memory>
#include <cmath>
#include <map>
#include <algorithm>
#include <vector>
#include <iostream>
#include <fstream>
#include <string>

```

Classes

- class [corenc::method::CDGMethodZero< Type >](#)
- class [corenc::method::DGMethodZero< Problem, Grid, Matrix >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)
- namespace [corenc::method](#)

7.74 DGMMethodZero.h

[Go to the documentation of this file.](#)

```

1 #ifndef DGMETHODZERO_H
2 #define DGMETHODZERO_H
3
4 // DGMMethodZero.h describes an abstract interface and functions for a DG method with zero Dirichlet
   boundaries
5 #include <functional>
6 #include <set>
7 #include "../Point.h"
8 #include "../Parameter.h"
9 #include "CSMethod.h"
10 #include <memory>
11 #include <cmath>
12 #include <map>
13 #include <algorithm>
14 #include <vector>
15 #include <iostream>
16 #include <fstream>
17 #include <string>
18 namespace corenc
19 {
20     namespace Mesh
21     {
22         class Point;
23     }
24     namespace method
25     {
26         // class Type = Type of the solution, for ex vector or double, or even more specific
27
28
29         template<class Type>
30         class CDGMethodZero
31         {
32         public:
33             CDGMethodZero() {};
34             virtual ~CDGMethodZero() {};
35             virtual const int Assemble() = 0;
36             virtual const Type GetSolution(const std::vector<double>& point)
37             const = 0;
38             virtual const std::vector<Type> GetSolution() const = 0;
39             virtual const Type GetMaxSolution() const = 0;
40             virtual const Type GetMinSolution() const = 0;
41         };
42
43         template<class Problem, class Grid, class Matrix>
44         class DGMMethodZero
45         {
46         public:
47             DGMMethodZero() :
48                 m_problem{nullptr},
49                 m_Grid{nullptr},
50                 m_GlobalMatrix{nullptr},
51                 m_RightMatrix{nullptr},
52                 m_rhsvector{nullptr}
53             {}
54             DGMMethodZero(
55                 Problem* p,
56                 Grid* g,
57                 Matrix* m,
58                 std::vector<double>* rhs):
59                 m_problem{ p },
60                 m_Grid{ g->Clone() },
61                 m_GlobalMatrix{ m },
62                 m_Ns{ g->GetNumberOfElements() },
63                 m_Ns{ g->GetNumberOfBoundaries() },
64                 m_rhsvector{ rhs }{
65                 //GeneratePortrait();
66             }
67             DGMMethodZero(
68                 Problem* p,
69                 Grid* g,
70                 Matrix* m,
71                 Matrix* rm,
72                 std::vector<double>* rhs):
73                 m_problem{ p },
74                 m_Grid{ g->Clone() },
75                 m_GlobalMatrix{ m },
76                 m_RightMatrix{ rm },
77                 m_Ns{ g->GetNumberOfElements() },
78                 m_Ns{ g->GetNumberOfBoundaries() },
79                 m_rhsvector{ rhs }{
80                 //GeneratePortrait();
81             }
82         };
83     }
84 }

```



```

81         DGMethodZero(const std::shared_ptr<Grid>& grid) :m_Grid{ grid->Clone() } {}
82         DGMethodZero(Grid* grid) :m_Grid{ grid->Clone() } {}
83         DGMethodZero(const DGMethodZero& meth) :
84             m_Grid{ meth.m_Grid->Clone() },
85             //m_GlobalMatrix{ meth.m_GlobalMatrix->Clone() },
86             //m_rhsvector{ meth.m_rhsvector },
87             //m_problem{ meth.m_problem },
88             m_time{ meth.m_time },
89             //m_solution{ meth.m_solution },
90             m_size{ meth.m_size },
91             m_N{ meth.m_N },
92             m_Ns{ meth.m_Ns },
93             m_nums{ meth.m_nums }
94     {};
95     void Discretization();
96     const double GetValue(const Mesh::Point&) const;
97     const double GetValue(const Mesh::Point&, const std::vector<double>& vec)
98     const;
99     const double GetValue(const Mesh::Point&, const std::vector<double>& vec,
100     const int num) const;
101     //const Mesh::Point GetGradValue(const Mesh::Point&, const std::vector<double>& vec)
102     const;
103     //const Mesh::Point GetLambdaGrad(const Mesh::Point&, const std::vector<double>&
104     vec) const;
105     const double GetEffective(const std::vector<double>& vec) const;
106     void ProjectSolution(std::vector<double>&, std::function<const
107     double(const Mesh::Point&, const std::vector<double>&, const int)> GetValue, std::vector<double>& sol, const
108     sol);
109     void ProjectSolution(std::vector<double>&, std::function<const
110     double(const Mesh::Point&, const std::vector<double>&)> GetValue, std::vector<double>& sol, const
111     int);
112     void LoadSolution(const std::vector<double>& vec);
113     const std::vector<double> SetSolution(const int sol, const int liq, const double, const
114     double, const double);
115     void GetSolution(std::vector<double>& vec);
116     void Rediscrretization(const std::shared_ptr<Grid>&);
117     void Rediscrretization();
118     void SetTimeStep(const double& step) { m_step = step; m_time = step;
119     }
120     Matrix* GetGlobalMatrix() const;
121     Grid* GetMesh() { return m_Grid; }
122     const std::vector<double> GetRightVector() const;
123     void OutDatFormat(const Mesh::Point& min, const Mesh::Point& max,
124     const std::string& file_name, const std::vector<double>& vec) const;
125     void OutMeshFormat(const std::string& file_name, const
126     std::vector<double>& vec);
127     void OutMeshTimeFormat(const std::string& file_name, const
128     std::vector<double>& vec);
129     static const double GetSolution(const Grid& g, const std::vector<double> &weights,
130     const Mesh::Point& p);
131     static const double GetSolution(const Grid& g, const std::vector<double> &weights,
132     const Mesh::Point& p, const int nfem);
133     static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
134     const Mesh::Point& p);
135     static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
136     const Mesh::Point& p, const int n);
137     ~DGMethodZero();
138     private:
139     void GeneratePortrait();
140     void AssemblGlobal();
141     void MainConditions();
142     void SecondConditions();
143     void ThirdConditions();
144     void StefanConditions();
145     void ApplySources();
146     const int AssembleLocalMatrix(const int);
147     const int AssembleIDUDVMMatrix(const int);
148     const int AssembleIDUVMMatrix(const int);
149     const int AssembleIUDVMMatrix(const int);
150     const int AssembleRUVMatrix(const int);
151     const int AssembleSUPGMatrix(const int);
152     const int AssembleLocalMatrix(const int, const int);
153     const int AssembleInter();
154     Grid* m_Grid = nullptr;
155     Matrix* m_GlobalMatrix = nullptr;
156     Matrix* m_RightMatrix = nullptr;
157     Problem* m_problem = nullptr;
158     std::vector<double> m_solution;
159     std::vector<double>* m_rhsvector;
160     unsigned int m_size;
161     double m_step{ 0.1 };
162     double m_time{ 0.1 };
163     unsigned int m_N;
164     unsigned int m_Ns;
165     std::vector<unsigned int> m_nums;
166     // interpolation nodes
167     std::vector<std::vector<int>> m_inums;

```

```

151
152     };
153
154     template<class Problem, class Grid, class Matrix>
155     void DGMethodZero<Problem, Grid, Matrix>::Discretization()
156     {
157         GeneratePortrait();
158         AssemblGlobal();
159         AssembleInter();
160         //ApplySources();
161         //SecondConditions();
162         //ThirdConditions();
163         //MainConditions();
164         //StefanConditions();
165     }
166     template<class Problem, class Grid, class Matrix>
167     void DGMethodZero<Problem, Grid, Matrix>::GeneratePortrait()
168     {
169         const auto& el = m_Grid->GetElement(0);
170         int order = m_Grid->GetElement(0)->GetDoFs();
171         std::vector<std::set<unsigned int> temp;
172         //m_Ns = m_Grid->GetNumberOfINodes();
173         m_Ns = m_Grid->GetNumberOfBoundaries();
174         m_N = m_Grid->GetNumberOfElements();
175         //temp.resize(m_Grid->GetNumberOfINodes());
176         unsigned i, j, k;
177         m_nums.resize(m_N);
178         m_inums.resize(m_N);
179         int size;
180         m_size = 0;
181         std::cout << "nums" << std::endl;
182         for (k = 0; k < m_N; ++k)
183         {
184             const auto& elem{ m_Grid->GetElement(k) };
185             size = 0;
186             m_inums[k].resize(order);
187             for (i = 0; i < order; ++i)
188             {
189                 if (m_Grid->interpolate(elem->GetNode(i)) > -1)
190                 {
191                     m_inums[k][i] = size;
192                     ++size;
193                 }
194             }
195             m_nums[k] = m_size;
196             m_size += size;
197             std::cout << k << "\t" << m_nums[k] << std::endl;
198         }
199         int sz = m_Ns;
200         int nk, ne;
201         int sizej = 0;
202         int sizei = 0;
203         temp.resize(m_size);
204         for (k = 0; k < sz; ++k)
205         {
206             auto bound = m_Grid->GetBoundary(k);
207             nk = bound->GetNeighbour(0);
208             ne = bound->GetNeighbour(1);
209             std::cout << nk << ne << std::endl;
210             sizei = 0;
211             sizej = 0;
212             if (ne != -1)
213             {
214                 auto elemk = m_Grid->GetElement(nk);
215                 auto eleme = m_Grid->GetElement(ne);
216                 size = 0;
217                 for (i = 0; i < order; ++i)
218                 {
219                     int inode = m_Grid->interpolate(elemk->GetNode(i));
220                     if (inode > -1)
221                     {
222                         //sizej = sizei + 1;
223                         for (j = i + 1; j < order; ++j)
224                         {
225                             int jnode = m_Grid->interpolate(eleme->GetNode(j));
226                             if (jnode > -1)
227                             {
228                                 temp[m_nums[nk] + m_inums[nk][j]].insert(m_nums[nk] +
229 m_inums[nk][i]);
230                                 //temp[m_nums[nk] + sizej].insert(m_nums[nk] + sizei);
231                                 //++sizej;
232                                 //std::cout << "k";
233                             }
234                         }
235                     }
236                     //++sizei;
237                 }
238             }
239         }

```

```

237         sizei = 0;
238         sizej = 0;
239         for (i = 0; i < order; ++i)
240         {
241             int inode = m_Grid->interpolate(elemk->GetNode(i));
242             if (inode > -1)
243             {
244                 sizej = 0;
245                 for (j = 0; j < order; ++j)
246                 {
247                     int jnode = m_Grid->interpolate(eleme->GetNode(j));
248                     if (jnode > -1)
249                     {
250                         temp[m_nums[ne] + m_inums[ne][j]].insert(m_nums[nk] +
m_inums[nk][i]);
251                         //temp[m_nums[ne] + sizej].insert(m_nums[nk] + sizei);
252                         ++sizej;
253                         std::cout << "k";
254                     }
255                 }
256                 ++sizei;
257             }
258         }
259     }
260     else
261     {
262         sizei = 0;
263         sizej = 0;
264         auto elemk = m_Grid->GetElement(nk);
265         size = 0;
266         for (i = 0; i < order; ++i)
267         {
268             int inode = m_Grid->interpolate(elemk->GetNode(i));
269             if (inode > -1)
270             {
271                 sizej = sizei + 1;
272                 for (j = i + 1; j < order; ++j)
273                 {
274                     int jnode = m_Grid->interpolate(elemk->GetNode(j));
275                     if (jnode > -1)
276                     {
277                         temp[m_nums[nk] + m_inums[nk][j]].insert(m_nums[nk] +
m_inums[nk][i]);
278                         //temp[m_nums[nk] + sizej].insert(m_nums[nk] + sizei);
279                         ++sizej;
280                         std::cout << "k";
281                     }
282                 }
283                 ++sizei;
284             }
285         }
286     }
287 }
288 if(m_problem->findTerm(Terms::RUV))
289     m_RightMatrix->Create(temp.size(), temp);
290
291 for (auto & it : temp)
292 {
293     for (auto& it2 : it)
294         std::cout << it2 << "\t";
295     std::cout << std::endl;
296 }
297 //m_GlobalMatrix = std::shared_ptr<Matrix>(new Matrix(m_Grid->GetNumberOfNodes(), temp));
298 //m_rhsvector.resize(m_Grid->GetNumberOfNodes());
299 //std::cout << temp.size() << std::endl;
300 m_GlobalMatrix->Create(temp.size(), temp);
301 m_rhsvector->resize(temp.size());
302 //m_solution.resize(m_Grid->GetNumberOfNodes());
303 //for (int l = 0; l < m_Grid->GetNumberOfNodes(); ++l)
304 //    m_solution[l] = 20;
305 }
306 template<class Problem, class Grid, class Matrix>
307 void DGMethodZero<Problem, Grid, Matrix>::AssemblGlobal()
308 {
309     int l;
310     //std::vector<std::future<int>> futures;
311     int i, j, k, nodes;
312     double mij;
313     const int terms{ (int)m_problem->getNumberOfTerms() };
314     for (k = 0; k < terms; ++k)
315     {
316         switch (m_problem->getTerm(k))
317         {
318             case Terms::IDUDV:
319                 for (l = 0; l < m_N; ++l)
320                 {
321                     std::cout << "IDUDV: " << l << std::endl;

```

```

322         AssembleIDUDVMatrix(l);
323     }
324     break;
325     case Terms::IDUV:
326         for (l = 0; l < m_N; ++l)
327             AssembleIDUVMatrix(l);
328         break;
329     case Terms::IUDV:
330         for (l = 0; l < m_N; ++l)
331             AssembleIUDVMatrix(l);
332         break;
333     case Terms::SUPG:
334         for (l = 0; l < m_N; ++l)
335             AssembleSUPGMatrix(l);
336         break;
337     case Terms::RUV:
338         for (l = 0; l < m_N; ++l)
339             AssembleRUVMatrix(l);
340         break;
341     default:
342         break;
343 }
344 }
345 //for (l = 0; l < m_N; ++l)
346 //futures.push_back(async(&DGMMethod<Problem, Grid, Matrix>::AssembleLocalMatrix, this,
1));
347 // AssembleLocalMatrix(l, 0);
348 //for (auto &it : futures)
349 //it.get();
350 }
351
352 template<class Problem, class Grid, class Matrix>
353 const int DGMMethodZero<Problem, Grid, Matrix>::AssembleIDUDVMatrix(const int l)
354 {
355     int i, j, k, nodes;
356     double mij;
357     const auto& elem{ m_Grid->GetElement(l) };
358     const int dofs{ (int)elem->GetDoFs() };
359     const int terms{ (int)m_problem->getNumberOfTerms() };
360     nodes = elem->GetNumberOfNodes();
361     std::vector<Mesh::Point> points(nodes);
362     for (i = 0; i < nodes; ++i)
363         points[i] = m_Grid->GetNode(elem->GetNode(i));
364     int sizei = 0, sizej = 0;
365     for (i = 0; i < (int)dofs; ++i)
366     {
367         auto inode = m_Grid->interpolate(elem->GetNode(i));
368         if (inode == -1)
369             continue;
370         sizej = 0;
371         for (j = 0; j < (int)dofs; ++j)
372         {
373             auto jnode = m_Grid->interpolate(elem->GetNode(j));
374             if (jnode == -1)
375                 continue;
376             auto M = [&](const Mesh::Point& p)
377             {
378                 //auto m = elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
379                 return m_problem->get_parameter(Terms::IDUDV, l, elem->GetType(), p) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
380             };
381             //mij = m_Grid->getParameter(Parameters::DIFFUSION, l, j) * elem->Integrate(M,
points);
382             mij = elem->Integrate(M, points);
383             //m_GlobalMatrix->AddElement(inode, jnode, mij);
384             m_GlobalMatrix->AddElement(m_nums[l] + sizei, m_nums[l] + sizej, mij);
385             ++sizej;
386         }
387         ++sizei;
388     }
389     return 0;
390 }
391
392 template<class Problem, class Grid, class Matrix>
393 const int DGMMethodZero<Problem, Grid, Matrix>::AssembleIDUDVMatrix(const int l)
394 {
395     int i, j, k, nodes;
396     double mij;
397     const auto& elem{ m_Grid->GetElement(l) };
398     const int dofs{ (int)elem->GetDoFs() };
399     const int terms{ (int)m_problem->getNumberOfTerms() };
400     nodes = elem->GetNumberOfNodes();
401     std::vector<Mesh::Point> points(nodes);
402     for (i = 0; i < nodes; ++i)
403         points[i] = m_Grid->GetNode(elem->GetNode(i));
404     int sizei = 0, sizej = 0;
405     for (i = 0; i < (int)dofs; ++i)

```

```

406     {
407         auto inode = m_Grid->interpolate(elem->GetNode(i));
408         if (inode == -1)
409             continue;
410         sizej = 0;
411         for (j = 0; j < (int)dofs; ++j)
412         {
413             auto jnode = m_Grid->interpolate(elem->GetNode(j));
414             if (jnode == -1)
415                 continue;
416             auto M = [&](const Mesh::Point& p)
417             {
418                 return m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
419             };
420             auto _mij = elem->Integrate(M, points);
421             //m_GlobalMatrix->AddElement(inode, jnode, _mij);
422             m_GlobalMatrix->AddElement(m_nums[l] + sizei, m_nums[l] + sizej, _mij);
423             ++sizej;
424         }
425         ++sizei;
426     }
427     return 0;
428 }
429
430 template<class Problem, class Grid, class Matrix>
431 const int DGMethodZero<Problem, Grid, Matrix>::AssembleIUDVMatrix(const int l)
432 {
433     int i, j, k, nodes;
434     double mij;
435     const auto& elem{ m_Grid->GetElement(l) };
436     const int dofs{ (int)elem->GetDoFs() };
437     const int terms{ (int)m_problem->getNumberOfTerms() };
438     nodes = elem->GetNumberOfNodes();
439     std::vector<Mesh::Point> points(nodes);
440     for (i = 0; i < nodes; ++i)
441         points[i] = m_Grid->GetNode(elem->GetNode(i));
442     int sizei = 0, sizej = 0;
443     for (i = 0; i < dofs; ++i)
444     {
445         auto inode = m_Grid->interpolate(elem->GetNode(i));
446         if (inode == -1)
447             continue;
448         sizej = 0;
449         for (j = 0; j < dofs; ++j)
450         {
451             auto jnode = m_Grid->interpolate(elem->GetNode(j));
452             if (jnode == -1)
453                 continue;
454             auto M = [&](const Mesh::Point& p)
455             {
456                 return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
457             };
458             //mij = m_CoarseGrid->getParameter(Parameters::ADVECTION, 1, j) *
m_flux(m_CoarseGrid->getSolution(l, j)) * elem->Integrate(M, points).x;
459             mij = elem->Integrate(M, points).x;
460             //m_GlobalMatrix->AddElement(inode, jnode, mij);
461             m_GlobalMatrix->AddElement(m_nums[l] + sizei, m_nums[l] + sizej, mij);
462             ++sizej;
463         }
464         ++sizei;
465     }
466     return 0;
467 }
468
469
470 template<class Problem, class Grid, class Matrix>
471 const int DGMethodZero<Problem, Grid, Matrix>::AssembleRUVMatrix(const int l)
472 {
473     int i, j, k, nodes;
474     double mij;
475     const auto& elem{ m_Grid->GetElement(l) };
476     const int dofs{ (int)elem->GetDoFs() };
477     const int terms{ (int)m_problem->getNumberOfTerms() };
478     nodes = elem->GetNumberOfNodes();
479     std::vector<Mesh::Point> points(nodes);
480     for (i = 0; i < nodes; ++i)
481         points[i] = m_Grid->GetNode(elem->GetNode(i));
482     int sizei = 0, sizej = 0;
483     for (i = 0; i < (int)dofs; ++i)
484     {
485         auto inode = m_Grid->interpolate(elem->GetNode(i));
486         if (inode == -1)
487             continue;
488         sizej = 0;
489         for (j = 0; j < (int)dofs; ++j)
490         {

```

```

491         auto M = [&](const Mesh::Point& p)
492         {
493             double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p,
0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
494             double h = elem->GetMeasure();
495             double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
496             double tau = 0.;
497             //double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
498
499             if (Pe >= 1)
500                 tau = h / 2. / vel;
501             else
502                 tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
503             auto supg = tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p,
0) * elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p) * elem->GetShapeFunction(i, p);
504             return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p); // + supg;
505         };
506         mij = elem->Integrate(M, points);
507         auto jnode = m_Grid->interpolate(elem->GetNode(j));
508         if (inode > -1 && jnode > -1)
509         {
510             m_RightMatrix->AddElement(m_nums[l] + sizei, m_nums[l] + sizej, mij);
511             ++sizej;
512         }
513     }
514     ++sizei;
515 }
516 return 0;
517 }
518 }
519
520 template<class Problem, class Grid, class Matrix>
521 const int DGMethodZero<Problem, Grid, Matrix>::AssembleSUPGMatrix(const int l)
522 {
523     int i, j, k, nodes;
524     double mij;
525     const auto& elem{ m_Grid->GetElement(l) };
526     const int dofs{ (int)elem->GetDoFs() };
527     const int terms{ (int)m_problem->getNumberOfTerms() };
528     nodes = elem->GetNumberOfNodes();
529     std::vector<Mesh::Point> points(nodes);
530     for (i = 0; i < nodes; ++i)
531         points[i] = m_Grid->GetNode(elem->GetNode(i));
532     for (i = 0; i < (int)dofs; ++i)
533     {
534         for (j = 0; j < (int)dofs; ++j)
535         {
536             auto inode = m_Grid->interpolate(elem->GetNode(i));
537             auto jnode = m_Grid->interpolate(elem->GetNode(j));
538             if (inode == -1 || jnode == -1)
539                 continue;
540             auto M = [&](const Mesh::Point& p)
541             {
542                 double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p,
0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
543                 double h = elem->GetMeasure();
544                 //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
545                 double tau = 0.;
546                 double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
547                 //double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. /
Pe);
548                 //double beta = h / std::sqrt(3.) * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) -
1. / Pe);
549                 //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. / Pe);
550                 //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. / Pe);
551                 //beta = 0.;
552                 //for (int ii = 0; ii < (int)dofs; ++ii)
553                     //beta += m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(ii, p);
554                 //return beta * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
// elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
555                 if (Pe >= 1)
556                     tau = h / 2. / vel;
557                 else
558                     tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
559                 //return 0.;
560                 return tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
561                     elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
562             };
563         }

```

```

564
565         //double tau =
566         auto _mij = elem->Integrate(M, points);
567         m_GlobalMatrix->AddElement(inode, jnode, _mij);
568     }
569 }
570 return 0;
571 }
572
573
574 template<class Problem, class Grid, class Matrix>
575 const int DGMethodZero<Problem, Grid, Matrix>::AssembleInter()
576 {
577     for (int l = 0; l < m_Ns; ++l)
578     {
579         const auto& bound{ m_Grid->GetBoundary(l) };
580         const auto& nk{ bound->GetNeighbour(0) };
581         const auto& ne{ bound->GetNeighbour(1) };
582         const auto& elemk{ m_Grid->GetElement(nk) };
583         const auto& dofs{ bound->GetDoFs() };
584         const auto& dofsk{ elemk->GetDoFs() };
585         std::vector<Mesh::Point> points(dofs);
586         for (int i = 0; i < dofs; ++i)
587         {
588             points[i] = m_Grid->GetNode(bound->GetNode(i));
589         }
590         if (ne < 0)
591             continue;
592         const auto& eleme{ m_Grid->GetElement(ne) };
593         for (int i = 0; i < dofsk; ++i)
594         {
595             int inode = m_Grid->interpolate(elemk->GetNode(i));
596             if (inode == -1)
597                 continue;
598             for (int j = 0; j < dofsk; ++j)
599             {
600                 int jnode = m_Grid->interpolate(elemk->GetNode(j));
601                 if (jnode == -1)
602                     continue;
603                 auto Tkk = [&](const Mesh::Point& p)
604                 {
605                     auto kappa = m_problem->get_parameter(Terms::IDUDV, l, elemk->GetType(), p);
606                     auto val1 = bound->GetNormal() * elemk->GetShapeFunction(j, p) *
elemk->GetGradShapeFunction(i, p);
607                     auto val2 = bound->GetNormal() * elemk->GetShapeFunction(i, p) *
elemk->GetGradShapeFunction(j, p);
608                     return 0.5 * kappa * (val2 - val1);
609                 };
610                 auto mj = bound->Integrate(Tkk, points);
611                 m_GlobalMatrix->AddElement(m_nums[nk] + m_inums[nk][i], m_nums[nk] +
m_inums[nk][j], mj);
612             }
613         }
614     }
615     for (int i = 0; i < dofsk; ++i)
616     {
617         int inode = m_Grid->interpolate(elemk->GetNode(i));
618         if (inode == -1)
619             continue;
620         for (int j = 0; j < dofsk; ++j)
621         {
622             int jnode = m_Grid->interpolate(elemk->GetNode(j));
623             if (jnode == -1)
624                 continue;
625             auto Tkk = [&](const Mesh::Point& p)
626             {
627                 auto kappa = m_problem->get_parameter(Terms::IDUDV, l, eleme->GetType(), p);
628                 auto val1 = bound->GetNormal() * eleme->GetShapeFunction(j, p) *
elemk->GetGradShapeFunction(i, p);
629                 auto val2 = bound->GetNormal() * elemk->GetShapeFunction(i, p) *
eleme->GetGradShapeFunction(j, p);
630                 return 0.5 * kappa * (val2 + val1);
631             };
632             auto mj = bound->Integrate(Tkk, points);
633             m_GlobalMatrix->AddElement(m_nums[nk] + m_inums[nk][i], m_nums[ne] +
m_inums[ne][j], mj);
634         }
635     }
636 }
637 return 0;
638 }
639
640 template<class Problem, class Grid, class Matrix>
641 const int DGMethodZero<Problem, Grid, Matrix>::AssembleLocalMatrix(const int l, const int old)
642 {
643     int i, j, k, nodes;
644     double mij;

```

```

645     const auto& elem{ m_Grid->GetElement(1) };
646     const int dofs{ (int)elem->GetDofs() };
647     const int terms{ (int)m_problem->getNumberOfTerms() };
648     nodes = elem->GetNumberOfNodes();
649     std::vector<Mesh::Point> points(nodes);
650     for (i = 0; i < nodes; ++i)
651         points[i] = m_Grid->GetNode(elem->GetNode(i));
652     for (k = 0; k < terms; ++k)
653     {
654         switch (m_problem->getTerm(k))
655         {
656             case Terms::IUV:
657                 for (i = 0; i < (int)dofs; ++i)
658                 {
659                     for (j = 0; j < (int)dofs; ++j)
660                     {
661                         auto M = [&](const Mesh::Point& p)
662                         {
663                             return m_problem->get_parameter(Terms::IUV, 1, elem->GetType(), p) *
elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
664                         };
665                         mij = elem->Integrate(M, points);
666                         auto inode = m_Grid->interpolate(elem->GetNode(i));
667                         auto jnode = m_Grid->interpolate(elem->GetNode(j));
668                         if (inode > -1 && jnode > -1)
669                             m_GlobalMatrix->AddElement(inode, jnode, mij);
670                     }
671                 }
672                 break;
673             case Terms::IDUDV:
674                 for (i = 0; i < (int)dofs; ++i)
675                 {
676                     for (j = 0; j < (int)dofs; ++j)
677                     {
678                         auto inode = m_Grid->interpolate(elem->GetNode(i));
679                         auto jnode = m_Grid->interpolate(elem->GetNode(j));
680                         if (inode == -1 || jnode == -1)
681                             continue;
682                         auto M = [&](const Mesh::Point& p)
683                         {
684                             //auto m = elem->GetGradShapeFunction(i, p) *
elem->GetGradShapeFunction(j, p);
685                             return m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(), p) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
686                         };
687                         //mij = m_Grid->getParameter(Parameters::DIFFUSION, 1, j) *
elem->Integrate(M, points);
688                         mij = elem->Integrate(M, points);
689                         m_GlobalMatrix->AddElement(inode, jnode, mij);
690                     }
691                 }
692                 break;
693             case Terms::IDUV:
694                 for (i = 0; i < (int)dofs; ++i)
695                 {
696                     for (j = 0; j < (int)dofs; ++j)
697                     {
698                         auto inode = m_Grid->interpolate(elem->GetNode(i));
699                         auto jnode = m_Grid->interpolate(elem->GetNode(j));
700                         if (inode == -1 || jnode == -1)
701                             continue;
702                         auto M = [&](const Mesh::Point& p)
703                         {
704                             return m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
705                         };
706                         auto _mij = elem->Integrate(M, points);
707                         m_GlobalMatrix->AddElement(inode, jnode, _mij);
708                     }
709                 }
710                 break;
711             case Terms::IUDV:
712                 for (i = 0; i < dofs; ++i)
713                 {
714                     for (j = 0; j < dofs; ++j)
715                     {
716                         auto inode = m_Grid->interpolate(elem->GetNode(i));
717                         auto jnode = m_Grid->interpolate(elem->GetNode(j));
718                         if (inode == -1 || jnode == -1)
719                             continue;
720                         auto M = [&](const Mesh::Point& p)
721                         {
722                             return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
723                         };
724                         //mij = m_CoarseGrid->getParameter(Parameters::ADVECTION, 1, j) *
m_flux(m_CoarseGrid->getSolution(1, j)) * elem->Integrate(M, points).x;
725                         mij = elem->Integrate(M, points).x;

```



```

726         m_GlobalMatrix->AddElement(inode, jnode, mij);
727     }
728 }
729 break;
730 case Terms::EUV:
731     for (i = 0; i < dofs; ++i)
732     {
733         for (j = 0; j < dofs; ++j)
734         {
735             auto M = [&](const Mesh::Point& p)
736             {
737                 return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
738             };
739             mij = elem->Integrate(M, points);
740             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::MASS, l, j) * m_Grid->getSolution(l, j) * mij;
741             //m_rhsvector->operator[] (m_nums[l] + i) +=
m_CoarseGrid->getParameter(Parameters::MASS, l, points[j]) * elem->GetValue(j) * mij;
742         }
743     }
744 break;
745 case Terms::EDUDV:
746     for (i = 0; i < dofs; ++i)
747     {
748         for (j = 0; j < dofs; ++j)
749         {
750             auto M = [&](const Mesh::Point& p)
751             {
752                 return elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
753             };
754             mij = elem->Integrate(M, points);
755             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::DIFFUSION, l, j) * m_Grid->getSolution(l, j) * mij;
756         }
757     }
758 break;
759 case Terms::EDUV:
760     for (i = 0; i < dofs; ++i)
761     {
762         for (j = 0; j < dofs; ++j)
763         {
764             auto M = [&](const Mesh::Point& p)
765             {
766                 return elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
767             };
768             mij = elem->Integrate(M, points).x;
769             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, l, j) * mij;
770         }
771     }
772 break;
773 case Terms::EUDV:
774     for (i = 0; i < dofs; ++i)
775     {
776         for (j = 0; j < dofs; ++j)
777         {
778             auto M = [&](const Mesh::Point& p)
779             {
780                 return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
781             };
782             mij = elem->Integrate(M, points).x;
783             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, l, j) * mij; // *mij;
784         }
785     }
786 break;
787 case Terms::EFV:
788     for (i = 0; i < dofs; ++i)
789     {
790         /*for (j = 0; j < dofs; ++j)
791         {
792             auto M = [&](const Mesh::Point& p)
793             {
794                 return m_problem->get_parameter(Terms::EFV, elem->GetType(), l, j, p) *
elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
795             };
796             mij = elem->Integrate(M, points);
797             m_rhsvector->operator[] (elem->GetNode(i)) += mij;
798         }*/
799         auto M = [&](const Mesh::Point& p)
800         {
801             return m_problem->get_parameter(Terms::EFV, elem->GetType(), l, i, p) *
elem->GetShapeFunction(i, p);
802         };
803         mij = elem->Integrate(M, points);
804         m_rhsvector->operator[] (elem->GetNode(i)) += mij;

```

```

805         }
806         break;
807     case Terms::RUV:
808         for (i = 0; i < (int)dofs; ++i)
809         {
810             for (j = 0; j < (int)dofs; ++j)
811             {
812                 auto M = [&](const Mesh::Point& p)
813                 {
814                     return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
815                 };
816                 mij = elem->Integrate(M, points);
817                 auto inode = m_Grid->interpolate(elem->GetNode(i));
818                 auto jnode = m_Grid->interpolate(elem->GetNode(j));
819                 if (inode > -1 && jnode > -1)
820                     m_RightMatrix->AddElement(inode, jnode, mij);
821             }
822         }
823         break;
824     case Terms::SUPG:
825         for (i = 0; i < (int)dofs; ++i)
826         {
827             for (j = 0; j < (int)dofs; ++j)
828             {
829                 auto inode = m_Grid->interpolate(elem->GetNode(i));
830                 auto jnode = m_Grid->interpolate(elem->GetNode(j));
831                 if (inode == -1 || jnode == -1)
832                     continue;
833                 auto M = [&](const Mesh::Point& p)
834                 {
835                     double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
836                     double h = elem->GetMeasure();
837                     //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
838                     double tau = 0.;
839                     double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
840                     //double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) -
1.) - 1. / Pe);
841                     double beta = h / std::sqrt(3.) * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) -
1.) - 1. / Pe);
842                     //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1.
/ Pe);
843                     //beta = 0.;
844                     //for (int ii = 0; ii < (int)dofs; ++ii)
845                     //beta += m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * elem->GetGradShapeFunction(ii, p);
846                     return beta * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
847                     if (Pe >= 1)
848                         tau = h / 2. / vel;
849                     else
850                         tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
851                     //return 0.;
852                     return tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
853                 };
854             }
855         }
856         //double tau =
857         auto _mij = elem->Integrate(M, points);
858         m_GlobalMatrix->AddElement(inode, jnode, _mij);
859     }
860 }
861 }
862 break;
863 default:
864     break;
865 }
866 }
867 return 0;
868 }
869 template<class Problem, class Grid, class Matrix>
870 void DGMethodZero<Problem, Grid, Matrix>::MainConditions()
871 {
872     double mu{ 1e10 };
873     const auto n = m_problem->get_number_of_boundaries();
874     const auto m = m_Grid->GetNumberOfBoundaries();
875     for (int i = 0; i < n; ++i)
876     {
877         const auto& type = m_problem->get_boundary_type(i);
878         for (int j = 0; j < m; ++j)
879         {

```

```

880         const auto& row = m_Grid->GetBoundary(j);
881         if (row->GetType() == type)
882         {
883             const int dofs = (int)row->GetDoFs();
884             const int dofs2 = 2;
885             const auto& elem_num = row->GetNeighbour(0);
886             const auto& elem = m_Grid->GetElement(elem_num);
887             const int dofs_elem = elem->GetDoFs();
888             std::vector<Mesh::Point> points(dofs_elem);
889             for (int k = 0; k < dofs_elem; ++k)
890                 points[k] = m_Grid->GetNode(elem->GetNode(k));
891             for (int k = 0; k < dofs; ++k)
892             {
893                 int l = 0;
894                 for (; l < dofs_elem; ++l)
895                 {
896                     if (elem->GetNode(l) == row->GetNode(k))
897                         break;
898                 }
899                 m_GlobalMatrix->NullRow(row->GetNode(k));
900                 //m_GlobalMatrix->operator()(row->GetNode(k), row->GetNode(k)) *= mu;
901                 //m_rhsvector->operator[](row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, m_Grid->GetNode(row->GetNode(k)));
902                 //m_rhsvector->operator[](row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, elem_num, l, m_Grid->GetNode(row->GetNode(k)));
903                 m_rhsvector->operator[](row->GetNode(k)) = elem->GetWeight(l, points,
[=](const Mesh::Point& p) { return m_problem->get_boundary_parameter(0, type, p); });
904                 if(m_problem->findTerm(Terms::RUV))
905                 {
906                     m_RightMatrix->NullRow(row->GetNode(k));
907                     //m_RightMatrix->operator()(row->GetNode(k), row->GetNode(k)) *= mu;
908                 }
909             }
910             /*for (int k = dofs2; k < dofs; ++k)
911             {
912                 m_GlobalMatrix->NullRow(row->GetNode(k));
913                 m_rhsvector->operator[](row->GetNode(k)) = 0;
914             }*/
915         }
916     }
917 }
918 /*for (auto bnd : m_Grid->GetBoundaryConditions())
919 {
920     if (get<0>(bnd.second) == 1)
921         for (auto row : m_Grid->GetBoundary())
922         {
923             if (bnd.first == row->GetType())
924             {
925                 for (int i = 0; i < row->GetDoF(); ++i)
926                 {
927                     m_GlobalMatrix->NullRow(row->GetNodes(i));
928                     m_rhsvector[row->GetNodes(i)] =
get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
929                 }
930             }
931         }
932     }*/
933 }
934 template<class Problem, class Grid, class Matrix>
935 void DGMethodZero<Problem, Grid, Matrix>::SecondConditions()
936 {
937     double theta = 0;
938     int nfem;
939     Mesh::Point temp[3];
940     std::vector<int> local;
941     for (auto bnd : m_Grid->GetBoundaryConditions())
942     {
943         //if (get<0>(bnd.second) == 2)
944         {
945             for (auto row : m_Grid->GetBoundary())
946             {
947                 if (bnd.first == row->GetType())
948                 {
949                     local.resize(0);
950                     int dofs = row->GetDoF();
951                     nfem = row->GetNumberOfElement(0);
952                     auto elem = m_Grid->GetElements()[nfem];
953                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
954                     for (int j = 0; j < dofs; ++j)
955                     {
956                         temp[j] = m_Grid->GetNodes() [row->GetNodes(j)];
957                         for (int i = 0; i < elem->GetDoF(); ++i)
958                         {
959                             if (row->GetNodes(j) == elem->GetNodes()[i])
960                             {
961                                 local.push_back(i);
962                                 break;

```

```

963         }
964     }
965 }
966 for (int i = 0; i < dofs; ++i)
967 {
968     for (int j = 0; j < dofs; ++j)
969     {
970         //theta = get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
971         theta = 0;
972         auto GetMass = [&](const Mesh::Point& p) {return
elem->GetBasis(local[j], p) * elem->GetBasis(local[i], p); };
973         auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
974         //if (i < 2 || j < 2)
975         m_rhsvector[row->GetNodes(i)] += theta * row->Integrate(GetMass,
temp);
976
977         //if (i < 3 || j < 3)
978         // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
979     }
980 }
981 }
982 }
983 }
984 }
985 }
986 template<class Problem, class Grid, class Matrix>
987 void DGMethodZero<Problem, Grid, Matrix>::StefanConditions()
988 {
989     double dest{ 0. }, lat{ 0 };
990     int nfem;
991     Mesh::Point temp[3];
992     std::vector<int> local;
993     for (auto bnd : m_Grid->GetBoundaryConditions())
994     {
995         //if (get<0>(bnd.second) == 4)
996         {
997             lat = 0;
998             //lat = get<2>(bnd.second);
999             for (auto row : m_Grid->GetBoundary())
1000             {
1001                 if (bnd.first == row->GetType())
1002                 {
1003                     local.resize(0);
1004                     int dofs = row->GetDoF();
1005                     nfem = row->GetNumberOfElement(0);
1006                     auto elem = m_Grid->GetElements()[nfem];
1007                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
1008                     for (int j = 0; j < dofs; ++j)
1009                     {
1010                         temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
1011                         for (int i = 0; i < elem->GetDoF(); ++i)
1012                         {
1013                             if (row->GetNodes(j) == elem->GetNodes()[i])
1014                             {
1015                                 local.push_back(i);
1016                                 break;
1017                             }
1018                         }
1019                     }
1020                     for (int i = 0; i < dofs; ++i)
1021                     {
1022                         for (int j = 0; j < dofs; ++j)
1023                         {
1024                             dest = 0;
1025                             //dest = get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
1026                             auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
1027                             //if (i < 2 || j < 2)
1028                             m_rhsvector[row->GetNodes(i)] += dest * lat *
row->Integrate(GetBBasis, temp);
1029
1030                             //if (i < 3 || j < 3)
1031                             // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
1032                         }
1033                     }
1034                 }
1035             }
1036         }
1037     }
1038 }
1039 template<class Problem, class Grid, class Matrix>
1040 void DGMethodZero<Problem, Grid, Matrix>::ThirdConditions()
1041 {
1042     double param{ 0 }, beta{ 0 };

```

```

1043         int nfem;
1044         Mesh::Point temp[6];
1045         std::vector<int> local;
1046         auto fxy = [&](const Mesh::Point& p) {return (10 * p.y*m_time + m_time) / 10; };
1047         //auto fxy = [&](const Point& p){return 10 * p.y + 10 * m_time; };
1048         for (auto bnd : m_Grid->GetBoundaryConditions())
1049         {
1050             //if (get<0>(bnd.second) == 3)
1051             {
1052                 for (auto row : m_Grid->GetBoundary())
1053                 {
1054                     if (bnd.first == row->GetType())
1055                     {
1056                         local.resize(0);
1057                         int dofs = row->GetDoF();
1058                         nfem = row->GetNumberOfElement(0);
1059                         auto elem = m_Grid->GetElements()[nfem];
1060                         //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
1061                         auto order = elem->GetDoF();
1062                         for (int j = 0; j < dofs; ++j)
1063                         {
1064                             temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
1065                             for (int i = 0; i < order; ++i)
1066                             {
1067                                 if (row->GetNodes(j) == elem->GetNodes()[i])
1068                                 {
1069                                     local.push_back(i);
1070                                     break;
1071                                 }
1072                             }
1073                         }
1074                         double val{ 0 };
1075                         for (int i = 0; i < dofs; ++i)
1076                         {
1077                             for (int j = 0; j < dofs; ++j)
1078                             {
1079                                 param = 0;
1080                                 beta = 0;
1081                                 //beta = get<2>(bnd.second);
1082                                 //param = get<1>(bnd.second) (m_Grid->GetNodes()[row->GetNodes(i)]);
1083                                 //param = fxy(temp[j]);
1084                                 auto GetBBasis = [&](const Mesh::Point& p) {return
1085                                     elem->GetBasis(local[j], p)*elem->GetBasis(local[i], p); };
1086                                 //val = row->GetElement(GetBBasis, temp);
1087                                 val = row->Integrate(GetBBasis, temp);
1088                                 m_GlobalMatrix->operator()(row->GetNodes(i), row->GetNodes(j)) +=
1089                                     beta * val;
1090                                 m_rhsvector[row->GetNodes(i)] += beta * param * val;
1091                             }
1092                         }
1093                     }
1094                 }
1095             }
1096         }
1097         template<class Problem, class Grid, class Matrix>
1098         Matrix* DGMethodZero<Problem, Grid, Matrix>::GetGlobalMatrix() const
1099         {
1100             return m_GlobalMatrix;
1101         }
1102         template<class Problem, class Grid, class Matrix>
1103         const double DGMethodZero<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p) const
1104         {
1105             if (!m_solution.size())
1106                 return -1;
1107             double val = 0;
1108             int nfem = -1;
1109             nfem = m_Grid->FindElement(p);
1110             if (nfem == -1)
1111                 return -1;
1112             auto elem = m_Grid->GetElements()[nfem];
1113             for (int i = 0; i < elem->GetDoF(); ++i)
1114                 val += m_solution[elem->GetNodes()[i]] * elem->GetBasis(i, p);
1115             return val;
1116         }
1117         template<class Problem, class Grid, class Matrix>
1118         const double DGMethodZero<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
1119             std::vector<double>& vec) const
1120         {
1121             if (!vec.size())
1122                 return -1;
1123             double val{ 0 };
1124             int nfem{ -1 };
1125             nfem = m_Grid->FindElement(p);
1126             if (nfem == -1)
1127                 return -1;

```

```

1127         auto elem = m_Grid->GetElements()[nfem];
1128         for (int i = 0; i < elem->GetDoFs(); ++i)
1129             val += vec[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1130         return val;
1131     }
1132     template<class Problem, class Grid, class Matrix>
1133     const double DGMethodZero<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
std::vector<double>& vec, const int num) const
1134     {
1135         if (!vec.size() || num < 0)
1136             return -1;
1137         double val{ 0 };
1138         auto elem = m_Grid->GetElements()[num];
1139         for (int i = 0; i < elem->GetDoF(); ++i)
1140             val += vec[elem->GetNodes()[i]] * elem->GetBasis(i, p);
1141         return val;
1142     }
1143     //template<class Problem, class Grid, class Matrix>
1144     //const Mesh::Point DGMethodZero<Problem, Grid, Matrix>::GetGradValue(const Mesh::Point& p,
const std::vector<double>& vec) const
1145     //{
1146     //    Mesh::Point val{ 0, 0, 0 };
1147     //    int nfem{ -1 };
1148     //    nfem = m_Grid->FindElement(p);
1149     //    if (nfem == -1)
1150     //        return val;
1151     //    auto elem = m_Grid->GetElements()[nfem];
1152     //    for (int i = 0; i < elem->GetDoF(); ++i)
1153     //    {
1154     //        val.x += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).x;
1155     //        val.y += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).y;
1156     //        val.z += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).z;
1157     //    }
1158     //    return val;
1159     //}
1160     template<class Problem, class Grid, class Matrix>
1161     const double DGMethodZero<Problem, Grid, Matrix>::GetEffective(const std::vector<double>& vec)
const
1162     {
1163         double sum = 0;
1164         //std::vector<int> dofs;
1165         //Mesh::Point points[10];
1166         //for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1167         //{
1168             //auto mb = [&](const Mesh::Point& b) {return GetGradValue(b, vec)*GetGradValue(b,
vec); };
1169             //dofs.resize(0);
1170             //auto elem = m_Grid->GetElements()[i];
1171             //int order = elem->GetDoF();
1172             //double diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
1173             //for (int j = 0; j < order; ++j)
1174             //{
1175                 //dofs.push_back(elem->GetNodes()[j]);
1176                 //points[j] = m_Grid->GetNodes()[dofs[j]];
1177             //}
1178             //sum += diff * elem->Integrate(mb, points);
1179             //}
1180             //std::cout << "Effect (local): " << sum << std::endl;
1181             //std::cout << "Effect (local) sqrt: " << sqrt(sum) << std::endl;
1182             return sum;
1183         }
1184     //template<class Problem, class Grid, class Matrix>
1185     //const Mesh::Point DGMethodZero<Problem, Grid, Matrix>::GetLambdaGrad(const Mesh::Point& p,
const std::vector<double>& vec) const
1186     //{
1187     //    Mesh::Point val{ 0, 0, 0 };
1188     //    //double val{ 0 };
1189     //    double diff{ 0 };
1190     //    Mesh::Point temp{ 0, 0, 0 };
1191     //    int nfem{ -1 };
1192     //    nfem = m_Grid->FindElement(p);
1193     //    if (nfem == -1)
1194     //        return val;
1195     //    auto elem = m_Grid->GetElements()[nfem];
1196     //    diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
1197     //    for (int i = 0; i < elem->GetDoF(); ++i)
1198     //    {
1199         //        //val += elem->GetGradBasis(i, p) * elem->GetGradBasis(i, p) * vec[elem->GetNodes()[i]]
* vec[elem->GetNodes()[i]] * diff;
1200         //        //val += elem->GetBasis(i, p) * vec[elem->GetNodes()[i]] * diff;
1201         //        temp = elem->GetGradBasis(i, p);
1202         //        val.x += temp.x * vec[elem->GetNodes()[i]] * (diff);
1203         //        val.y += temp.y * vec[elem->GetNodes()[i]] * (diff);
1204         //        val.z += temp.z * vec[elem->GetNodes()[i]] * (diff);
1205         //    }
1206         //    return val;
1207     //}

```

```

1208     template<class Problem, class Grid, class Matrix>
1209     const std::vector<double> DGMethodZero<Problem, Grid, Matrix>::GetRightVector() const
1210     {
1211         return *m_rhsvector;
1212     }
1213     template<class Problem, class Grid, class Matrix>
1214     void DGMethodZero<Problem, Grid, Matrix>::OutDatFormat(const Mesh::Point& mn, const
1215     Mesh::Point& mx, const std::string& file_name, const std::vector<double>& vec) const
1216     {
1217         std::ofstream of(file_name + ".dat");
1218         std::streambuf *buf = std::cout.rdbuf();
1219         std::cout.rdbuf(of.rdbuf());
1220         std::cout << "TITLE = FE-METHOD\n";
1221         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1222         std::cout << "ZONE i=51, j=51, F=POINT\n";
1223         double stepx = (mx.x - mn.x) / 51;
1224         double stepy = (mx.y - mn.y) / 51;
1225         for (int i = 0; i < 51; ++i)
1226             for (int j = 0; j < 51; ++j)
1227                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
1228                 GetValue(Mesh::Point(mn.x + j * stepx, mn.y + i * stepy, mn.z), vec) << std::endl;
1229         std::cout.rdbuf(buf);
1230         of.close();
1231         of.open(file_name + "x.dat");
1232         buf = std::cout.rdbuf();
1233         std::cout.rdbuf(of.rdbuf());
1234         std::cout << "TITLE = FE-METHOD\n";
1235         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1236         std::cout << "ZONE i=51, j=51, F=POINT\n";
1237         for (int i = 0; i < 51; ++i)
1238             for (int j = 0; j < 51; ++j)
1239                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
1240                 GetValue(Mesh::Point(mn.x + j * stepx, mn.y + i * stepy, mn.z), vec) << std::endl;
1241         std::cout.rdbuf(buf);
1242         of.close();
1243         of.open(file_name + "y.dat");
1244         buf = std::cout.rdbuf();
1245         std::cout.rdbuf(of.rdbuf());
1246         std::cout << "TITLE = FE-METHOD\n";
1247         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1248         std::cout << "ZONE i=51, j=51, F=POINT\n";
1249         for (int i = 0; i < 51; ++i)
1250             for (int j = 0; j < 51; ++j)
1251                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
1252                 GetValue(Mesh::Point(mn.x + j * stepx, mn.z, mn.y + i * stepy), vec) << std::endl;
1253         std::cout.rdbuf(buf);
1254         of.close();
1255     }
1256     template<class Problem, class Grid, class Matrix>
1257     void DGMethodZero<Problem, Grid, Matrix>::ApplySources()
1258     {
1259         int nfem = -1;
1260         auto total = m_problem->get_total_sources();
1261         for (int i = 0; i < total; ++i)
1262         {
1263             auto src = m_problem->get_point_source(i);
1264             auto point = src.get_point();
1265             nfem = m_Grid->FindElement(point);
1266             if (nfem != -1)
1267             {
1268                 auto val = src.get_value();
1269                 auto elem = m_Grid->GetElement(nfem);
1270                 for (int j = 0; j < 3; ++j)
1271                     m_rhsvector->operator[](elem->GetNode(j)) += val * elem->GetShapeFunction(j,
1272                     point);
1273             }
1274             nfem = -1;
1275         }
1276         /*for (auto srd : m_Grid->GetDottedSources())
1277         {
1278             nfem = m_Grid->FindElement(srd.first);
1279             if (nfem != -1)
1280             {
1281                 auto elem = m_Grid->GetElements()[nfem];
1282                 for (int i = 0; i < elem->GetDoF(); ++i)
1283                     m_rhsvector[elem->GetNodes()[i]] += srd.second * elem->GetBasis(i, srd.first);
1284             }
1285             nfem = -1;
1286         }*/
1287     }
1288     template<class Problem, class Grid, class Matrix>
1289     void DGMethodZero<Problem, Grid, Matrix>::Rediscretization(const std::shared_ptr<Grid>& grid)
1290     {
1291         m_GlobalMatrix->NullMatrix();
1292         for (unsigned int i = 0; i < m_rhsvector->size(); ++i)

```

```

1290         (*m_rhsvector)[i] = 0;
1291     AssemblGlobal();
1292     //SecondConditions();
1293     //ApplySources();
1294     //StefanConditions();
1295     MainConditions();
1296 }
1297 template<class Problem, class Grid, class Matrix>
1298 void DGMethodZero<Problem, Grid, Matrix>::Rediscretization()
1299 {
1300     m_time += m_step;
1301     m_GlobalMatrix->NullMatrix();
1302     for (unsigned int i = 0; i < m_rhsvector->size(); ++i)
1303         (*m_rhsvector)[i] = 0;
1304     AssemblGlobal();
1305     SecondConditions();
1306     ThirdConditions();
1307     StefanConditions();
1308     //ApplySources();
1309     MainConditions();
1310 }
1311 template<class Problem, class Grid, class Matrix>
1312 void DGMethodZero<Problem, Grid, Matrix>::GetSolution(std::vector<double>& vec)
1313 {
1314     int size = vec.size();
1315     //Translation(vec);
1316     for (int i = 0; i < size; ++i)
1317         vec[i] = m_solution[i];
1318 }
1319 template<class Problem, class Grid, class Matrix>
1320 const double DGMethodZero<Problem, Grid, Matrix>::GetSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
1321 {
1322     double sum{ 0 };
1323     auto nfem{ g.FindElement(p) };
1324     if (nfem < 0)
1325         return 0.;
1326     auto elem{ g.GetElement(nfem) };
1327     auto dofs{ elem->GetDoFs() };
1328     for (auto i{ 0 }; i < dofs; ++i)
1329         sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1330     return sum;
1331 }
1332 template<class Problem, class Grid, class Matrix>
1333 const double DGMethodZero<Problem, Grid, Matrix>::GetSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p, const int nfem)
1334 {
1335     double sum{ 0 };
1336     //if (nfem < 0)
1337     //    return 0.;
1338     auto elem{ g.GetElement(nfem) };
1339     auto dofs{ elem->GetDoFs() };
1340     //std::cout << nfem << std::endl;
1341     for (auto i{ 0 }; i < dofs; ++i)
1342         sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1343     return sum;
1344 }
1345 template<class Problem, class Grid, class Matrix>
1346 const Mesh::Point DGMethodZero<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
1347 {
1348     Mesh::Point sum{ 0, 0, 0 };
1349     auto nfem{ g.FindElement(p) };
1350     auto elem{ g.GetElement(nfem) };
1351     auto dofs{ elem->GetDoFs() };
1352     for (auto i{ 0 }; i < dofs; ++i)
1353         sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
1354     return sum;
1355 }
1356 template<class Problem, class Grid, class Matrix>
1357 const Mesh::Point DGMethodZero<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p, const int nfem)
1358 {
1359     Mesh::Point sum{ 0, 0, 0 };
1360     auto elem{ g.GetElement(nfem) };
1361     auto dofs{ elem->GetDoFs() };
1362     for (auto i{ 0 }; i < dofs; ++i)
1363         sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
1364     return sum;
1365 }
1366 template<class Problem, class Grid, class Matrix>
1367 void DGMethodZero<Problem, Grid, Matrix>::LoadSolution(const std::vector<double>& vec)
1368 {
1369     m_solution.resize(vec.size());
1370     for (unsigned int i = 0; i < vec.size(); ++i)
1371         m_solution[i] = vec[i];
1372 }

```



```

1373     template<class Problem, class Grid, class Matrix>
1374     void DGMethodZero<Problem, Grid, Matrix>::OutMeshFormat(const std::string& file_name, const
std::vector<double>& vec)
1375     {
1376         const int size{ (int)m_Grid->GetNodes().size() };
1377         const int number{ (int)m_Grid->GetElements().size() };
1378         //const int size{ number * 4 };
1379         std::ofstream ofs(file_name + ".dat", std::ios::out);
1380         std::string title("TITLE = \"Mesh data\"\n Variables = \"X\", \"Y\", \"Z\", \"U\"\n Zone N
= \" + std::to_string(size) + \", E = \" + std::to_string(number) + \", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\n");
1381         ofs << title;
1382         Mesh::Point p;
1383         for (int i = 0; i < size; ++i)
1384         {
1385             p = m_Grid->GetNodes()[i];
1386             ofs << p.x << "\t" << p.y << "\t" << p.z << "\t" << GetValue(p, vec, 1) << std::endl;
1387         }
1388         for (int i = 0; i < number; ++i)
1389         {
1390             auto elem = m_Grid->GetElements()[i];
1391             for (int k = 0; k < 4; ++k)
1392             {
1393                 ofs << elem->GetNodes()[k] + 1 << "\t";
1394             }
1395             ofs << std::endl;
1396         }
1397         ofs.close();
1398     }
1399     template<class Problem, class Grid, class Matrix>
1400     void DGMethodZero<Problem, Grid, Matrix>::OutMeshTimeFormat(const std::string& file_name, const
std::vector<double>& vec)
1401     {
1402         const int size{ (int)m_Grid->GetNodes().size() };
1403         const int number{ (int)m_Grid->GetElements().size() };
1404         //const int size{ number * 4 };
1405         std::ofstream ofs(file_name + ".dat", std::ios::out | std::ios::app);
1406         std::string title("TITLE = \"Mesh data\"\n Variables = \"X\", \"Y\", \"Z\", \"U\"\n Zone N
= \" + std::to_string(size) + \", E = \" + std::to_string(number) + \", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\n");
1407         ofs << title;
1408         Mesh::Point p;
1409         for (int i = 0; i < size; ++i)
1410         {
1411             p = m_Grid->GetNodes()[i];
1412             ofs << p.x << "\t" << p.y << "\t" << p.z << "\t" << GetValue(p, vec, 1) << std::endl;
1413         }
1414         for (int i = 0; i < number; ++i)
1415         {
1416             auto elem = m_Grid->GetElements()[i];
1417             for (int k = 0; k < 4; ++k)
1418             {
1419                 ofs << elem->GetNodes()[k] + 1 << "\t";
1420             }
1421             ofs << std::endl;
1422         }
1423         ofs.close();
1424     }
1425     template<class Problem, class Grid, class Matrix>
1426     void DGMethodZero<Problem, Grid, Matrix>::ProjectSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&, const int)> GetVal,
std::vector<double>& vec)
1427     {
1428         for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1429         {
1430             auto elem = m_Grid->GetElements()[i];
1431             int order = elem->GetDoF();
1432             for (int j = 0; j < order; ++j)
1433                 sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes()[j]], vec, i);
1434         }
1435     }
1436     template<class Problem, class Grid, class Matrix>
1437     void DGMethodZero<Problem, Grid, Matrix>::ProjectSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&)> GetVal,
std::vector<double>& vec, const int)
1438     {
1439         for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1440         {
1441             auto elem = m_Grid->GetElements()[i];
1442             int order = elem->GetDoF();
1443             for (int j = 0; j < order; ++j)
1444                 sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes()[j]], vec);
1445         }
1446     }
1447     template<class Problem, class Grid, class Matrix>
1448     const std::vector<double> DGMethodZero<Problem, Grid, Matrix>::SetSolution(const int sol, const
int liq, const double s, const double l, const double m)

```

```

1449     {
1450         int i;
1451         m_solution.resize(m_Grid->GetNodes().size());
1452         for (i = 0; i < m_Grid->GetElements().size(); ++i)
1453         {
1454             auto elem = m_Grid->GetElements()[i];
1455             int order = elem->GetDoF();
1456             if (m_Grid->GetElements()[i]->GetType() == liq)
1457                 for (int j = 0; j < order; ++j)
1458                     m_solution[elem->GetNodes()[j]] = 1;
1459             else
1460                 for (int j = 0; j < order; ++j)
1461                     m_solution[elem->GetNodes()[j]] = s;
1462         }
1463
1464         for (auto bnd : m_Grid->GetBoundaryConditions())
1465         {
1466             //if (get<0>(bnd.second) == 4)
1467             {
1468                 for (auto row : m_Grid->GetBoundary())
1469                 {
1470                     if (bnd.first == row->GetType())
1471                     {
1472                         int dofs = row->GetDoF();
1473                         for (int i = 0; i < dofs; ++i)
1474                         {
1475                             m_solution[row->GetNodes(i)] = m;
1476                         }
1477                     }
1478                 }
1479             }
1480         }
1481         return m_solution;
1482     }
1483     template<class Problem, class Grid, class Matrix>
1484     DGMethodZero<Problem, Grid, Matrix>::~DGMethodZero()
1485     {
1486         delete m_Grid;
1487     }
1488 }
1489 }
1490
1491 #endif // !CORENC_METHODS_DGMethodZero_h

```

7.75 CoreNCFEM/Methods/DGSolution.h File Reference

```
#include "DGMethod.h"
```

Classes

- class [corenc::method::DGSolution< Grid >](#)
- class [corenc::method::STSolution< Grid >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::method](#)

7.76 DGSolution.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_METHODS_DGSOLUTION_H_
2 #define CORENC_METHODS_DGSOLUTION_H_
3
4 #include "DGMMethod.h"
5 namespace corenc
6 {
7     namespace method
8     {
9         template<class Grid>
10         class DGSolution
11         {
12         public:
13             DGSolution() {}
14             DGSolution(const std::vector<double>& w) :m_w{ w } {}
15             DGSolution(const DGSolution<Grid>& dg) :m_w{ dg.m_w } {}
16             DGSolution<Grid>& operator=(const DGSolution<Grid>& dg)
17             {
18                 m_w = dg.m_w;
19                 return *this;
20             }
21             ~DGSolution()
22             {
23                 if (m_w.size() > 0)
24                     std::vector<double>().swap(m_w);
25             }
26             const double getWeight(const Grid& g, const Mesh::Point& p) const
27             {
28                 if (m_w.size() > 0)
29                     return DGMMethod<int, Grid, int>::GetSolution(g, m_w, p);
30                 return 0.;
31             };
32             const std::vector<double> getWeights() const { return m_w; }
33             const int updateWeight(const unsigned int i, const double val)
34             {
35                 if (i < m_w.size())
36                 {
37                     m_w[i] = val;
38                     return 0;
39                 }
40                 return 1;
41             }
42         private:
43             std::vector<double> m_w;
44         };
45         template<class Grid>
46         class STSolution
47         {
48         public:
49             STSolution() {}
50             STSolution(const Grid& g):m_grid{g}{}
51             STSolution(
52                 const std::vector<DGSolution<Grid>>& w,
53                 const std::vector<double> time,
54                 const Grid& g) : m_w{ w }, m_time{ time }, m_grid{g} {}
55             STSolution(const STSolution<Grid>& st) :m_w{ st.m_w }, m_time{ st.m_time }, m_grid{st.m_grid}
56         {}
57
58             STSolution<Grid>& operator=(const STSolution<Grid>& st)
59             {
60                 m_w = st.m_w;
61                 m_time = st.m_time;
62                 m_grid = st.m_grid;
63                 return *this;
64             }
65             ~STSolution()
66             {
67                 if (m_w.size() > 0)
68                     std::vector<DGSolution<Grid>>().swap(m_w);
69                 if (m_time.size() > 0)
70                     std::vector<double>().swap(m_time);
71             }
72             const double getWeight(const Mesh::Point& p, const double time) const
73             {
74                 int i = 0;
75                 auto sz = m_time.size();
76                 if (fabs(time) < 1e-14)
77                     return DGMMethod<Grid>::GetSolution(m_grid, m_w[0].getWeights(), p);
78                 for (; i < sz; ++i)
79                 {
80                     if (time < m_time[i])
81                         break;
82                 }
83                 if (i == sz)

```

```

82         --i;
83         double dt = m_time[i] - m_time[i - 1];
84         auto temp = DGMethod<Grid>::GetSolution(m_grid, m_w[i - 1].getWeights(), p);
85         double du = DGMethod<Grid>::GetSolution(m_grid, m_w[i].getWeights(), p) - temp;
86         return temp + du * (time - m_time[i - 1]) / dt;
87     };
88     const int updateWeight(
89         const std::vector<double> time,
90         const std::vector<DGSolution<Grid>> w
91     )
92     {
93         m_time = time;
94         m_w = w;
95     }
96     const int addTimeLayer(
97         const double time,
98         const DGSolution<Grid> w
99     )
100    {
101        m_time.push_back(time);
102        m_w.push_back(w);
103        return 0;
104    }
105    const std::vector<DGSolution<Grid>> getWeights() const { return m_w; }
106 private:
107     std::vector<DGSolution<Grid>> m_w;
108     std::vector<double> m_time;
109     Grid m_grid;
110 };
111 }
112 }
113 #endif // !CORENC_METHODS_DGSOLUTION_H_

```

7.77 CoreNCFEM/Methods/FEAnalysis.h File Reference

```

#include <vector>
#include "../Point.h"

```

Classes

- class [corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::method](#)

Macros

- #define [CORENC_METHODS_FEANALYSIS_H_](#)

7.77.1 Macro Definition Documentation

7.77.1.1 CORENC_METHODS_FEANALYSIS_H_

```
#define CORENC_METHODS_FEANALYSIS_H_
```

7.78 FEAnalysis.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #ifndef CORENC_METHODS_FEANALYSIS_H_
3 #define CORENC_METHODS_FEANALYSIS_H_
4 #include <vector>
5 #include "../Point.h"
6 namespace corenc
7 {
8     namespace method
9     {
10         template<class Method1, class Method2, class Mesh1, class Mesh2>
11         class FEAnalysis
12         {
13         public:
14             FEAnalysis() {};
15             ~FEAnalysis() {};
16             const double L2Norm( const Method1& method1,
17                                  const Method2& method2,
18                                  const Mesh1& mesh1,
19                                  const Mesh2& mesh2,
20                                  const std::vector<double>& w1,
21                                  const std::vector<double>& w2) const;
22         };
23         template<class Method1, class Method2, class Mesh1, class Mesh2>
24         const double FEAnalysis<Method1, Method2, Mesh1, Mesh2>::L2Norm(
25             const Method1& method1,
26             const Method2& method2,
27             const Mesh1& mesh1,
28             const Mesh2& mesh2,
29             const std::vector<double>& w1,
30             const std::vector<double>& w2) const
31         {
32             double sum{ 0 }, sum2{0};
33             double res, res2;
34             int j;
35             std::vector<int> dofs;
36             int order = mesh1.GetElement(0)->GetDoFs();
37             dofs.resize(order);
38             std::vector<Mesh::Point> points(order);
39             auto sub = [&](const Mesh::Point& p)
40             {
41                 return (method1.GetValue(p, w1) - method2.GetValue(p, w2)) * (method1.GetValue(p, w1) -
42                 method2.GetValue(p, w2));
43             };
44             auto r = [&](const Mesh::Point& p)
45             {
46                 return method1.GetValue(p, w1);
47             };
48             const int n = (int)mesh1.GetNumberOfElements();
49             for (int i = 0; i < n; ++i)
50             {
51                 const auto& elem = mesh1.GetElement(i);
52                 for (j = 0; j < order; ++j)
53                     points[j] = mesh1.GetNode(elem->GetNode(j));
54                 res = elem->Integrate(sub, points);
55                 res2 = elem->Integrate(r, points);
56                 sum += res;
57                 sum2 += res2;
58             }
59             if (dofs.size() > 0)
60                 std::vector<int>().swap(dofs);
61             if (points.size() > 0)
62                 std::vector<Mesh::Point>().swap(points);
63             return sqrt(sum/sum2);
64         }
65     }
66 }
67 #endif // !CORENC_METHODS_FEANALYSIS_H_
68

```

7.79 CoreNCFEM/Methods/FEMethod.h File Reference

```

#include <functional>
#include <set>
#include "../Point.h"

```

```
#include "../Parameter.h"
#include "CSMethod.h"
#include <memory>
#include <cmath>
#include <map>
#include <algorithm>
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
```

Classes

- class [corenc::method::CFEMethod< Type >](#)
- class [corenc::method::FEMethod< Problem, Grid, Matrix >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)
- namespace [corenc::method](#)

Enumerations

- enum class [corenc::method::BoundaryType](#) { [corenc::method::MAIN](#) , [corenc::method::SECOND](#) , [corenc::method::THIRD](#) , [corenc::method::FREE](#) }

7.80 FEMethod.h

[Go to the documentation of this file.](#)

```
1 // FEMethod.h describes an abstract interface and functions for a general finite element method
2
3 #ifndef CORENC_METHODS_FEMethod_h
4 #define CORENC_METHODS_FEMethod_h
5 #include <functional>
6 #include <set>
7 #include "../Point.h"
8 #include "../Parameter.h"
9 #include "CSMethod.h"
10 #include <memory>
11 #include <cmath>
12 #include <map>
13 #include <algorithm>
14 #include <vector>
15 #include <iostream>
16 #include <fstream>
17 #include <string>
18 namespace corenc
19 {
20     namespace Mesh
21     {
22         class Point;
23     }
24     namespace method
25     {
26         enum class BoundaryType
27         {
28             MAIN,
29             SECOND,
30             THIRD,
31             FREE
```

```

32     };
33     // class Type = Type of the solution, for ex vector or double, or even more specific
34
35
36     template<class Type>
37     class CFEMethod
38     {
39     public:
40         CFEMethod() {};
41         virtual ~CFEMethod() {};
42         virtual const int Assemble() = 0;
43         virtual const Type GetSolution(const std::vector<double>& point)
44     const = 0;
45         virtual const std::vector<Type> GetSolution() const = 0;
46         virtual const Type GetMaxSolution() const = 0;
47         virtual const Type GetMinSolution() const = 0;
48     };
49
50     template<class Problem, class Grid, class Matrix>
51     class FEMethod
52     {
53     public:
54         FEMethod() :
55             m_problem{nullptr},
56             m_Grid{nullptr},
57             m_GlobalMatrix{nullptr},
58             m_RightMatrix{nullptr},
59             m_rhsvector{nullptr}
60         {}
61         FEMethod(
62             Problem* p,
63             Grid* g,
64             Matrix* m,
65             std::vector<double>* rhs):
66             m_problem{ p },
67             m_Grid{ g->Clone() },
68             m_GlobalMatrix{ m },
69             m_N{ g->GetNumberOfElements() },
70             m_Ns{ g->GetNumberOfBoundaries() },
71             m_rhsvector{ rhs }{
72             //GeneratePortrait();
73         }
74         FEMethod(
75             Problem* p,
76             Grid* g,
77             Matrix* m,
78             Matrix* rm,
79             std::vector<double>* rhs):
80             m_problem{ p },
81             m_Grid{ g->Clone() },
82             m_GlobalMatrix{ m },
83             m_RightMatrix{ rm },
84             m_N{ g->GetNumberOfElements() },
85             m_Ns{ g->GetNumberOfBoundaries() },
86             m_rhsvector{ rhs }{
87             //GeneratePortrait();
88         }
89         FEMethod(const std::shared_ptr<Grid>& grid) :m_Grid{ grid->Clone() } {}
90         FEMethod(Grid* grid) :m_Grid{ grid->Clone() } {}
91         FEMethod(const FEMethod& meth) :
92             m_Grid{ meth.m_Grid->Clone() },
93             //m_GlobalMatrix{ meth.m_GlobalMatrix->Clone() },
94             //m_rhsvector{ meth.m_rhsvector },
95             //m_problem{ meth.m_problem },
96             m_time{ meth.m_time },
97             //m_solution{ meth.m_solution },
98             m_size{ meth.m_size },
99             m_N{ meth.m_N },
100             m_Ns{ meth.m_Ns },
101             m_nums{ meth.m_nums }
102         {};
103         FEMethod& operator=(const FEMethod& fem)
104         {
105             m_Grid = fem.m_Grid->Clone();
106             m_time = fem.m_time;
107             m_size = fem.m_size;
108             m_N = fem.m_N;
109             m_Ns = fem.m_Ns;
110             m_nums = fem.m_nums;
111             return *this;
112         }
113         void Discretization();
114         const double GetValue(const Mesh::Point&) const;
115         const double GetValue(const Mesh::Point&, const std::vector<double>& vec)
116     const;
117         const double GetValue(const Mesh::Point&, const std::vector<double>& vec,
118     const int num) const;

```

```

116         //const Mesh::Point          GetGradValue(const Mesh::Point&, const std::vector<double>& vec)
const;
117         //const Mesh::Point          GetLambdaGrad(const Mesh::Point&, const std::vector<double>&
vec) const;
118         const double                  GetEffective(const std::vector<double>& vec) const;
119         void                          ProjectSolution(std::vector<double>&, std::function<const
double(const Mesh::Point&, const std::vector<double>&, const int)> GetValue, std::vector<double>&
sol);
120         void                          ProjectSolution(std::vector<double>&, std::function<const
double(const Mesh::Point&, const std::vector<double>&)> GetValue, std::vector<double>& sol, const
int);
121         void                          LoadSolution(const std::vector<double>& vec);
122         const std::vector<double>      SetSolution(const int sol, const int liq, const double, const
double, const double);
123         void                          GetSolution(std::vector<double>& vec);
124         void                          Rediscretization(const std::shared_ptr<Grid>&);
125         void                          Rediscretization();
126         void                          SetTimeStep(const double& step) { m_step = step; m_time = step;
}
127         Matrix*                      GetGlobalMatrix() const;
128         Grid*                        GetMesh() { return m_Grid; }
129         const std::vector<double>      GetRightVector() const;
130         void                          OutDatFormat(const Mesh::Point& min, const Mesh::Point& max,
const std::string& file_name, const std::vector<double>& vec) const;
131         void                          OutMeshFormat(const std::string& file_name, const
std::vector<double>& vec);
132         void                          OutMeshTimeFormat(const std::string& file_name, const
std::vector<double>& vec);
133         static const double            GetSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p);
134         static const double            GetSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p, const int nfem);
135         static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p);
136         static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p, const int n);
137         ~FEMethod();
138     private:
139         void                          GeneratePortrait();
140         void                          AssemblGlobal();
141         void                          MainConditions();
142         void                          SecondConditions();
143         void                          ThirdConditions();
144         void                          StefanConditions();
145         void                          ApplySources();
146         const int                     AssembleLocalMatrix(const int);
147         Grid*                        m_Grid = nullptr;
148         Matrix*                      m_GlobalMatrix = nullptr;
149         Matrix*                      m_RightMatrix = nullptr;
150         Problem*                     m_problem = nullptr;
151         std::vector<double>           m_solution;
152         std::vector<double>*          m_rhsvector;
153         unsigned int                  m_size;
154         double                        m_step{ 0.1 };
155         double                        m_time{ 0.1 };
156         unsigned int                  m_N;
157         unsigned int                  m_Ns;
158         std::vector<unsigned int>      m_nums;
159
160     };
161
162     template<class Problem, class Grid, class Matrix>
163     void FEMethod<Problem, Grid, Matrix>::Discretization()
164     {
165         GeneratePortrait();
166         AssemblGlobal();
167         //ApplySources();
168         //SecondConditions();
169         //ThirdConditions();
170         MainConditions();
171         //StefanConditions();
172     }
173     template<class Problem, class Grid, class Matrix>
174     void FEMethod<Problem, Grid, Matrix>::GeneratePortrait()
175     {
176         const auto& el = m_Grid->GetElement(0);
177         int order = m_Grid->GetElement(0)->GetDoFs();
178         std::vector<std::set<unsigned int> temp;
179         m_Ns = m_Grid->GetNumberOfNodes();
180         m_N = m_Grid->GetNumberOfElements();
181         temp.resize(m_Grid->GetNumberOfNodes());
182         unsigned i, j, k;
183         for (k = 0; k < m_N; ++k)
184         {
185             const auto& elem{ m_Grid->GetElement(k) };
186             for (i = 0; i < order; ++i)
187                 for (j = 0; j < order; ++j)

```



```

188         if (elem->GetNode(j) > elem->GetNode(i))
189             temp[elem->GetNode(j)].insert(elem->GetNode(i));
190     }
191     if(m_problem->findTerm(Terms::RUV))
192         m_RightMatrix->Create(temp.size(), temp);
193
194     //m_GlobalMatrix = std::shared_ptr<Matrix>(new Matrix(m_Grid->GetNumberOfNodes(), temp));
195     //m_rhsvector.resize(m_Grid->GetNumberOfNodes());
196     //std::cout << temp.size() << std::endl;
197     m_GlobalMatrix->Create(temp.size(), temp);
198     m_rhsvector->resize(temp.size());
199     //m_solution.resize(m_Grid->GetNumberOfNodes());
200     //for (int l = 0; l < m_Grid->GetNumberOfNodes(); ++l)
201     //    m_solution[l] = 20;
202 }
203 template<class Problem, class Grid, class Matrix>
204 void FEMethod<Problem, Grid, Matrix>::AssemblGlobal()
205 {
206     int l;
207     //std::vector<std::future<int>> futures;
208     for (l = 0; l < m_N; ++l)
209         //futures.push_back(async(&DGMMethod<Problem, Grid, Matrix>::AssembleLocalMatrix, this,
210 l));
211         AssembleLocalMatrix(l);
212     //for (auto &it : futures)
213     //    //it.get();
214 }
215 template<class Problem, class Grid, class Matrix>
216 const int FEMethod<Problem, Grid, Matrix>::AssembleLocalMatrix(const int l)
217 {
218     int i, j, k, nodes;
219     double mij;
220     const auto& elem{ m_Grid->GetElement(l) };
221     const int dofs{ (int)elem->GetDoFs() };
222     const int terms{ (int)m_problem->getNumberOfTerms() };
223     nodes = elem->GetNumberOfNodes();
224     std::vector<Mesh::Point> points(nodes);
225     for (i = 0; i < nodes; ++i)
226         points[i] = m_Grid->GetNode(elem->GetNode(i));
227     for (k = 0; k < terms; ++k)
228     {
229         switch (m_problem->getTerm(k))
230         {
231             case Terms::IUV:
232                 for (i = 0; i < (int)dofs; ++i)
233                 {
234                     for (j = 0; j < (int)dofs; ++j)
235                     {
236                         auto M = [&](const Mesh::Point& p)
237                         {
238                             return m_problem->get_parameter(Terms::IUV, l, elem->GetType(), p) *
239                             elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
240                         };
241                         mij = elem->Integrate(M, points);
242                         m_GlobalMatrix->AddElement(elem->GetNode(i), elem->GetNode(j), mij);
243                     }
244                 }
245                 break;
246             case Terms::IDUDV:
247                 for (i = 0; i < (int)dofs; ++i)
248                 {
249                     for (j = 0; j < (int)dofs; ++j)
250                     {
251                         auto M = [&](const Mesh::Point& p)
252                         {
253                             //auto m = elem->GetGradShapeFunction(i, p) *
254                             elem->GetGradShapeFunction(j, p);
255                             return m_problem->get_parameter(Terms::IDUDV, l, elem->GetType(), p) *
256                             elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
257                         };
258                         //mij = m_Grid->getParameter(Parameters::DIFFUSION, l, j) *
259                         elem->Integrate(M, points);
260                         mij = elem->Integrate(M, points);
261                         m_GlobalMatrix->AddElement(elem->GetNode(i), elem->GetNode(j), mij);
262                     }
263                 }
264                 break;
265             case Terms::IDUV:
266                 for (i = 0; i < (int)dofs; ++i)
267                 {
268                     for (j = 0; j < (int)dofs; ++j)
269                     {
270                         auto M = [&](const Mesh::Point& p)
271                         {
272                             return m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0) *
273                             elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
274                         };
275                     }
276                 }
277             }
278     }
279 }

```

```

269         auto _mij = elem->Integrate(M, points);
270         m_GlobalMatrix->AddElement(elem->GetNode(i), elem->GetNode(j), _mij);
271     }
272 }
273 break;
274 case Terms::IUDV:
275     for (i = 0; i < dofs; ++i)
276     {
277         for (j = 0; j < dofs; ++j)
278         {
279             auto M = [&](const Mesh::Point& p)
280             {
281                 return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
282             };
283             //mij = m_CoarseGrid->getParameter(Parameters::ADVECTION, 1, j) *
m_flux(m_CoarseGrid->getSolution(1, j)) * elem->Integrate(M, points).x;
284             mij = elem->Integrate(M, points).x;
285             m_GlobalMatrix->AddElement(elem->GetNode(i), elem->GetNode(j), mij);
286         }
287     }
288 break;
289 case Terms::EUV:
290     for (i = 0; i < dofs; ++i)
291     {
292         for (j = 0; j < dofs; ++j)
293         {
294             auto M = [&](const Mesh::Point& p)
295             {
296                 return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
297             };
298             mij = elem->Integrate(M, points);
299             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::MASS, 1, j) * m_Grid->getSolution(1, j) * mij;
300             //m_rhsvector->operator[] (m_nums[1] + i) +=
m_CoarseGrid->getParameter(Parameters::MASS, 1, points[j]) * elem->GetValue(j) * mij;
301         }
302     }
303 break;
304 case Terms::EDUDV:
305     for (i = 0; i < dofs; ++i)
306     {
307         for (j = 0; j < dofs; ++j)
308         {
309             auto M = [&](const Mesh::Point& p)
310             {
311                 return elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
312 p);
313             };
314             mij = elem->Integrate(M, points);
315             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::DIFFUSION, 1, j) * m_Grid->getSolution(1, j) * mij;
316         }
317     }
318 break;
319 case Terms::EDUV:
320     for (i = 0; i < dofs; ++i)
321     {
322         for (j = 0; j < dofs; ++j)
323         {
324             auto M = [&](const Mesh::Point& p)
325             {
326                 return elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
327             };
328             mij = elem->Integrate(M, points).x;
329             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, 1, j) * mij;
330         }
331     }
332 break;
333 case Terms::EUDV:
334     for (i = 0; i < dofs; ++i)
335     {
336         for (j = 0; j < dofs; ++j)
337         {
338             auto M = [&](const Mesh::Point& p)
339             {
340                 return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
341             };
342             mij = elem->Integrate(M, points).x;
343             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, 1, j) * mij; // *mij;
344         }
345     }
346 break;
347 case Terms::EFV:
348     for (i = 0; i < dofs; ++i)
349     {

```

```

349         /*for (j = 0; j < dofs; ++j)
350         {
351             auto M = [&](const Mesh::Point& p)
352             {
353                 return m_problem->get_parameter(Terms::EFV, elem->GetType(), 1, j, p) *
elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
354             };
355             mij = elem->Integrate(M, points);
356             m_rhsvector->operator[] (elem->GetNode(i)) += mij;
357         }*/
358         auto M = [&](const Mesh::Point& p)
359         {
360             return m_problem->get_parameter(Terms::EFV, elem->GetType(), 1, i, p) *
elem->GetShapeFunction(i, p);
361         };
362         mij = elem->Integrate(M, points);
363         m_rhsvector->operator[] (elem->GetNode(i)) += mij;
364     }
365     break;
366     case Terms::RUV:
367         for (i = 0; i < (int)dofs; ++i)
368         {
369             for (j = 0; j < (int)dofs; ++j)
370             {
371                 auto M = [&](const Mesh::Point& p)
372                 {
373                     return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
374                 };
375                 mij = elem->Integrate(M, points);
376                 m_RightMatrix->AddElement(elem->GetNode(i), elem->GetNode(j), mij);
377             }
378         }
379     break;
380     case Terms::SUPG:
381     {
382         for (i = 0; i < (int)dofs; ++i)
383         {
384             for (j = 0; j < (int)dofs; ++j)
385             {
386                 /*auto inode = m_Grid->interpolate(elem->GetNode(i));
387                 auto jnode = m_Grid->interpolate(elem->GetNode(j));
388                 if (inode == -1 || jnode == -1)
389                     continue;*/
390                 auto M = [&](const Mesh::Point& p)
391                 {
392                     double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
393                     double h = elem->GetMeasure();
394                     //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
395                     double tau = 0.;
396                     double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
397                     //double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) -
1.) - 1. / Pe);
398                     //double beta = h / std::sqrt(3.) * ((exp(2. * Pe) + 1.) / (exp(2. * Pe)
- 1.) - 1. / Pe);
399                     //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1.
/ Pe);
400                     //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1.
/ Pe);
401                     //beta = 0.;
402                     //for (int ii = 0; ii < (int)dofs; ++ii)
403                     //beta += m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * elem->GetGradShapeFunction(ii, p);
404                     //return beta * m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
405                     // elem->GetGradShapeFunction(i, p) *
elem->GetGradShapeFunction(j, p);
406                     if (Pe >= 1.)
407                         tau = h / 2. / vel;
408                     else
409                         tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
410                     //return 0.;
411                     return tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
412                 };
413             };
414         }
415         //double tau =
416         auto _mij = elem->Integrate(M, points);
417         m_GlobalMatrix->AddElement(elem->GetNode(i), elem->GetNode(j), _mij);
418     }
419     auto M = [&](const Mesh::Point& p)
420     {

```

```

421         double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
422         double h = elem->GetMeasure();
423         double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
424         double tau = 0.;
425         //double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
426
427         if (Pe >= 1.)
428             tau = h / 2. / vel;
429         else
430             tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
431         auto supg = tau * m_problem->get_parameter(Terms::EFV, elem->GetType(), 1,
i, p) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(i, p);
432
433         return supg;
434     };
435     mij = elem->Integrate(M, points);
436     m_rhsvector->operator[] (elem->GetNode(i)) += mij;
437 }
438 }
439     break;
440 default:
441     break;
442 }
443 }
444     return 0;
445 }
446 template<class Problem, class Grid, class Matrix>
447 void FEMethod<Problem, Grid, Matrix>::MainConditions()
448 {
449     double mu{ 1e10 };
450     const auto n = m_problem->get_number_of_boundaries();
451     const auto m = m_Grid->GetNumberOfBoundaries();
452     for (int i = 0; i < n; ++i)
453     {
454         const auto& type = m_problem->get_boundary_type(i);
455         for (int j = 0; j < m; ++j)
456         {
457             const auto& row = m_Grid->GetBoundary(j);
458             if (row->GetType() == type)
459             {
460                 const int dofs = (int)row->GetDoFs();
461                 const int dofs2 = 2;
462                 const auto& elem_num = row->GetNeighbour(0);
463                 const auto& elem = m_Grid->GetElement(elem_num);
464                 const int dofs_elem = elem->GetDoFs();
465                 std::vector<Mesh::Point> points(dofs_elem);
466                 for (int k = 0; k < dofs_elem; ++k)
467                     points[k] = m_Grid->GetNode(elem->GetNode(k));
468                 for (int k = 0; k < dofs; ++k)
469                 {
470                     int l = 0;
471                     for (; l < dofs_elem; ++l)
472                     {
473                         if (elem->GetNode(l) == row->GetNode(k))
474                             break;
475                     }
476                     m_GlobalMatrix->NullRow(row->GetNode(k));
477                     m_GlobalMatrix->operator() (row->GetNode(k), row->GetNode(k)) = 0;
478                     //m_GlobalMatrix->operator() (row->GetNode(k), row->GetNode(k)) *= mu;
479                     //m_rhsvector->operator[] (row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, m_Grid->GetNode(row->GetNode(k)));
480                     //m_rhsvector->operator[] (row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, elem_num, 1, m_Grid->GetNode(row->GetNode(k)));
481                     m_rhsvector->operator[] (row->GetNode(k)) = elem->GetWeight(1, points,
[=](const Mesh::Point& p) { return m_problem->get_boundary_parameter(0, type, p); });
482                     if (m_problem->findTerm(Terms::RUV))
483                     {
484                         m_RightMatrix->NullRow(row->GetNode(k));
485                         //m_RightMatrix->operator() (row->GetNode(k), row->GetNode(k)) *= mu;
486                     }
487                 }
488                 /*for (int k = dofs2; k < dofs; ++k)
489                 {
490                     m_GlobalMatrix->NullRow(row->GetNode(k));
491                     m_rhsvector->operator[] (row->GetNode(k)) = 0;
492                 }*/
493             }
494         }
495     }
496     /*for (auto bnd : m_Grid->GetBoundaryConditions())
497     {
498         if (get<0>(bnd.second) == 1)

```

```

499         for (auto row : m_Grid->GetBoundary())
500         {
501             if (bnd.first == row->GetType())
502             {
503                 for (int i = 0; i < row->GetDoF(); ++i)
504                 {
505                     m_GlobalMatrix->NullRow(row->GetNodes(i));
506                     m_rhsvector[row->GetNodes(i)] =
get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
507                 }
508             }
509         }
510     }*/
511 }
512 template<class Problem, class Grid, class Matrix>
513 void FEMethod<Problem, Grid, Matrix>::SecondConditions()
514 {
515     double theta = 0;
516     int nfem;
517     Mesh::Point temp[3];
518     std::vector<int> local;
519     for (auto bnd : m_Grid->GetBoundaryConditions())
520     {
521         //if (get<0>(bnd.second) == 2)
522         {
523             for (auto row : m_Grid->GetBoundary())
524             {
525                 if (bnd.first == row->GetType())
526                 {
527                     local.resize(0);
528                     int dofs = row->GetDoF();
529                     nfem = row->GetNumberOfElement(0);
530                     auto elem = m_Grid->GetElements()[nfem];
531                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
532                     for (int j = 0; j < dofs; ++j)
533                     {
534                         temp[j] = m_Grid->GetNodes() [row->GetNodes(j)];
535                         for (int i = 0; i < elem->GetDoF(); ++i)
536                         {
537                             if (row->GetNodes(j) == elem->GetNodes() [i])
538                             {
539                                 local.push_back(i);
540                                 break;
541                             }
542                         }
543                     }
544                     for (int i = 0; i < dofs; ++i)
545                     {
546                         for (int j = 0; j < dofs; ++j)
547                         {
548                             //theta = get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
549                             theta = 0;
550                             auto GetMass = [&](const Mesh::Point& p) {return
elem->GetBasis(local[j], p) * elem->GetBasis(local[i], p); };
551                             auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
552                             //if (i < 2 || j < 2)
553                             m_rhsvector[row->GetNodes(i)] += theta * row->Integrate(GetMass,
temp);
554
555                             //if (i < 3 || j < 3)
556                             // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
557                         }
558                     }
559                 }
560             }
561         }
562     }
563 }
564 template<class Problem, class Grid, class Matrix>
565 void FEMethod<Problem, Grid, Matrix>::StefanConditions()
566 {
567     double dest{ 0. }, lat{ 0 };
568     int nfem;
569     Mesh::Point temp[3];
570     std::vector<int> local;
571     for (auto bnd : m_Grid->GetBoundaryConditions())
572     {
573         //if (get<0>(bnd.second) == 4)
574         {
575             lat = 0;
576             //lat = get<2>(bnd.second);
577             for (auto row : m_Grid->GetBoundary())
578             {
579                 if (bnd.first == row->GetType())
580                 {

```

```

581         local.resize(0);
582         int dofs = row->GetDoF();
583         nfem = row->GetNumberOfElement(0);
584         auto elem = m_Grid->GetElements()[nfem];
585         //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
586         for (int j = 0; j < dofs; ++j)
587         {
588             temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
589             for (int i = 0; i < elem->GetDoF(); ++i)
590             {
591                 if (row->GetNodes(j) == elem->GetNodes()[i])
592                 {
593                     local.push_back(i);
594                     break;
595                 }
596             }
597         }
598         for (int i = 0; i < dofs; ++i)
599         {
600             for (int j = 0; j < dofs; ++j)
601             {
602                 dest = 0;
603                 //dest = get<1>(bnd.second)(m_Grid->GetNodes()[row->GetNodes(i)]);
604                 auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
605                 //if (i < 2 || j < 2)
606                 m_rhsvector[row->GetNodes(i)] += dest * lat *
row->Integrate(GetBBasis, temp);
607
608                 //if (i < 3 || j < 3)
609                 // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
610             }
611         }
612     }
613 }
614 }
615 }
616 }
617 template<class Problem, class Grid, class Matrix>
618 void FEMethod<Problem, Grid, Matrix>::ThirdConditions()
619 {
620     double param{ 0 }, beta{ 0 };
621     int nfem;
622     Mesh::Point temp[6];
623     std::vector<int> local;
624     auto fxy = [&](const Mesh::Point& p) {return (10 * p.y*m_time + m_time) / 10; };
625     //auto fxy = [&](const Point& p){return 10 * p.y + 10 * m_time; };
626     for (auto bnd : m_Grid->GetBoundaryConditions())
627     {
628         //if (get<0>(bnd.second) == 3)
629         {
630
631             for (auto row : m_Grid->GetBoundary())
632             {
633                 if (bnd.first == row->GetType())
634                 {
635                     local.resize(0);
636                     int dofs = row->GetDoF();
637                     nfem = row->GetNumberOfElement(0);
638                     auto elem = m_Grid->GetElements()[nfem];
639                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
640                     auto order = elem->GetDoF();
641                     for (int j = 0; j < dofs; ++j)
642                     {
643                         temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
644                         for (int i = 0; i < order; ++i)
645                         {
646                             if (row->GetNodes(j) == elem->GetNodes()[i])
647                             {
648                                 local.push_back(i);
649                                 break;
650                             }
651                         }
652                     }
653                     double val{ 0 };
654                     for (int i = 0; i < dofs; ++i)
655                     {
656                         for (int j = 0; j < dofs; ++j)
657                         {
658                             param = 0;
659                             beta = 0;
660                             //beta = get<2>(bnd.second);
661                             //param = get<1>(bnd.second)(m_Grid->GetNodes()[row->GetNodes(i)]);
662                             //param = fxy(temp[j]);
663                             auto GetBBasis = [&](const Mesh::Point& p) {return
elem->GetBasis(local[j], p)*elem->GetBasis(local[i], p); };

```

```

664                                     //val = row->GetElement(GetBBasis, temp);
665                                     val = row->Integrate(GetBBasis, temp);
666                                     m_GlobalMatrix->operator()(row->GetNodes(i), row->GetNodes(j)) +=
        beta * val;
        m_rhsvector[row->GetNodes(i)] += beta * param * val;
    }
}
}
}
}
}
}
}
template<class Problem, class Grid, class Matrix>
Matrix* FEMethod<Problem, Grid, Matrix>::GetGlobalMatrix() const
{
    return m_GlobalMatrix;
}
template<class Problem, class Grid, class Matrix>
const double FEMethod<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p) const
{
    if (!m_solution.size())
        return -1;
    double val = 0;
    int nfem = -1;
    nfem = m_Grid->FindElement(p);
    if (nfem == -1)
        return -1;
    auto elem = m_Grid->GetElements()[nfem];
    for (int i = 0; i < elem->GetDoF(); ++i)
        val += m_solution[elem->GetNodes()[i]] * elem->GetBasis(i, p);
    return val;
}
template<class Problem, class Grid, class Matrix>
const double FEMethod<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
std::vector<double>& vec) const
{
    if (!vec.size())
        return -1;
    double val{ 0 };
    int nfem{ -1 };
    nfem = m_Grid->FindElement(p);
    if (nfem == -1)
        return -1;
    auto elem = m_Grid->GetElements()[nfem];
    for (int i = 0; i < elem->GetDoFs(); ++i)
        val += vec[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
    return val;
}
template<class Problem, class Grid, class Matrix>
const double FEMethod<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
std::vector<double>& vec, const int num) const
{
    if (!vec.size() || num < 0)
        return -1;
    double val{ 0 };
    auto elem = m_Grid->GetElements()[num];
    for (int i = 0; i < elem->GetDoF(); ++i)
        val += vec[elem->GetNodes()[i]] * elem->GetBasis(i, p);
    return val;
}
//template<class Problem, class Grid, class Matrix>
//const Mesh::Point FEMethod<Problem, Grid, Matrix>::GetGradValue(const Mesh::Point& p, const
std::vector<double>& vec) const
//{
//    Mesh::Point val{ 0, 0 };
//    int nfem{ -1 };
//    nfem = m_Grid->FindElement(p);
//    if (nfem == -1)
//        return val;
//    auto elem = m_Grid->GetElements()[nfem];
//    for (int i = 0; i < elem->GetDoF(); ++i)
//    {
//        val.x += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).x;
//        val.y += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).y;
//        val.z += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).z;
//    }
//    return val;
//}
template<class Problem, class Grid, class Matrix>
const double FEMethod<Problem, Grid, Matrix>::GetEffective(const std::vector<double>& vec) const
{
    double sum = 0;
    //std::vector<int> dofs;
    //Mesh::Point points[10];
    //for (int i = 0; i < m_Grid->GetElements().size(); ++i)
    //{
        //auto mb = [&](const Mesh::Point& b) {return GetGradValue(b, vec)*GetGradValue(b, vec);

```

```

    };

747         //dofs.resize(0);
748         //auto elem = m_Grid->GetElements()[i];
749         //int order = elem->GetDoF();
750         //double diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
751         //for (int j = 0; j < order; ++j)
752         //{
753             //dofs.push_back(elem->GetNodes()[j]);
754             //points[j] = m_Grid->GetNodes()[dofs[j]];
755         //}
756         //sum += diff * elem->Integrate(mb, points);
757     //}
758     //std::cout << "Effect (local): " << sum << std::endl;
759     //std::cout << "Effect (local) sqrt: " << sqrt(sum) << std::endl;
760     return sum;
761 }
762 //template<class Problem, class Grid, class Matrix>
763 //const Mesh::Point FEMethod<Problem, Grid, Matrix>::GetLambdaGrad(const Mesh::Point& p, const
std::vector<double>& vec) const
764 //{
765     // Mesh::Point val{ 0, 0, 0 };
766     // //double val{ 0 };
767     // double diff{ 0 };
768     // Mesh::Point temp{ 0, 0, 0 };
769     // int nfem{ -1 };
770     // nfem = m_Grid->FindElement(p);
771     // if (nfem == -1)
772     //     return val;
773     // auto elem = m_Grid->GetElements()[nfem];
774     // diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
775     // for (int i = 0; i < elem->GetDoF(); ++i)
776     // {
777         // //val += elem->GetGradBasis(i, p) * elem->GetGradBasis(i, p) * vec[elem->GetNodes()[i]]
* vec[elem->GetNodes()[i]] * diff;
778         // //val += elem->GetBasis(i, p) * vec[elem->GetNodes()[i]] * diff;
779         // temp = elem->GetGradBasis(i, p);
780         // val.x += temp.x * vec[elem->GetNodes()[i]] * (diff);
781         // val.y += temp.y * vec[elem->GetNodes()[i]] * (diff);
782         // val.z += temp.z * vec[elem->GetNodes()[i]] * (diff);
783     // }
784     // return val;
785 //}
786 template<class Problem, class Grid, class Matrix>
787 const std::vector<double> FEMethod<Problem, Grid, Matrix>::GetRightVector() const
788 {
789     return *m_rhsvector;
790 }
791 template<class Problem, class Grid, class Matrix>
792 void FEMethod<Problem, Grid, Matrix>::OutDatFormat(const Mesh::Point& mn, const Mesh::Point& mx,
const std::string& file_name, const std::vector<double>& vec) const
793 {
794     std::ofstream of(file_name + ".z.dat");
795     std::streambuf *buf = std::cout.rdbuf();
796     std::cout.rdbuf(of.rdbuf());
797     std::cout << "TITLE = FE-METHOD\n";
798     std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
799     std::cout << "ZONE i=51, j=51, F=POINT\n";
800     double stepx = (mx.x - mn.x) / 51;
801     double stepy = (mx.y - mn.y) / 51;
802     for (int i = 0; i < 51; ++i)
803         for (int j = 0; j < 51; ++j)
804             std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.x + j * stepx, mn.y + i * stepy, mn.z), vec) << std::endl;
805     std::cout.rdbuf(buf);
806     of.close();
807     of.open(file_name + ".x.dat");
808     buf = std::cout.rdbuf();
809     std::cout.rdbuf(of.rdbuf());
810     std::cout << "TITLE = FE-METHOD\n";
811     std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
812     std::cout << "ZONE i=51, j=51, F=POINT\n";
813     for (int i = 0; i < 51; ++i)
814         for (int j = 0; j < 51; ++j)
815             std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.z, mn.x + j * stepx, mn.y + i * stepy), vec) << std::endl;
816     std::cout.rdbuf(buf);
817     of.close();
818     of.open(file_name + ".y.dat");
819     buf = std::cout.rdbuf();
820     std::cout.rdbuf(of.rdbuf());
821     std::cout << "TITLE = FE-METHOD\n";
822     std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
823     std::cout << "ZONE i=51, j=51, F=POINT\n";
824     for (int i = 0; i < 51; ++i)
825         for (int j = 0; j < 51; ++j)
826             std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.x + j * stepx, mn.z, mn.y + i * stepy), vec) << std::endl;

```



```

827         std::cout.rdbuf(buf);
828         of.close();
829     }
830     template<class Problem, class Grid, class Matrix>
831     void FEMethod<Problem, Grid, Matrix>::ApplySources()
832     {
833         int nfem = -1;
834         auto total = m_problem->get_total_sources();
835         for (int i = 0; i < total; ++i)
836         {
837             auto src = m_problem->get_point_source(i);
838             auto point = src.get_point();
839             nfem = m_Grid->FindElement(point);
840             if (nfem != -1)
841             {
842                 auto val = src.get_value();
843                 auto elem = m_Grid->GetElement(nfem);
844                 for (int j = 0; j < 3; ++j)
845                     m_rhsvector->operator[] (elem->GetNode(j)) += val * elem->GetShapeFunction(j,
point);
846             }
847             nfem = -1;
848         }
849         /*for (auto srd : m_Grid->GetDottedSources())
850         {
851             nfem = m_Grid->FindElement(srd.first);
852             if (nfem != -1)
853             {
854                 auto elem = m_Grid->GetElements()[nfem];
855                 for (int i = 0; i < elem->GetDoF(); ++i)
856                 {
857                     m_rhsvector[elem->GetNodes()[i]] += srd.second * elem->GetBasis(i, srd.first);
858                 }
859             }
860             nfem = -1;
861         }*/
862     }
863     template<class Problem, class Grid, class Matrix>
864     void FEMethod<Problem, Grid, Matrix>::Rediscretization(const std::shared_ptr<Grid>& grid)
865     {
866         m_GlobalMatrix->NullMatrix();
867         for (unsigned int i = 0; i < m_rhsvector->size(); ++i)
868             (*m_rhsvector)[i] = 0;
869         AssemblGlobal();
870         //SecondConditions();
871         //ApplySources();
872         //StefanConditions();
873         MainConditions();
874     }
875     template<class Problem, class Grid, class Matrix>
876     void FEMethod<Problem, Grid, Matrix>::Rediscretization()
877     {
878         m_time += m_step;
879         m_GlobalMatrix->NullMatrix();
880         for (unsigned int i = 0; i < m_rhsvector->size(); ++i)
881             (*m_rhsvector)[i] = 0;
882         AssemblGlobal();
883         SecondConditions();
884         ThirdConditions();
885         StefanConditions();
886         //ApplySources();
887         MainConditions();
888     }
889     template<class Problem, class Grid, class Matrix>
890     void FEMethod<Problem, Grid, Matrix>::GetSolution(std::vector<double>& vec)
891     {
892         int size = vec.size();
893         //Translation(vec);
894         for (int i = 0; i < size; ++i)
895             vec[i] = m_solution[i];
896     }
897     template<class Problem, class Grid, class Matrix>
898     const double FEMethod<Problem, Grid, Matrix>::GetSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
899     {
900         double sum{ 0 };
901         auto nfem{ g.FindElement(p) };
902         if (nfem < 0)
903             return 0.;
904         auto elem{ g.GetElement(nfem) };
905         auto dofs{ elem->GetDoFs() };
906         for (auto i{ 0 }; i < dofs; ++i)
907             sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
908         return sum;
909     }
910     template<class Problem, class Grid, class Matrix>
911     const double FEMethod<Problem, Grid, Matrix>::GetSolution(const Grid& g, const

```

```

std::vector<double> &weights, const Mesh::Point& p, const int nfem)
912 {
913     double sum{ 0 };
914     //if (nfem < 0)
915     //    return 0.;
916     auto elem{ g.GetElement(nfem) };
917     auto dofs{ elem->GetDoFs() };
918     //std::cout << nfem << std::endl;
919     for (auto i{ 0 }; i < dofs; ++i)
920         sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
921     return sum;
922 }
923 template<class Problem, class Grid, class Matrix>
924 const Mesh::Point FEMethod<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
925 {
926     Mesh::Point sum{ 0, 0, 0 };
927     auto nfem{ g.FindElement(p) };
928     auto elem{ g.GetElement(nfem) };
929     auto dofs{ elem->GetDoFs() };
930     for (auto i{ 0 }; i < dofs; ++i)
931         sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
932     return sum;
933 }
934 template<class Problem, class Grid, class Matrix>
935 const Mesh::Point FEMethod<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p, const int nfem)
936 {
937     Mesh::Point sum{ 0, 0, 0 };
938     auto elem{ g.GetElement(nfem) };
939     auto dofs{ elem->GetDoFs() };
940     for (auto i{ 0 }; i < dofs; ++i)
941         sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
942     return sum;
943 }
944 template<class Problem, class Grid, class Matrix>
945 void FEMethod<Problem, Grid, Matrix>::LoadSolution(const std::vector<double>& vec)
946 {
947     m_solution.resize(vec.size());
948     for (unsigned int i = 0; i < vec.size(); ++i)
949         m_solution[i] = vec[i];
950 }
951 template<class Problem, class Grid, class Matrix>
952 void FEMethod<Problem, Grid, Matrix>::OutMeshFormat(const std::string& file_name, const
std::vector<double>& vec)
953 {
954     const int size{ (int)m_Grid->GetNodes().size() };
955     const int number{ (int)m_Grid->GetElements().size() };
956     //const int size{ number * 4 };
957     std::ofstream ofs(file_name + ".dat", std::ios::out);
958     std::string title("TITLE = \\Mesh data\\n Variables = \\X\\", \\Y\\", \\Z\\", \\U\\n Zone N =
" + std::to_string(size) + ", E = " + std::to_string(number) + ", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\\n");
959     ofs << title;
960     Mesh::Point p;
961     for (int i = 0; i < size; ++i)
962     {
963         p = m_Grid->GetNodes()[i];
964         ofs << "p.x << "\\t" << p.y << "\\t" << p.z << "\\t" << GetValue(p, vec, 1) << std::endl;
965     }
966     for (int i = 0; i < number; ++i)
967     {
968         auto elem = m_Grid->GetElements()[i];
969         for (int k = 0; k < 4; ++k)
970         {
971             ofs << elem->GetNodes()[k] + 1 << "\\t";
972         }
973         ofs << std::endl;
974     }
975     ofs.close();
976 }
977 template<class Problem, class Grid, class Matrix>
978 void FEMethod<Problem, Grid, Matrix>::OutMeshTimeFormat(const std::string& file_name, const
std::vector<double>& vec)
979 {
980     const int size{ (int)m_Grid->GetNodes().size() };
981     const int number{ (int)m_Grid->GetElements().size() };
982     //const int size{ number * 4 };
983     std::ofstream ofs(file_name + ".dat", std::ios::out | std::ios::app);
984     std::string title("TITLE = \\Mesh data\\n Variables = \\X\\", \\Y\\", \\Z\\", \\U\\n Zone N =
" + std::to_string(size) + ", E = " + std::to_string(number) + ", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\\n");
985     ofs << title;
986     Mesh::Point p;
987     for (int i = 0; i < size; ++i)
988     {
989         p = m_Grid->GetNodes()[i];

```

```

990         ofs << "p.x << "\t" << p.y << "\t" << p.z << "\t" << GetValue(p, vec, 1) << std::endl;
991     }
992     for (int i = 0; i < number; ++i)
993     {
994         auto elem = m_Grid->GetElements()[i];
995         for (int k = 0; k < 4; ++k)
996         {
997             ofs << elem->GetNodes()[k] + 1 << "\t";
998         }
999         ofs << std::endl;
1000     }
1001     ofs.close();
1002 }
1003 template<class Problem, class Grid, class Matrix>
1004 void FEMethod<Problem, Grid, Matrix>::ProjectSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&, const int)> GetVal,
std::vector<double>& vec)
1005 {
1006     for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1007     {
1008         auto elem = m_Grid->GetElements()[i];
1009         int order = elem->GetDoF();
1010         for (int j = 0; j < order; ++j)
1011             sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes(j)], vec, i);
1012     }
1013 }
1014 template<class Problem, class Grid, class Matrix>
1015 void FEMethod<Problem, Grid, Matrix>::SetSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&)> GetVal,
std::vector<double>& vec, const int)
1016 {
1017     for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1018     {
1019         auto elem = m_Grid->GetElements()[i];
1020         int order = elem->GetDoF();
1021         for (int j = 0; j < order; ++j)
1022             sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes(j)], vec);
1023     }
1024 }
1025 template<class Problem, class Grid, class Matrix>
1026 const std::vector<double> FEMethod<Problem, Grid, Matrix>::SetSolution(const int sol, const int
liq, const double s, const double l, const double m)
1027 {
1028     int i;
1029     m_solution.resize(m_Grid->GetNodes().size());
1030     for (i = 0; i < m_Grid->GetElements().size(); ++i)
1031     {
1032         auto elem = m_Grid->GetElements()[i];
1033         int order = elem->GetDoF();
1034         if (m_Grid->GetElements()[i]->GetType() == liq)
1035             for (int j = 0; j < order; ++j)
1036                 m_solution[elem->GetNodes()[j]] = 1;
1037         else
1038             for (int j = 0; j < order; ++j)
1039                 m_solution[elem->GetNodes()[j]] = s;
1040     }
1041 }
1042 for (auto bnd : m_Grid->GetBoundaryConditions())
1043 {
1044     //if (get<0>(bnd.second) == 4)
1045     {
1046         for (auto row : m_Grid->GetBoundary())
1047         {
1048             if (bnd.first == row->GetType())
1049             {
1050                 int dofs = row->GetDoF();
1051                 for (int i = 0; i < dofs; ++i)
1052                 {
1053                     m_solution[row->GetNodes(i)] = m;
1054                 }
1055             }
1056         }
1057     }
1058 }
1059 return m_solution;
1060 }
1061 template<class Problem, class Grid, class Matrix>
1062 FEMethod<Problem, Grid, Matrix>::~FEMethod()
1063 {
1064     delete m_Grid;
1065 }
1066 }
1067 }
1068
1069 #endif // !CORENC_METHODS_FEMethod_h
1070

```

7.81 CoreNCFEM/Methods/FEMethodZero.h File Reference

```
#include <functional>
#include <set>
#include "../Point.h"
#include "../Parameter.h"
#include "CSMethod.h"
#include <memory>
#include <cmath>
#include <map>
#include <algorithm>
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
```

Classes

- class [corenc::method::CFEMethodZero< Type >](#)
- class [corenc::method::FEMethodZero< Problem, Grid, Matrix >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)
- namespace [corenc::method](#)

7.82 FEMethodZero.h

[Go to the documentation of this file.](#)

```
1 // FEMethodZero.h describes an abstract interface and functions for a general finite element method with
   zero Dirichlet boundaries
2 #ifndef CORENC_METHODS_FEMethodZeroZero_h
3 #define CORENC_METHODS_FEMethodZeroZero_h
4 #include <functional>
5 #include <set>
6 #include "../Point.h"
7 #include "../Parameter.h"
8 #include "CSMethod.h"
9 #include <memory>
10 #include <cmath>
11 #include <map>
12 #include <algorithm>
13 #include <vector>
14 #include <iostream>
15 #include <fstream>
16 #include <string>
17 namespace corenc
18 {
19     namespace Mesh
20     {
21         class Point;
22     }
23     namespace method
24     {
25         // class Type = Type of the solution, for ex vector or double, or even more specific
26
27         template<class Type>
28         class CFEMethodZero
29         {
30         public:
```

```

32         CFEMethodZero() {};
33         virtual ~CFEMethodZero() {};
34         virtual const int Assemble() = 0;
35         virtual const Type GetSolution(const std::vector<double>& point)
const = 0;
36         virtual const std::vector<Type> GetSolution() const = 0;
37         virtual const Type GetMaxSolution() const = 0;
38         virtual const Type GetMinSolution() const = 0;
39     };
40
41     template<class Problem, class Grid, class Matrix>
42     class FEMethodZero
43     {
44     public:
45         FEMethodZero() :
46             m_problem{nullptr},
47             m_Grid{nullptr},
48             m_GlobalMatrix{nullptr},
49             m_RightMatrix{nullptr},
50             m_rhsvector{nullptr}
51         {}
52         FEMethodZero(
53             Problem* p,
54             Grid* g,
55             Matrix* m,
56             std::vector<double>* rhs):
57             m_problem{ p },
58             m_Grid{ g->Clone() },
59             m_GlobalMatrix{ m },
60             m_N{ g->GetNumberOfElements() },
61             m_Ns{ g->GetNumberOfBoundaries() },
62             m_rhsvector{ rhs }{
63             //GeneratePortrait();
64         }
65         FEMethodZero(
66             Problem* p,
67             Grid* g,
68             Matrix* m,
69             Matrix* rm,
70             std::vector<double>* rhs):
71             m_problem{ p },
72             m_Grid{ g->Clone() },
73             m_GlobalMatrix{ m },
74             m_RightMatrix{ rm },
75             m_N{ g->GetNumberOfElements() },
76             m_Ns{ g->GetNumberOfBoundaries() },
77             m_rhsvector{ rhs }{
78             //GeneratePortrait();
79         }
80         FEMethodZero(const std::shared_ptr<Grid>& grid) :m_Grid{ grid->Clone() } {}
81         FEMethodZero(Grid* grid) :m_Grid{ grid->Clone() } {}
82         FEMethodZero(const FEMethodZero& meth) :
83             m_Grid{ meth.m_Grid->Clone() },
84             //m_GlobalMatrix{ meth.m_GlobalMatrix->Clone() },
85             //m_rhsvector{ meth.m_rhsvector },
86             //m_problem{ meth.m_problem },
87             m_time{ meth.m_time },
88             //m_solution{ meth.m_solution },
89             m_size{ meth.m_size },
90             m_N{ meth.m_N },
91             m_Ns{ meth.m_Ns },
92             m_nums{ meth.m_nums }
93         {};
94         void Discretization();
95         const double GetValue(const Mesh::Point&) const;
96         const double GetValue(const Mesh::Point&, const std::vector<double>& vec)
const;
97         const double GetValue(const Mesh::Point&, const std::vector<double>& vec,
const int num) const;
98         //const Mesh::Point GetGradValue(const Mesh::Point&, const std::vector<double>& vec)
const;
99         //const Mesh::Point GetLambdaGrad(const Mesh::Point&, const std::vector<double>& vec)
const;
100         const double GetEffective(const std::vector<double>& vec) const;
101         void ProjectSolution(std::vector<double>&, std::function<const
double(const Mesh::Point&, const std::vector<double>&, const int)> GetValue, std::vector<double>&
sol);
102         void ProjectSolution(std::vector<double>&, std::function<const
double(const Mesh::Point&, const std::vector<double>&)> GetValue, std::vector<double>& sol, const
int);
103         void LoadSolution(const std::vector<double>& vec);
104         const std::vector<double> SetSolution(const int sol, const int liq, const double, const
double, const double);
105         void GetSolution(std::vector<double>& vec);
106         void Rediscretization(const std::shared_ptr<Grid>&);
107         void Rediscretization();
108         void SetTimeStep(const double& step) { m_step = step; m_time = step;

```

```

    }
109     Matrix*          GetGlobalMatrix() const;
110     Grid*            GetMesh() { return m_Grid; }
111     const std::vector<double> GetRightVector() const;
112     void              OutDatFormat(const Mesh::Point& min, const Mesh::Point& max,
const std::string& file_name, const std::vector<double>& vec) const;
113     void              OutMeshFormat(const std::string& file_name, const
std::vector<double>& vec);
114     void              OutMeshTimeFormat(const std::string& file_name, const
std::vector<double>& vec);
115     static const double GetSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p);
116     static const double GetSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p, const int nfem);
117     static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p);
118     static const Mesh::Point GetGradSolution(const Grid& g, const std::vector<double> &weights,
const Mesh::Point& p, const int n);
119     ~FEMethodZero();
120 private:
121     void          GeneratePortrait();
122     void          AssemblGlobal();
123     void          MainConditions();
124     void          SecondConditions();
125     void          ThirdConditions();
126     void          StefanConditions();
127     void          ApplySources();
128     const int      AssembleLocalMatrix(const int);
129     const int      AssembleIDUDVMatrix(const int);
130     const int      AssembleIDUVMMatrix(const int);
131     const int      AssembleIUDVMatrix(const int);
132     const int      AssembleRUVMatrix(const int);
133     const int      AssembleSUPGMatrix(const int);
134     const int      AssembleLocalMatrix(const int, const int);
135     Grid*          m_Grid = nullptr;
136     Matrix*        m_GlobalMatrix = nullptr;
137     Matrix*        m_RightMatrix = nullptr;
138     Problem*       m_problem = nullptr;
139     std::vector<double> m_solution;
140     std::vector<double>* m_rhsvector;
141     unsigned int      m_size;
142     double            m_step{ 0.1 };
143     double            m_time{ 0.1 };
144     unsigned int      m_N;
145     unsigned int      m_Ns;
146     std::vector<unsigned int> m_nums;
147
148 };
149
150 template<class Problem, class Grid, class Matrix>
151 void FEMethodZero<Problem, Grid, Matrix>::Discretization()
152 {
153     GeneratePortrait();
154     AssemblGlobal();
155     //ApplySources();
156     //SecondConditions();
157     //ThirdConditions();
158     //MainConditions();
159     //StefanConditions();
160 }
161 template<class Problem, class Grid, class Matrix>
162 void FEMethodZero<Problem, Grid, Matrix>::GeneratePortrait()
163 {
164     const auto& el = m_Grid->GetElement(0);
165     int order = m_Grid->GetElement(0)->GetDoFs();
166     std::vector<std::set<unsigned int>> temp;
167     m_Ns = m_Grid->GetNumberOfINodes();
168     m_N = m_Grid->GetNumberOfElements();
169     temp.resize(m_Grid->GetNumberOfINodes());
170     unsigned i, j, k;
171     for (k = 0; k < m_N; ++k)
172     {
173         const auto& elem{ m_Grid->GetElement(k) };
174         for (i = 0; i < order; ++i)
175             for (j = 0; j < order; ++j)
176             {
177                 //std::cout << "inside" << std::endl;
178                 int jnode = m_Grid->interpolate(elem->GetNode(j));
179                 int inode = m_Grid->interpolate(elem->GetNode(i));
180                 //std::cout << jnode << "\t" << inode << std::endl;
181                 //std::cout << "outside" << std::endl;
182                 if (jnode > -1 && inode > -1)
183                     if (jnode > inode)
184                     {
185                         temp[jnode].insert(inode);
186                     }
187             }
188     }

```

```

188     }
189     if(m_problem->findTerm(Terms::RUV))
190         m_RightMatrix->Create(temp.size(), temp);
191
192     //m_GlobalMatrix = std::shared_ptr<Matrix>(new Matrix(m_Grid->GetNumberOfNodes(), temp));
193     //m_rhsvector.resize(m_Grid->GetNumberOfNodes());
194     //std::cout << temp.size() << std::endl;
195     m_GlobalMatrix->Create(temp.size(), temp);
196     m_rhsvector->resize(temp.size());
197     //m_solution.resize(m_Grid->GetNumberOfNodes());
198     //for (int l = 0; l < m_Grid->GetNumberOfNodes(); ++l)
199     //    m_solution[l] = 20;
200 }
201 template<class Problem, class Grid, class Matrix>
202 void FEMethodZero<Problem, Grid, Matrix>::AssemblGlobal()
203 {
204     int l;
205     //std::vector<std::future<int> > futures;
206     int i, j, k, nodes;
207     double mij;
208     const int terms{ (int)m_problem->getNumberOfTerms() };
209     for (k = 0; k < terms; ++k)
210     {
211         switch (m_problem->getTerm(k))
212         {
213             case Terms::IDUDV:
214                 for (l = 0; l < m_N; ++l)
215                     AssembleIDUDVMatrix(l);
216                 break;
217             case Terms::IDUV:
218                 for (l = 0; l < m_N; ++l)
219                     AssembleIDUVMatrix(l);
220                 break;
221             case Terms::IUDV:
222                 for (l = 0; l < m_N; ++l)
223                     AssembleIUDVMatrix(l);
224                 break;
225             case Terms::SUPG:
226                 for (l = 0; l < m_N; ++l)
227                     AssembleSUPGMatrix(l);
228                 break;
229             case Terms::RUV:
230                 for (l = 0; l < m_N; ++l)
231                     AssembleRUVMatrix(l);
232                 break;
233             default:
234                 break;
235         }
236     }
237     //for (l = 0; l < m_N; ++l)
238     //    futures.push_back(async(&DGMMethod<Problem, Grid, Matrix>::AssembleLocalMatrix, this,
239     //    // AssembleLocalMatrix(l, 0);
240     //    //for (auto &it : futures)
241     //    //it.get();
242     //);
243
244     template<class Problem, class Grid, class Matrix>
245     const int FEMethodZero<Problem, Grid, Matrix>::AssembleIDUDVMatrix(const int l)
246     {
247         int i, j, k, nodes;
248         double mij;
249         const auto& elem{ m_Grid->GetElement(l) };
250         const int dofs{ (int)elem->GetDoFs() };
251         const int terms{ (int)m_problem->getNumberOfTerms() };
252         nodes = elem->GetNumberOfNodes();
253         std::vector<Mesh::Point> points(nodes);
254         for (i = 0; i < nodes; ++i)
255             points[i] = m_Grid->GetNode(elem->GetNode(i));
256         for (i = 0; i < (int)dofs; ++i)
257         {
258             for (j = 0; j < (int)dofs; ++j)
259             {
260                 auto inode = m_Grid->interpolate(elem->GetNode(i));
261                 auto jnode = m_Grid->interpolate(elem->GetNode(j));
262                 if (inode == -1 || jnode == -1)
263                     continue;
264                 auto M = [&](const Mesh::Point& p)
265                 {
266                     //auto m = elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
267                     return m_problem->get_parameter(Terms::IDUDV, l, elem->GetType(), p) *
268                     elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
269                 };
270                 //mij = m_Grid->getParameter(Parameters::DIFFUSION, l, j) * elem->Integrate(M,
271                 //points);
272                 mij = elem->Integrate(M, points);
273                 m_GlobalMatrix->AddElement(inode, jnode, mij);

```

```

272     }
273 }
274 return 0;
275 }
276
277 template<class Problem, class Grid, class Matrix>
278 const int FEMethodZero<Problem, Grid, Matrix>::AssembleIDUVMatrix(const int l)
279 {
280     int i, j, k, nodes;
281     double mij;
282     const auto& elem{ m_Grid->GetElement(l) };
283     const int dofs{ (int)elem->GetDoFs() };
284     const int terms{ (int)m_problem->getNumberOfTerms() };
285     nodes = elem->GetNumberOfNodes();
286     std::vector<Mesh::Point> points(nodes);
287     for (i = 0; i < nodes; ++i)
288         points[i] = m_Grid->GetNode(elem->GetNode(i));
289     for (i = 0; i < (int)dofs; ++i)
290     {
291         for (j = 0; j < (int)dofs; ++j)
292         {
293             auto inode = m_Grid->interpolate(elem->GetNode(i));
294             auto jnode = m_Grid->interpolate(elem->GetNode(j));
295             if (inode == -1 || jnode == -1)
296                 continue;
297             auto M = [&](const Mesh::Point& p)
298             {
299                 return m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0) *
300                     elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
301             };
302             auto _mij = elem->Integrate(M, points);
303             m_GlobalMatrix->AddElement(inode, jnode, _mij);
304         }
305     }
306     return 0;
307 }
308
309 template<class Problem, class Grid, class Matrix>
310 const int FEMethodZero<Problem, Grid, Matrix>::AssembleIUDVMMatrix(const int l)
311 {
312     int i, j, k, nodes;
313     double mij;
314     const auto& elem{ m_Grid->GetElement(l) };
315     const int dofs{ (int)elem->GetDoFs() };
316     const int terms{ (int)m_problem->getNumberOfTerms() };
317     nodes = elem->GetNumberOfNodes();
318     std::vector<Mesh::Point> points(nodes);
319     for (i = 0; i < nodes; ++i)
320         points[i] = m_Grid->GetNode(elem->GetNode(i));
321     for (i = 0; i < dofs; ++i)
322     {
323         for (j = 0; j < dofs; ++j)
324         {
325             auto inode = m_Grid->interpolate(elem->GetNode(i));
326             auto jnode = m_Grid->interpolate(elem->GetNode(j));
327             if (inode == -1 || jnode == -1)
328                 continue;
329             auto M = [&](const Mesh::Point& p)
330             {
331                 return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
332             };
333             //mij = m_CoarseGrid->getParameter(Parameters::ADVECTION, l, j) *
334             m_flux(m_CoarseGrid->getSolution(l, j)) * elem->Integrate(M, points).x;
335             mij = elem->Integrate(M, points).x;
336             m_GlobalMatrix->AddElement(inode, jnode, mij);
337         }
338     }
339     return 0;
340 }
341
342 template<class Problem, class Grid, class Matrix>
343 const int FEMethodZero<Problem, Grid, Matrix>::AssembleRUVMatrix(const int l)
344 {
345     int i, j, k, nodes;
346     double mij;
347     const auto& elem{ m_Grid->GetElement(l) };
348     const int dofs{ (int)elem->GetDoFs() };
349     const int terms{ (int)m_problem->getNumberOfTerms() };
350     nodes = elem->GetNumberOfNodes();
351     std::vector<Mesh::Point> points(nodes);
352     for (i = 0; i < nodes; ++i)
353         points[i] = m_Grid->GetNode(elem->GetNode(i));
354     double minPec = -1;
355     auto MM = [&](const Mesh::Point& p)
356     {
357         double vel = sqrt(m_problem->get_parameter(Terms::IDUV, l, elem->GetType(), p, 0) *

```



```

    m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
357     double h = elem->GetMeasure();
358     //h = fabs(points[1].x - points[0].x);
359     h = sqrt(h);
360     //h = fabs(m_Grid->GetNode(0).x - m_Grid->GetNode(1).x);
361     double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(),
p);
362
363     //double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(),
p);
364
365     if (Pe > minPec)
366         minPec = Pe;
367     return 0.;
368 };
369 elem->Integrate(MM, points);
370 for (i = 0; i < (int)dofs; ++i)
371 {
372     for (j = 0; j < (int)dofs; ++j)
373     {
374         auto inode = m_Grid->interpolate(elem->GetNode(i));
375         auto jnode = m_Grid->interpolate(elem->GetNode(j));
376         if (inode == -1 || jnode == -1)
377             continue;
378         auto M = [&](const Mesh::Point& p)
379         {
380             double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p,
0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
381             double h = elem->GetMeasure();
382             //h = fabs(points[1].x - points[0].x);
383             h = sqrt(h);
384             //h = fabs(m_Grid->GetNode(0).x - m_Grid->GetNode(1).x);
385             //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
386             double Pe = minPec;
387             double tau = 0.;
388             double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. /
Pe);
389             //double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
390
391             if (Pe >= 1)
392                 tau = h / 2. / vel;
393             else
394                 tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
395             //tau = 0.;
396             //tau = 1e-7;
397             //std::cout << "tau Pe:\t" << Pe << std::endl;
398             //std::cout << "tau vel:\t" << vel << std::endl;
399             //std::cout << "tau h:\t" << h << std::endl;
400             //std::cout << "tau:\t" << tau << std::endl;
401             auto supg = tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p,
0) * elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p) * elem->GetShapeFunction(i, p);
402             return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p); // + supg;
403         };
404         mij = elem->Integrate(M, points);
405         m_RightMatrix->AddElement(inode, jnode, mij);
406     }
407 }
408 return 0;
409 }
410
411 template<class Problem, class Grid, class Matrix>
412 const int FEMethodZero<Problem, Grid, Matrix>::AssembleSUPGMatrix(const int l)
413 {
414     int i, j, k, nodes;
415     double mij;
416     const auto& elem{ m_Grid->GetElement(l) };
417     const int dofs{ (int)elem->GetDoFs() };
418     const int terms{ (int)m_problem->getNumberOfTerms() };
419     nodes = elem->GetNumberOfNodes();
420     std::vector<Mesh::Point> points(nodes);
421     for (i = 0; i < nodes; ++i)
422         points[i] = m_Grid->GetNode(elem->GetNode(i));
423     double minPec = -1;
424     auto MM = [&](const Mesh::Point& p)
425     {
426         double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
427         double h = elem->GetMeasure();
428         //h = fabs(points[1].x - points[0].x);
429         h = sqrt(h);
430         //h = fabs(m_Grid->GetNode(0).x - m_Grid->GetNode(1).x);
431         double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(),
p);
432

```

```

433         //double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(),
p);
434
435         if (Pe > minPec)
436             minPec = Pe;
437         return 0.;
438     };
439     elem->Integrate(MM, points);
440     for (i = 0; i < (int)dofs; ++i)
441     {
442         for (j = 0; j < (int)dofs; ++j)
443         {
444             auto inode = m_Grid->interpolate(elem->GetNode(i));
445             auto jnode = m_Grid->interpolate(elem->GetNode(j));
446             if (inode == -1 || jnode == -1)
447                 continue;
448             auto M = [&](const Mesh::Point& p)
449             {
450                 double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p,
0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
451                 double h = elem->GetMeasure();
452                 h = sqrt(h);
453                 //h = fabs(points[1].x - points[0].x);
454                 //h = fabs(m_Grid->GetNode(0).x - m_Grid->GetNode(1).x);
455                 //h *= h;
456                 //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
457                 double tau = 0.;
458                 //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
459                 double Pe = minPec;
460                 double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. /
Pe);
461                 //double beta = h / std::sqrt(3.) * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) -
1. / Pe);
462                 //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. / Pe);
463                 //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1. / Pe);
464                 //beta = 0.;
465                 //for (int ii = 0; ii < (int)dofs; ++ii)
466                 //beta += m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(ii, p);
467                 //return beta * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0)
* m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
468                 // elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
469                 if (Pe >= 1)
470                     tau = h / 2. / vel;
471                 else
472                     tau = h * h / 12. / m_problem->get_parameter(Terms::IDUDV, 1,
elem->GetType(), p);
473                 //tau = 0;
474                 //return 0.;
475                 //tau = 1e-7;
476                 //std::cout << "Stau Pe:\t" << Pe << std::endl;
477                 //std::cout << "Stau vel:\t" << vel << std::endl;
478                 //std::cout << "Stau h:\t" << h << std::endl;
479                 //std::cout << "Stau:\t" << tau << std::endl;
480                 auto ret = tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0)
* m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
481                 elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
482                 //std::cout << ret << std::endl;
483                 return ret;
484             };
485
486             //double tau =
487             auto _mij = elem->Integrate(M, points);
488             m_GlobalMatrix->AddElement(inode, jnode, _mij);
489         }
490     }
491     return 0;
492 }
493
494 template<class Problem, class Grid, class Matrix>
495 const int FEMethodZero<Problem, Grid, Matrix>::AssembleLocalMatrix(const int l, const int old)
496 {
497     int i, j, k, nodes;
498     double mij;
499     const auto& elem{ m_Grid->GetElement(l) };
500     const int dofs{ (int)elem->GetDoFs() };
501     const int terms{ (int)m_problem->getNumberOfTerms() };
502     nodes = elem->GetNumberOfNodes();
503     std::vector<Mesh::Point> points(nodes);
504     for (i = 0; i < nodes; ++i)
505         points[i] = m_Grid->GetNode(elem->GetNode(i));
506     for (k = 0; k < terms; ++k)
507     {
508         switch (m_problem->getTerm(k))
509         {

```

```

510         case Terms::IUV:
511             for (i = 0; i < (int)dofs; ++i)
512             {
513                 for (j = 0; j < (int)dofs; ++j)
514                 {
515                     auto M = [&](const Mesh::Point& p)
516                     {
517                         return m_problem->get_parameter(Terms::IUV, 1, elem->GetType(), p) *
elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
518                     };
519                     mij = elem->Integrate(M, points);
520                     auto inode = m_Grid->interpolate(elem->GetNode(i));
521                     auto jnode = m_Grid->interpolate(elem->GetNode(j));
522                     if (inode > -1 && jnode > -1)
523                         m_GlobalMatrix->AddElement(inode, jnode, mij);
524                 }
525             }
526             break;
527         case Terms::IDUDV:
528             for (i = 0; i < (int)dofs; ++i)
529             {
530                 for (j = 0; j < (int)dofs; ++j)
531                 {
532                     auto inode = m_Grid->interpolate(elem->GetNode(i));
533                     auto jnode = m_Grid->interpolate(elem->GetNode(j));
534                     if (inode == -1 || jnode == -1)
535                         continue;
536                     auto M = [&](const Mesh::Point& p)
537                     {
538                         //auto m = elem->GetGradShapeFunction(i, p) *
elem->GetGradShapeFunction(j, p);
539                         return m_problem->get_parameter(Terms::IDUDV, 1, elem->GetType(), p) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
540                     };
541                     //mij = m_Grid->getParameter(Parameters::DIFFUSION, 1, j) *
elem->Integrate(M, points);
542                     mij = elem->Integrate(M, points);
543                     m_GlobalMatrix->AddElement(inode, jnode, mij);
544                 }
545             }
546             break;
547         case Terms::IDUV:
548             for (i = 0; i < (int)dofs; ++i)
549             {
550                 for (j = 0; j < (int)dofs; ++j)
551                 {
552                     auto inode = m_Grid->interpolate(elem->GetNode(i));
553                     auto jnode = m_Grid->interpolate(elem->GetNode(j));
554                     if (inode == -1 || jnode == -1)
555                         continue;
556                     auto M = [&](const Mesh::Point& p)
557                     {
558                         return m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
559                     };
560                     auto _mij = elem->Integrate(M, points);
561                     m_GlobalMatrix->AddElement(inode, jnode, _mij);
562                 }
563             }
564             break;
565         case Terms::IUDV:
566             for (i = 0; i < dofs; ++i)
567             {
568                 for (j = 0; j < dofs; ++j)
569                 {
570                     auto inode = m_Grid->interpolate(elem->GetNode(i));
571                     auto jnode = m_Grid->interpolate(elem->GetNode(j));
572                     if (inode == -1 || jnode == -1)
573                         continue;
574                     auto M = [&](const Mesh::Point& p)
575                     {
576                         return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
577                     };
578                     //mij = m_CoarseGrid->getParameter(Parameters::ADVECTION, 1, j) *
m_flux(m_CoarseGrid->getSolution(1, j)) * elem->Integrate(M, points).x;
579                     mij = elem->Integrate(M, points).x;
580                     m_GlobalMatrix->AddElement(inode, jnode, mij);
581                 }
582             }
583             break;
584         case Terms::EUV:
585             for (i = 0; i < dofs; ++i)
586             {
587                 for (j = 0; j < dofs; ++j)
588                 {
589                     auto M = [&](const Mesh::Point& p)
590                     {

```

```

591         return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
592     };
593     mij = elem->Integrate(M, points);
594     m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::MASS, l, j) * m_Grid->getSolution(l, j) * mij;
595     //m_rhsvector->operator[] (m_nums[l] + i) +=
m_CoarseGrid->getParameter(Parameters::MASS, l, points[j]) * elem->GetValue(j) * mij;
596     }
597 }
598 break;
599 case Terms::EDUDV:
600     for (i = 0; i < dofs; ++i)
601     {
602         for (j = 0; j < dofs; ++j)
603         {
604             auto M = [&](const Mesh::Point& p)
605             {
606                 return elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
607             };
608             mij = elem->Integrate(M, points);
609             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::DIFFUSION, l, j) * m_Grid->getSolution(l, j) * mij;
610         }
611     }
612     break;
613 case Terms::EDUV:
614     for (i = 0; i < dofs; ++i)
615     {
616         for (j = 0; j < dofs; ++j)
617         {
618             auto M = [&](const Mesh::Point& p)
619             {
620                 return elem->GetShapeFunction(i, p) * elem->GetGradShapeFunction(j, p);
621             };
622             mij = elem->Integrate(M, points).x;
623             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, l, j) * mij;
624         }
625     }
626     break;
627 case Terms::EUDV:
628     for (i = 0; i < dofs; ++i)
629     {
630         for (j = 0; j < dofs; ++j)
631         {
632             auto M = [&](const Mesh::Point& p)
633             {
634                 return elem->GetGradShapeFunction(i, p) * elem->GetShapeFunction(j, p);
635             };
636             mij = elem->Integrate(M, points).x;
637             m_rhsvector->operator[] (elem->GetNode(i)) +=
m_Grid->getParameter(Parameters::ADVECTION, l, j) * mij; // *mij;
638         }
639     }
640     break;
641 case Terms::EFV:
642     for (i = 0; i < dofs; ++i)
643     {
644         /*for (j = 0; j < dofs; ++j)
645         {
646             auto M = [&](const Mesh::Point& p)
647             {
648                 return m_problem->get_parameter(Terms::EFV, elem->GetType(), l, j, p) *
elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
649             };
650             mij = elem->Integrate(M, points);
651             m_rhsvector->operator[] (elem->GetNode(i)) += mij;
652             */
653             auto M = [&](const Mesh::Point& p)
654             {
655                 return m_problem->get_parameter(Terms::EFV, elem->GetType(), l, i, p) *
elem->GetShapeFunction(i, p);
656             };
657             mij = elem->Integrate(M, points);
658             m_rhsvector->operator[] (elem->GetNode(i)) += mij;
659         }
660     }
661     break;
662 case Terms::RUV:
663     for (i = 0; i < (int)dofs; ++i)
664     {
665         for (j = 0; j < (int)dofs; ++j)
666         {
667             auto M = [&](const Mesh::Point& p)
668             {
669                 return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);

```

```

670         mij = elem->Integrate(M, points);
671         auto inode = m_Grid->interpolate(elem->GetNode(i));
672         auto jnode = m_Grid->interpolate(elem->GetNode(j));
673         if (inode > -1 && jnode > -1)
674             m_RightMatrix->AddElement(inode, jnode, mij);
675     }
676 }
677 break;
678 case Terms::SUPG:
679     for (i = 0; i < (int)dofs; ++i)
680     {
681         for (j = 0; j < (int)dofs; ++j)
682         {
683             auto inode = m_Grid->interpolate(elem->GetNode(i));
684             auto jnode = m_Grid->interpolate(elem->GetNode(j));
685             if (inode == -1 || jnode == -1)
686                 continue;
687             auto M = [&](const Mesh::Point& p)
688             {
689                 double vel = sqrt(m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0));
690                 double h = elem->GetMeasure();
691                 //double Pe = vel * h / 6. / m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p);
692                 double tau = 0.;
693                 double Pe = vel * h / 2. / m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p);
694                 //double beta = h / 2. / vel * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) -
1.) - 1. / Pe);
695                 double beta = h / std::sqrt(3.) * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) -
1.) - 1. / Pe);
696                 //double beta = h / 2. * ((exp(2. * Pe) + 1.) / (exp(2. * Pe) - 1.) - 1.
/ Pe);
697                 //beta = 0.;
698                 //for (int ii = 0; ii < (int)dofs; ++ii)
699                     //beta += m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * elem->GetGradShapeFunction(ii, p);
700                 return beta * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
701                 if (Pe >= 1)
702                     tau = h / 2. / vel;
703                 else
704                     tau = h * h / 12. / m_problem->get_parameter(Terms::IDUV, 1,
elem->GetType(), p);
705                 //return 0.;
706                 return tau * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(),
p, 0) * m_problem->get_parameter(Terms::IDUV, 1, elem->GetType(), p, 0) *
elem->GetGradShapeFunction(i, p) * elem->GetGradShapeFunction(j,
p);
707             };
708         };
709     };
710     //double tau =
711     auto _mij = elem->Integrate(M, points);
712     m_GlobalMatrix->AddElement(inode, jnode, _mij);
713 }
714 }
715 break;
716 default:
717     break;
718 }
719 }
720 }
721 return 0;
722 }
723 template<class Problem, class Grid, class Matrix>
724 void FEMethodZero<Problem, Grid, Matrix>::MainConditions()
725 {
726     double mu{ 1e10 };
727     const auto n = m_problem->get_number_of_boundaries();
728     const auto m = m_Grid->GetNumberOfBoundaries();
729     for (int i = 0; i < n; ++i)
730     {
731         const auto& type = m_problem->get_boundary_type(i);
732         for (int j = 0; j < m; ++j)
733         {
734             const auto& row = m_Grid->GetBoundary(j);
735             if (row->GetType() == type)
736             {
737                 const int dofs = (int)row->GetDoFs();
738                 const int dofs2 = 2;
739                 const auto& elem_num = row->GetNeighbour(0);
740                 const auto& elem = m_Grid->GetElement(elem_num);
741                 const int dofs_elem = elem->GetDoFs();
742                 std::vector<Mesh::Point> points(dofs_elem);
743                 for (int k = 0; k < dofs_elem; ++k)
744                     points[k] = m_Grid->GetNode(elem->GetNode(k));

```

```

745         for (int k = 0; k < dofs; ++k)
746         {
747             int l = 0;
748             for (; l < dofs_elem; ++l)
749             {
750                 if (elem->GetNode(l) == row->GetNode(k))
751                     break;
752             }
753             m_GlobalMatrix->NullRow(row->GetNode(k));
754             //m_GlobalMatrix->operator() (row->GetNode(k), row->GetNode(k)) *= mu;
755             //m_rhsvector->operator[] (row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, m_Grid->GetNode(row->GetNode(k)));
756             //m_rhsvector->operator[] (row->GetNode(k)) =
m_problem->get_boundary_parameter(0, type, elem_num, l, m_Grid->GetNode(row->GetNode(k)));
757             m_rhsvector->operator[] (row->GetNode(k)) = elem->GetWeight(l, points,
[=](const Mesh::Point& p) { return m_problem->get_boundary_parameter(0, type, p); });
758             if(m_problem->findTerm(Terms::RUV))
759             {
760                 m_RightMatrix->NullRow(row->GetNode(k));
761                 //m_RightMatrix->operator() (row->GetNode(k), row->GetNode(k)) *= mu;
762             }
763         }
764         /*for (int k = dofs2; k < dofs; ++k)
765         {
766             m_GlobalMatrix->NullRow(row->GetNode(k));
767             m_rhsvector->operator[] (row->GetNode(k)) = 0;
768         }*/
769     }
770 }
771 }
772 /*for (auto bnd : m_Grid->GetBoundaryConditions())
773 {
774     if (get<0>(bnd.second) == 1)
775         for (auto row : m_Grid->GetBoundary())
776         {
777             if (bnd.first == row->GetType())
778             {
779                 for (int i = 0; i < row->GetDoF(); ++i)
780                 {
781                     m_GlobalMatrix->NullRow(row->GetNodes(i));
782                     m_rhsvector[row->GetNodes(i)] =
get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
783                 }
784             }
785         }
786     }*/
787 }
788 template<class Problem, class Grid, class Matrix>
789 void FEMethodZero<Problem, Grid, Matrix>::SecondConditions()
790 {
791     double theta = 0;
792     int nfem;
793     Mesh::Point temp[3];
794     std::vector<int> local;
795     for (auto bnd : m_Grid->GetBoundaryConditions())
796     {
797         //if (get<0>(bnd.second) == 2)
798         {
799             for (auto row : m_Grid->GetBoundary())
800             {
801                 if (bnd.first == row->GetType())
802                 {
803                     local.resize(0);
804                     int dofs = row->GetDoF();
805                     nfem = row->GetNumberOfElement(0);
806                     auto elem = m_Grid->GetElements()[nfem];
807                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
808                     for (int j = 0; j < dofs; ++j)
809                     {
810                         temp[j] = m_Grid->GetNodes() [row->GetNodes(j)];
811                         for (int i = 0; i < elem->GetDoF(); ++i)
812                         {
813                             if (row->GetNodes(j) == elem->GetNodes() [i])
814                             {
815                                 local.push_back(i);
816                                 break;
817                             }
818                         }
819                     }
820                     for (int i = 0; i < dofs; ++i)
821                     {
822                         for (int j = 0; j < dofs; ++j)
823                         {
824                             //theta = get<1>(bnd.second) (m_Grid->GetNodes() [row->GetNodes(i)]);
825                             theta = 0;
826                             auto GetMass = [&](const Mesh::Point& p) {return
elem->GetBasis(local[j], p) * elem->GetBasis(local[i], p); };

```

```

827         auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
828         //if (i < 2 || j < 2)
829         m_rhsvector[row->GetNodes(i)] += theta * row->Integrate(GetMass,
temp);
830
831         //if (i < 3 || j < 3)
832         // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
833     }
834 }
835 }
836 }
837 }
838 }
839 }
840 template<class Problem, class Grid, class Matrix>
841 void FEMethodZero<Problem, Grid, Matrix>::StefanConditions()
842 {
843     double dest{ 0. }, lat{ 0 };
844     int nfem;
845     Mesh::Point temp[3];
846     std::vector<int> local;
847     for (auto bnd : m_Grid->GetBoundaryConditions())
848     {
849         //if (get<0>(bnd.second) == 4)
850         {
851             lat = 0;
852             //lat = get<2>(bnd.second);
853             for (auto row : m_Grid->GetBoundary())
854             {
855                 if (bnd.first == row->GetType())
856                 {
857                     local.resize(0);
858                     int dofs = row->GetDoF();
859                     nfem = row->GetNumberOfElement(0);
860                     auto elem = m_Grid->GetElements()[nfem];
861                     //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
862                     for (int j = 0; j < dofs; ++j)
863                     {
864                         temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
865                         for (int i = 0; i < elem->GetDoF(); ++i)
866                         {
867                             if (row->GetNodes(j) == elem->GetNodes()[i])
868                             {
869                                 local.push_back(i);
870                                 break;
871                             }
872                         }
873                     }
874                     for (int i = 0; i < dofs; ++i)
875                     {
876                         for (int j = 0; j < dofs; ++j)
877                         {
878                             dest = 0;
879                             //dest = get<1>(bnd.second) (m_Grid->GetNodes()[row->GetNodes(i)]);
880                             auto GetBBasis = [&](const Mesh::Point& p) {return row->GetBasis(j,
p)*row->GetBasis(i, p); };
881                             //if (i < 2 || j < 2)
882                             m_rhsvector[row->GetNodes(i)] += dest * lat *
row->Integrate(GetBBasis, temp);
883
884                             //if (i < 3 || j < 3)
885                             // m_rhsvector[row[i + 1]] += theta * row->Integrate(GetBBasis,
temp);
886                         }
887                     }
888                 }
889             }
890         }
891     }
892 }
893 template<class Problem, class Grid, class Matrix>
894 void FEMethodZero<Problem, Grid, Matrix>::ThirdConditions()
895 {
896     double param{ 0 }, beta{ 0 };
897     int nfem;
898     Mesh::Point temp[6];
899     std::vector<int> local;
900     auto fxy = [&](const Mesh::Point& p) {return (10 * p.y*m_time + m_time) / 10; };
901     //auto fxy = [&](const Point& p){return 10 * p.y + 10 * m_time; };
902     for (auto bnd : m_Grid->GetBoundaryConditions())
903     {
904         //if (get<0>(bnd.second) == 3)
905         {
906
907             for (auto row : m_Grid->GetBoundary())

```

```

908         {
909             if (bnd.first == row->GetType())
910             {
911                 local.resize(0);
912                 int dofs = row->GetDoF();
913                 nfem = row->GetNumberOfElement(0);
914                 auto elem = m_Grid->GetElements()[nfem];
915                 //auto GetBasis = [&](int t, Point p){return elem->GetBasis(t, p); };
916                 auto order = elem->GetDoF();
917                 for (int j = 0; j < dofs; ++j)
918                 {
919                     temp[j] = m_Grid->GetNodes()[row->GetNodes(j)];
920                     for (int i = 0; i < order; ++i)
921                     {
922                         if (row->GetNodes(j) == elem->GetNodes()[i])
923                         {
924                             local.push_back(i);
925                             break;
926                         }
927                     }
928                 }
929                 double val{ 0 };
930                 for (int i = 0; i < dofs; ++i)
931                 {
932                     for (int j = 0; j < dofs; ++j)
933                     {
934                         param = 0;
935                         beta = 0;
936                         //beta = get<2>(bnd.second);
937                         //param = get<1>(bnd.second)(m_Grid->GetNodes()[row->GetNodes(i)]);
938                         //param = fxy(temp[j]);
939                         auto GetBBasis = [&](const Mesh::Point& p) {return
elem->GetBasis(local[j], p)*elem->GetBasis(local[i], p); };
940                         //val = row->GetElement(GetBBasis, temp);
941                         val = row->Integrate(GetBBasis, temp);
942                         m_GlobalMatrix->operator()(row->GetNodes(i), row->GetNodes(j)) +=
beta * val;
943                         m_rhsvector[row->GetNodes(i)] += beta * param * val;
944                     }
945                 }
946             }
947         }
948     }
949 }
950
951 template<class Problem, class Grid, class Matrix>
952 Matrix* FEMethodZero<Problem, Grid, Matrix>::GetGlobalMatrix() const
953 {
954     return m_GlobalMatrix;
955 }
956 template<class Problem, class Grid, class Matrix>
957 const double FEMethodZero<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p) const
958 {
959     if (!m_solution.size())
960         return -1;
961     double val = 0;
962     int nfem = -1;
963     nfem = m_Grid->FindElement(p);
964     if (nfem == -1)
965         return -1;
966     auto elem = m_Grid->GetElements()[nfem];
967     for (int i = 0; i < elem->GetDoF(); ++i)
968         val += m_solution[elem->GetNodes()[i]] * elem->GetBasis(i, p);
969     return val;
970 }
971 template<class Problem, class Grid, class Matrix>
972 const double FEMethodZero<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
std::vector<double>& vec) const
973 {
974     if (!vec.size())
975         return -1;
976     double val{ 0 };
977     int nfem{ -1 };
978     nfem = m_Grid->FindElement(p);
979     if (nfem == -1)
980         return -1;
981     auto elem = m_Grid->GetElements()[nfem];
982     for (int i = 0; i < elem->GetDoFs(); ++i)
983         val += vec[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
984     return val;
985 }
986 template<class Problem, class Grid, class Matrix>
987 const double FEMethodZero<Problem, Grid, Matrix>::GetValue(const Mesh::Point& p, const
std::vector<double>& vec, const int num) const
988 {
989     if (!vec.size() || num < 0)
990         return -1;

```



```

991         double val{ 0 };
992         auto elem = m_Grid->GetElements()[num];
993         for (int i = 0; i < elem->GetDoF(); ++i)
994             val += vec[elem->GetNodes()[i]] * elem->GetBasis(i, p);
995         return val;
996     }
997     //template<class Problem, class Grid, class Matrix>
998     //const Mesh::Point FEMethodZero<Problem, Grid, Matrix>::GetGradValue(const Mesh::Point& p,
const std::vector<double>& vec) const
999     //{
1000     //    Mesh::Point val{ 0, 0 };
1001     //    int nfem{ -1 };
1002     //    nfem = m_Grid->FindElement(p);
1003     //    if (nfem == -1)
1004     //        return val;
1005     //    auto elem = m_Grid->GetElements()[nfem];
1006     //    for (int i = 0; i < elem->GetDoF(); ++i)
1007     //    {
1008     //        val.x += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).x;
1009     //        val.y += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).y;
1010     //        val.z += vec[elem->GetNodes()[i]] * elem->GetGradBasis(i, p).z;
1011     //    }
1012     //    return val;
1013     //}
1014     template<class Problem, class Grid, class Matrix>
1015     const double FEMethodZero<Problem, Grid, Matrix>::GetEffective(const std::vector<double>& vec)
const
1016     {
1017         double sum = 0;
1018         //std::vector<int> dofs;
1019         //Mesh::Point points[10];
1020         //for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1021         //{
1022             //auto mb = [&](const Mesh::Point& b) {return GetGradValue(b, vec)*GetGradValue(b,
vec); };
1023             //dofs.resize(0);
1024             //auto elem = m_Grid->GetElements()[i];
1025             //int order = elem->GetDoF();
1026             //double diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
1027             //for (int j = 0; j < order; ++j)
1028             //{
1029                 //dofs.push_back(elem->GetNodes()[j]);
1030                 //points[j] = m_Grid->GetNodes()[dofs[j]];
1031             //}
1032             //sum += diff * elem->Integrate(mb, points);
1033             //}
1034             //std::cout << "Effect (local): " << sum << std::endl;
1035             //std::cout << "Effect (local) sqrt: " << sqrt(sum) << std::endl;
1036             return sum;
1037         }
1038     //template<class Problem, class Grid, class Matrix>
1039     //const Mesh::Point FEMethodZero<Problem, Grid, Matrix>::GetLambdaGrad(const Mesh::Point& p,
const std::vector<double>& vec) const
1040     //{
1041     //    Mesh::Point val{ 0, 0, 0 };
1042     //    //double val{ 0 };
1043     //    double diff{ 0 };
1044     //    Mesh::Point temp{ 0, 0, 0 };
1045     //    int nfem{ -1 };
1046     //    nfem = m_Grid->FindElement(p);
1047     //    if (nfem == -1)
1048     //        return val;
1049     //    auto elem = m_Grid->GetElements()[nfem];
1050     //    diff = std::get<0>(m_Grid->GetDiffusion().find(elem->GetType())->second);
1051     //    for (int i = 0; i < elem->GetDoF(); ++i)
1052     //    {
1053         //        //val += elem->GetGradBasis(i, p) * elem->GetGradBasis(i, p) * vec[elem->GetNodes()[i]]
* vec[elem->GetNodes()[i]] * diff;
1054         //        //val += elem->GetBasis(i, p) * vec[elem->GetNodes()[i]] * diff;
1055         //        temp = elem->GetGradBasis(i, p);
1056         //        val.x += temp.x * vec[elem->GetNodes()[i]] * (diff);
1057         //        val.y += temp.y * vec[elem->GetNodes()[i]] * (diff);
1058         //        val.z += temp.z * vec[elem->GetNodes()[i]] * (diff);
1059         //    }
1060         //    return val;
1061     //}
1062     template<class Problem, class Grid, class Matrix>
1063     const std::vector<double> FEMethodZero<Problem, Grid, Matrix>::GetRightVector() const
1064     {
1065         return *m_rhsvector;
1066     }
1067     template<class Problem, class Grid, class Matrix>
1068     void FEMethodZero<Problem, Grid, Matrix>::OutDatFormat(const Mesh::Point& mn, const
Mesh::Point& mx, const std::string& file_name, const std::vector<double>& vec) const
1069     {
1070         std::ofstream of(file_name + ".z.dat");
1071         std::streambuf *buf = std::cout.rdbuf();

```

```

1072         std::cout.rdbuf(of.rdbuf());
1073         std::cout << "TITLE = FE-METHOD\n";
1074         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1075         std::cout << "ZONE i=51, j=51, F=POINT\n";
1076         double stepx = (mx.x - mn.x) / 51;
1077         double stepy = (mx.y - mn.y) / 51;
1078         for (int i = 0; i < 51; ++i)
1079             for (int j = 0; j < 51; ++j)
1080                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.x + j * stepx, mn.y + i * stepy, mn.z), vec) << std::endl;
1081         std::cout.rdbuf(buf);
1082         of.close();
1083         of.open(file_name + "x.dat");
1084         buf = std::cout.rdbuf();
1085         std::cout.rdbuf(of.rdbuf());
1086         std::cout << "TITLE = FE-METHOD\n";
1087         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1088         std::cout << "ZONE i=51, j=51, F=POINT\n";
1089         for (int i = 0; i < 51; ++i)
1090             for (int j = 0; j < 51; ++j)
1091                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.z, mn.x + j * stepx, mn.y + i * stepy), vec) << std::endl;
1092         std::cout.rdbuf(buf);
1093         of.close();
1094         of.open(file_name + "y.dat");
1095         buf = std::cout.rdbuf();
1096         std::cout.rdbuf(of.rdbuf());
1097         std::cout << "TITLE = FE-METHOD\n";
1098         std::cout << "VARIABLES = \"dx1\", \"dx2\", \"u\"\n";
1099         std::cout << "ZONE i=51, j=51, F=POINT\n";
1100         for (int i = 0; i < 51; ++i)
1101             for (int j = 0; j < 51; ++j)
1102                 std::cout << mn.x + j * stepx << "\t" << mn.y + stepy * i << "\t" <<
GetValue(Mesh::Point(mn.x + j * stepx, mn.z, mn.y + i * stepy), vec) << std::endl;
1103         std::cout.rdbuf(buf);
1104         of.close();
1105     }
1106     template<class Problem, class Grid, class Matrix>
1107     void FEMethodZero<Problem, Grid, Matrix>::ApplySources()
1108     {
1109         int nfem = -1;
1110         auto total = m_problem->get_total_sources();
1111         for (int i = 0; i < total; ++i)
1112         {
1113             auto src = m_problem->get_point_source(i);
1114             auto point = src.get_point();
1115             nfem = m_Grid->FindElement(point);
1116             if (nfem != -1)
1117             {
1118                 auto val = src.get_value();
1119                 auto elem = m_Grid->GetElement(nfem);
1120                 for (int j = 0; j < 3; ++j)
1121                     m_rhsvector->operator[] (elem->GetNode(j)) += val * elem->GetShapeFunction(j,
point);
1122             }
1123             nfem = -1;
1124         }
1125         /*for (auto srd : m_Grid->GetDottedSources())
1126         {
1127             nfem = m_Grid->FindElement(srd.first);
1128             if (nfem != -1)
1129             {
1130                 auto elem = m_Grid->GetElements()[nfem];
1131                 for (int i = 0; i < elem->GetDoF(); ++i)
1132                 {
1133                     m_rhsvector[elem->GetNodes()[i]] += srd.second * elem->GetBasis(i, srd.first);
1134                 }
1135             }
1136             nfem = -1;
1137         }*/
1138     }
1139     template<class Problem, class Grid, class Matrix>
1140     void FEMethodZero<Problem, Grid, Matrix>::Rediscretization(const std::shared_ptr<Grid>& grid)
1141     {
1142         m_GlobalMatrix->NullMatrix();
1143         for (unsigned int i = 0; i < m_rhsvector->size(); ++i)
1144             (*m_rhsvector)[i] = 0;
1145         AssemblGlobal();
1146         //SecondConditions();
1147         //ApplySources();
1148         //StefanConditions();
1149         MainConditions();
1150     }
1151     template<class Problem, class Grid, class Matrix>
1152     void FEMethodZero<Problem, Grid, Matrix>::Rediscretization()
1153     {
1154         m_time += m_step;

```

```

1155         m_GlobalMatrix->NullMatrix();
1156         for (unsigned int i = 0; i < m_rhsvector->size(); ++i)
1157             (*m_rhsvector)[i] = 0;
1158         AssemblGlobal();
1159         SecondConditions();
1160         ThirdConditions();
1161         StefanConditions();
1162         //ApplySources();
1163         MainConditions();
1164     }
1165     template<class Problem, class Grid, class Matrix>
1166     void FEMethodZero<Problem, Grid, Matrix>::GetSolution(std::vector<double>& vec)
1167     {
1168         int size = vec.size();
1169         //Translation(vec);
1170         for (int i = 0; i < size; ++i)
1171             vec[i] = m_solution[i];
1172     }
1173     template<class Problem, class Grid, class Matrix>
1174     const double FEMethodZero<Problem, Grid, Matrix>::GetSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
1175     {
1176         double sum{ 0 };
1177         auto nfem{ g.FindElement(p) };
1178         if (nfem < 0)
1179             return 0.;
1180         auto elem{ g.GetElement(nfem) };
1181         auto dofs{ elem->GetDoFs() };
1182         for (auto i{ 0 }; i < dofs; ++i)
1183             sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1184         return sum;
1185     }
1186     template<class Problem, class Grid, class Matrix>
1187     const double FEMethodZero<Problem, Grid, Matrix>::GetSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p, const int nfem)
1188     {
1189         double sum{ 0 };
1190         //if (nfem < 0)
1191             // return 0.;
1192         auto elem{ g.GetElement(nfem) };
1193         auto dofs{ elem->GetDoFs() };
1194         //std::cout << nfem << std::endl;
1195         for (auto i{ 0 }; i < dofs; ++i)
1196             sum += weights[elem->GetNode(i)] * elem->GetShapeFunction(i, p);
1197         return sum;
1198     }
1199     template<class Problem, class Grid, class Matrix>
1200     const Mesh::Point FEMethodZero<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p)
1201     {
1202         Mesh::Point sum{ 0, 0, 0 };
1203         auto nfem{ g.FindElement(p) };
1204         auto elem{ g.GetElement(nfem) };
1205         auto dofs{ elem->GetDoFs() };
1206         for (auto i{ 0 }; i < dofs; ++i)
1207             sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
1208         return sum;
1209     }
1210     template<class Problem, class Grid, class Matrix>
1211     const Mesh::Point FEMethodZero<Problem, Grid, Matrix>::GetGradSolution(const Grid& g, const
std::vector<double> &weights, const Mesh::Point& p, const int nfem)
1212     {
1213         Mesh::Point sum{ 0, 0, 0 };
1214         auto elem{ g.GetElement(nfem) };
1215         auto dofs{ elem->GetDoFs() };
1216         for (auto i{ 0 }; i < dofs; ++i)
1217             sum += weights[elem->GetNode(i)] * elem->GetGradShapeFunction(i, p);
1218         return sum;
1219     }
1220     template<class Problem, class Grid, class Matrix>
1221     void FEMethodZero<Problem, Grid, Matrix>::LoadSolution(const std::vector<double>& vec)
1222     {
1223         m_solution.resize(vec.size());
1224         for (unsigned int i = 0; i < vec.size(); ++i)
1225             m_solution[i] = vec[i];
1226     }
1227     template<class Problem, class Grid, class Matrix>
1228     void FEMethodZero<Problem, Grid, Matrix>::OutMeshFormat(const std::string& file_name, const
std::vector<double>& vec)
1229     {
1230         const int size{ (int)m_Grid->GetNodes().size() };
1231         const int number{ (int)m_Grid->GetElements().size() };
1232         //const int size{ number * 4 };
1233         std::ofstream ofs(file_name + ".dat", std::ios::out);
1234         std::string title("TITLE = \"Mesh data\\n Variables = \"X\", \"Y\", \"Z\", \"U\"\\n Zone N
= \" + std::to_string(size) + \", E = \" + std::to_string(number) + \", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\\n");

```

```

1235         ofs << title;
1236         Mesh::Point p;
1237         for (int i = 0; i < size; ++i)
1238         {
1239             p = m_Grid->GetNodes()[i];
1240             ofs << p.x << "\t" << p.y << "\t" << p.z << "\t" << GetValue(p, vec, 1) << std::endl;
1241         }
1242         for (int i = 0; i < number; ++i)
1243         {
1244             auto elem = m_Grid->GetElements()[i];
1245             for (int k = 0; k < 4; ++k)
1246             {
1247                 ofs << elem->GetNodes()[k] + 1 << "\t";
1248             }
1249             ofs << std::endl;
1250         }
1251         ofs.close();
1252     }
1253     template<class Problem, class Grid, class Matrix>
1254     void FEMethodZero<Problem, Grid, Matrix>::OutMeshTimeFormat(const std::string& file_name, const
std::vector<double>& vec)
1255     {
1256         const int size{ (int)m_Grid->GetNodes().size() };
1257         const int number{ (int)m_Grid->GetElements().size() };
1258         //const int size{ number * 4 };
1259         std::ofstream ofs(file_name + ".dat", std::ios::out | std::ios::app);
1260         std::string title("TITLE = \"Mesh data\"\n Variables = \"X\", \"Y\", \"Z\", \"U\"\n Zone N
= \" + std::to_string(size) + \", E = \" + std::to_string(number) + \", DATAPACKING = POINT, ZONETYPE =
FETETRAHEDRON\n");
1261         ofs << title;
1262         Mesh::Point p;
1263         for (int i = 0; i < size; ++i)
1264         {
1265             p = m_Grid->GetNodes()[i];
1266             ofs << p.x << "\t" << p.y << "\t" << p.z << "\t" << GetValue(p, vec, 1) << std::endl;
1267         }
1268         for (int i = 0; i < number; ++i)
1269         {
1270             auto elem = m_Grid->GetElements()[i];
1271             for (int k = 0; k < 4; ++k)
1272             {
1273                 ofs << elem->GetNodes()[k] + 1 << "\t";
1274             }
1275             ofs << std::endl;
1276         }
1277         ofs.close();
1278     }
1279     template<class Problem, class Grid, class Matrix>
1280     void FEMethodZero<Problem, Grid, Matrix>::ProjectSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&, const int)> GetVal,
std::vector<double>& vec)
1281     {
1282         for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1283         {
1284             auto elem = m_Grid->GetElements()[i];
1285             int order = elem->GetDoF();
1286             for (int j = 0; j < order; ++j)
1287                 sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes()[j]], vec, i);
1288         }
1289     }
1290     template<class Problem, class Grid, class Matrix>
1291     void FEMethodZero<Problem, Grid, Matrix>::ProjectSolution(std::vector<double>& sol,
std::function<const double(const Mesh::Point&, const std::vector<double>&)> GetVal,
std::vector<double>& vec, const int)
1292     {
1293         for (int i = 0; i < m_Grid->GetElements().size(); ++i)
1294         {
1295             auto elem = m_Grid->GetElements()[i];
1296             int order = elem->GetDoF();
1297             for (int j = 0; j < order; ++j)
1298                 sol[elem->GetNodes()[j]] = GetVal(m_Grid->GetNodes()[elem->GetNodes()[j]], vec);
1299         }
1300     }
1301     template<class Problem, class Grid, class Matrix>
1302     const std::vector<double> FEMethodZero<Problem, Grid, Matrix>::SetSolution(const int sol, const
int liq, const double s, const double l, const double m)
1303     {
1304         int i;
1305         m_solution.resize(m_Grid->GetNodes().size());
1306         for (i = 0; i < m_Grid->GetElements().size(); ++i)
1307         {
1308             auto elem = m_Grid->GetElements()[i];
1309             int order = elem->GetDoF();
1310             if (m_Grid->GetElements()[i]->GetType() == liq)
1311                 for (int j = 0; j < order; ++j)
1312                     m_solution[elem->GetNodes()[j]] = 1;
1313             else

```

```

1314         for (int j = 0; j < order; ++j)
1315             m_solution[elem->GetNodes()[j]] = s;
1316     }
1317
1318     for (auto bnd : m_Grid->GetBoundaryConditions())
1319     {
1320         //if (get<0>(bnd.second) == 4)
1321         {
1322             for (auto row : m_Grid->GetBoundary())
1323             {
1324                 if (bnd.first == row->GetType())
1325                 {
1326                     int dofs = row->GetDoF();
1327                     for (int i = 0; i < dofs; ++i)
1328                     {
1329                         m_solution[row->GetNodes(i)] = m;
1330                     }
1331                 }
1332             }
1333         }
1334     }
1335     return m_solution;
1336 }
1337 template<class Problem, class Grid, class Matrix>
1338 FVMethodZero<Problem, Grid, Matrix>::~FVMethodZero()
1339 {
1340     delete m_Grid;
1341 }
1342 }
1343 }
1344
1345 #endif // !CORENC_METHODS_FVMethodZeroZero_h
1346

```

7.83 CoreNCFEM/Methods/FVMethod.cpp File Reference

```
#include "FVMethod.h"
```

7.84 CoreNCFEM/Methods/FVMethod.h File Reference

```
#include "../Grids/Mesh1D.h"
```

Classes

- class [corenc::method::FVMethod1d](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::method](#)

Enumerations

- enum class [corenc::method::FVFlux](#) { [corenc::method::LaxFriedrichs](#) , [corenc::method::Upwind](#) , [corenc::method::Central](#) , [corenc::method::NOFLUX](#) }

7.85 FVMMethod.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_METHODS_FINITEVOLUME_H_
2 #define CORENC_METHODS_FINITEVOLUME_H_
3
4 #include "../Grids/Mesh1D.h"
5
6 namespace corenc
7 {
8     namespace method
9     {
10         enum class FVFlux
11         {
12             LaxFriedrichs,
13             Upwind,
14             Central,
15             NOFLUX,
16         };
17         class FVMMethod1d
18         {
19         public:
20             FVMMethod1d();
21             ~FVMMethod1d();
22             static const int
23
24             flux_func,
25
26             static const double
27             Mesh::Point& p);
28         };
29     }
30 }
31 #endif // CORENC_METHODS_FINITEVOLUME_H_

```

```

Solve(Mesh::CMesh<CFESolution>* mesh,
const std::function<const double(const double)>&

const FVFlux& flux_type,
std::vector<double>& new_solution,
const double time_step);
GetSolution(const Mesh::CMesh1D& g, const

```

7.86 CoreNCFEM/Methods/RungeKutta.h File Reference

```

#include <memory>
#include "../Point.h"
#include <functional>

```

Classes

- class [corenc::method::RungeKutta< Problem, Type >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::method](#)

7.87 RungeKutta.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_METHODS_RUNGEKUTTA
2 #define CORENC_METHODS_RUNGEKUTTA
3
4 #include <memory>
5 #include "../Point.h"
6 #include <functional>
7 namespace corenc

```

```

8 {
9     namespace method
10     {
11         template<class Problem, class Type>
12         class RungeKutta
13         {
14         public:
15             RungeKutta() {};
16             RungeKutta(const double step, const double final, Problem* problem, const Type* solution) :
17                 m_step{ step },
18                 m_final{ final },
19                 m_problem{problem} {}
20             const Type discretize(const Type& solution, const std::function<const Type(const
double time, const double time_step, const Type& curr_sol, Type* result)>& func);
21             const Type explicitEuler(const Type& solution, const std::function<const
Type(const double time, const double time_step, const Type& curr_sol, Type* result)>& func);
22             void updateTimeStep(const double step) { m_step = step; };
23             ~RungeKutta() {};
24         private:
25             double m_step;
26             double m_final;
27             double m_curr;
28             Problem* m_problem;
29             Type* m_solution;
30             static const std::vector<double> vector_mult(const std::vector<double>& lhs, const double
rhs)
31             {
32                 std::vector<double> vc(lhs);
33                 for (auto &it : vc)
34                     it *= rhs;
35                 return vc;
36             }
37             static const std::vector<double> vector_mult(const double rhs, const std::vector<double>&
lhs)
38             {
39                 std::vector<double> vc(lhs);
40                 for (auto &it : vc)
41                     it *= rhs;
42                 return vc;
43             }
44             static const std::vector<double> vector_divide(const std::vector<double>& lhs, const double
rhs)
45             {
46                 std::vector<double> vc(lhs);
47                 for (auto &it : vc)
48                     it /= rhs;
49                 return vc;
50             }
51             static const std::vector<double> vector_divide(const double rhs, const std::vector<double>&
lhs)
52             {
53                 std::vector<double> vc(lhs);
54                 for (auto &it : vc)
55                     it /= rhs;
56                 return vc;
57             }
58             static const std::vector<double> vector_add(const std::vector<double>& rhs, const
std::vector<double>& lhs)
59             {
60                 std::vector<double> vc(lhs);
61                 for (unsigned i{ 0 }; i < vc.size(); ++i)
62                     vc[i] += rhs[i];
63                 return vc;
64             }
65             };
66
67         template<class Problem, class Type>
68         const Type RungeKutta<Problem, Type>::discretize(const Type& u_pr, const std::function<const
Type(const double time, const double time_step, const Type& curr_sol, Type* result)>& func)
69         {
70             Type k[4];
71             const int n{ int(m_final / m_step) };
72             func(m_curr, m_step, u_pr, &k[0]);
73             //std::vector<double> tempc(m_curr.size());
74             std::vector<double> tempu(u_pr.size());
75             std::vector<double> tempk(u_pr.size());
76             tempk = vector_divide(k[0], 2);
77             tempu = vector_add(u_pr, tempk);
78             func(m_curr + m_step / 2, m_step, tempu, &k[1]);
79             //func(m_curr + m_step / 2, m_step, u_pr + k[0] / 2, &k[1]);
80         }
81     }
82 }

```

```

87         tempk = vector_divide(k[1], 2);
88         tempu = vector_add(u_pr, tempk);
89         func(m_curr + m_step / 2, m_step, tempu, &k[2]);
90         //func(m_curr + m_step / 2, m_step, u_pr + k[1] / 2, &k[2]);
91
92         tempu = vector_add(u_pr, k[2]);
93         func(m_curr + m_step, m_step, tempu, &k[3]);
94         //func(m_curr + m_step, m_step, u_pr + k[2], &k[3]);
95
96         tempk = vector_mult(k[1], 2);
97         tempu = vector_mult(k[2], 2);
98         k[3] = vector_add(k[3], tempu);
99         k[3] = vector_add(k[3], tempk);
100        k[3] = vector_add(k[3], k[0]);
101        k[3] = vector_divide(k[3], 6.);
102        //k[3] = k[0] + 2 * k[1] + 2 * k[2] + k[3];
103        //k[3] = 1. / 6 * k[3];
104        m_problem->addTerm(Terms::EUV);
105        m_problem->addTerm(Terms::IUV);
106        m_curr += m_step;
107        return k[3];
108    }
109
110    template<class Problem, class Type>
111    const Type RungeKutta<Problem, Type>::explicitEuler(const Type& u_pr, const std::function<const
Type(const double time, const double time_step, const Type& curr_sol, Type* result)>& func)
112    {
113        Type k;
114        func(m_curr, m_step, u_pr, &k);
115        m_problem->addTerm(Terms::EUV);
116        m_problem->addTerm(Terms::IUV);
117        m_curr += m_step;
118        return k;
119    }
120    }
121 }
122 #endif // !CORENC_METHODS_RUNGEKUTTA

```

7.88 CoreNCFEM/Methods/system_dg_method.h File Reference

```

#include <functional>
#include <set>
#include "../Point.h"
#include <memory>
#include <cmath>
#include "FEMethod.h"
#include <map>
#include <algorithm>
#include <vector>
#include "dg_flux.h"

```

Classes

- class [corenc::method::system_dg_method< Problem, Grid, Matrix >](#)
- class [corenc::method::system_dg_method< Grid, bool, bool >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::method](#)

Macros

- #define [CORENC_METHODS_SYSTEM_DG_METHOD_H_](#)

7.88.1 Macro Definition Documentation

7.88.1.1 CORENC_METHODS_SYSTEM_DG_METHOD_H_

```
#define CORENC_METHODS_SYSTEM_DG_METHOD_H_
```

7.89 system_dg_method.h

[Go to the documentation of this file.](#)

```
1 // NO GENERALIZATION HERE
2 // JUST PLAIN DG FOR SYSTEM IN N - DIMENSIONAL SPACE FOR ONE TIME STEP
3 // CONSTANT BASIS FUNCTIONS
4
5 #pragma once
6 #ifndef CORENC_METHODS_SYSTEM_DG_METHOD_H_
7 #define CORENC_METHODS_SYSTEM_DG_METHOD_H_
8 #include <functional>
9 #include <set>
10 #include "../Point.h"
11 #include <memory>
12 #include <cmath>
13 #include "FEMethod.h"
14 #include <map>
15 #include <algorithm>
16 #include <vector>
17 #include "dg_flux.h"
18
19 namespace corenc
20 {
21     namespace method
22     {
23         template<class Problem, class Grid, class Matrix>
24         class system_dg_method
25         {
26         public:
27             system_dg_method() :
28                 m_problem{ nullptr },
29                 m_CoarseGrid{ nullptr },
30                 m_GlobalMatrix{ nullptr },
31                 m_rhsvector{ nullptr }
32             {};
33             system_dg_method(
34                 Problem* p,
35                 Grid* g,
36                 Matrix* m,
37                 //Solution* s,
38                 const size_t sys_size,
39                 std::vector<double>* rhs):
40                 //const std::function<const double(const double)>& flux_function,
41                 //const DGFlux flux_type) :
42                 m_problem{ p },
43                 m_CoarseGrid{ g },
44                 m_GlobalMatrix{ m },
45                 m_N{ g->GetNumberOfElements() },
46                 m_Ns{ g->GetNumberOfBoundaries() },
47                 m_rhsvector{ rhs },
48                 //m_flux(flux_function),
49                 m_sys_size{sys_size}{
50                 GeneratePortrait();
51             }
52             ~system_dg_method() {};
53             const int Assemble();
54             const int changeFlux(const DGFlux flux_type) { m_fluxtype = flux_type;
55
56             return 0; };
57             const Matrix* GetGlobalMatrix() const { return m_GlobalMatrix; };
58             const std::vector<double> GetSolution() const { return m_vec; };
59             const double GetSolution(const std::vector<double>& point) const;
60             const double GetMaxSolution() const;
61             const double GetMinSolution() const;
62             static const double GetSolution(const Grid& g, const std::vector<double> &dg_sol,
63                 const Mesh::Point& p)
64             {
65                 double sum{ 0 };
66             }
```

```

63         auto nfem{ g.FindElement(p) };
64         auto elem{ g.GetElement(nfem) };
65         auto dofs{ elem->GetDoFs() };
66         for (auto i{ 0 }; i < dofs; ++i)
67         {
68             sum += dg_sol[nfem * dofs + i] * elem->GetShapeFunction(i, p);
69         }
70         return sum;
71     }
72     const double          GetSolution(const std::vector<double> &dg_sol, const Mesh::Point& p)
73     {
74         double sum{ 0 };
75         auto nfem{ m_CoarseGrid->FindElement(p) };
76         auto elem{ m_CoarseGrid->GetElement(nfem) };
77         auto dofs{ elem->GetDoFs() };
78         for (auto i{ 0 }; i < dofs; ++i)
79         {
80             sum += dg_sol[nfem * dofs + i] * elem->GetShapeFunction(i, p);
81         }
82         return sum;
83     }
84     const int             toDGSolution(const Grid& g, std::vector<double>& dg_result) const
85     {
86         //dg_result->resize(m_rhsvector->size());
87         dg_result.resize(m_rhsvector->size());
88         for (unsigned i{ 0 }; i < g.GetNumberOfElements(); ++i)
89         {
90             auto elem{ g.GetElement(i) };
91             auto dofs{ elem->GetDoFs() };
92             for (unsigned j{ 0 }; j < dofs; ++j)
93                 //dg_result->operator[](m_nums[i] + j) = g.getSolution(i, j);
94                 dg_result[m_nums[i] + j] = g.getSolution(i, j);
95         }
96         return 0;
97     }
98     const int             updateWeights(const std::vector<double>& dg_result)
99     {
100         for (unsigned int i{ 0 }; i < (unsigned int)m_CoarseGrid->GetNumberOfElements(); ++i)
101         {
102             for (unsigned int j{ 0 }; j < (unsigned int)m_CoarseGrid->GetElement(i)->GetDoFs();
103             ++i)
104                 m_CoarseGrid->updateSolution(i, j, dg_result[m_nums[i] + j]);
105         }
106         return 0;
107     }
108     const int             DGtoStandart(const std::vector<double>& dg_result)
109     {
110         for (unsigned int i{ 0 }; i < (unsigned int)m_CoarseGrid->GetNumberOfElements(); ++i)
111         {
112             auto elem{ m_CoarseGrid->GetElement(i) };
113             auto dofs{ elem->GetDoFs() };
114             for (unsigned int j{ 0 }; j < (unsigned int)dofs; ++j)
115                 //m_CoarseGrid->updateSolution(i, j, dg_result[m_nums[i] + j]);
116                 m_CoarseGrid->updateSolution(i, j, dg_result[m_nums[i] + j]);
117         }
118         return 0;
119     }
120 private:
121     const int             GeneratePortrait();
122     void                  assembleBoundaries();
123     void                  assemble_flux(const unsigned boundary);
124     const double          numerical_flux(const double ul, const double ur, const double
fl, const double fr) const;
125     void                  MainConditions();
126     void                  SecondConditions();
127     void                  ThirdConditions();
128     const int             AssembleGlobal();
129     const int             AssembleFluxMatrix();
130     Grid*                 m_CoarseGrid;
131     Matrix*               m_GlobalMatrix;
132     std::vector<double>*   m_rhsvector;
133     std::vector<unsigned int> m_nums;
134     unsigned int           m_N; // number of elements
135     unsigned int           m_Ns; // number of boundaries
136     unsigned int           m_size;
137     Problem*              m_problem;
138     std::vector<double>     m_vec;
139     std::vector<double>     m_solution;
140     //std::function<const Mesh::Point(const Mesh::Point)> m_numflux;
141     //std::function<const Mesh::Point(const Mesh::Point)> m_flux;
142     DGFlux                m_fluxtype;
143     size_t                 m_sys_size;
144     std::function<const double(const double)> m_flux;
145     const int             AssembleLocalMatrix(const int);
146 };
147

```

```

148     template<class Grid>
149     class system_dg_method<Grid, bool, bool>
150     {
151     public:
152         static const double          GetSolution(const Grid& g, const std::vector<double> &dg_sol,
const Mesh::Point& p)
153         {
154             double sum{ 0 };
155             auto nfem{ g.FindElement(p) };
156             auto elem{ g.GetElement(nfem) };
157             auto dofs{ elem->GetDoFs() };
158             for (auto i{ 0 }; i < dofs; ++i)
159             {
160                 sum += dg_sol[nfem * dofs + i] * elem->GetShapeFunction(i, p);
161             }
162             return sum;
163         }
164     };
165
166     template<class Problem, class Grid, class Matrix>
167     const int system_dg_method<Problem, Grid, Matrix>::Assemble()
168     {
169         //GeneratePortrait();
170         AssembleGlobal();
171         AssembleFluxMatrix();
172         MainConditions();
173         SecondConditions();
174         ThirdConditions();
175         return 0;
176     }
177
178     template<class Problem, class Grid, class Matrix>
179     const int system_dg_method<Problem, Grid, Matrix>::GeneratePortrait()
180     {
181         int lorder, rorder, order;
182         std::vector<std::set<unsigned int>> temp;
183         unsigned int i, j, nk, ne, k, sz, size;
184         m_size = 0;
185         m_nums.resize(m_N * m_sys_size);
186
187         nk = 0;
188         sz = m_N * m_sys_size;
189         for (i = 0, k = 0; k < sz; ++i, k += m_sys_size)
190         {
191             size = m_CoarseGrid->GetElement(i)->GetDoFs() * m_sys_size;
192             for(j = 0; j < m_sys_size; ++j)
193                 m_nums[k + j] = m_size + j * m_CoarseGrid->GetElement(i)->GetDoFs();
194             m_size += size;
195         }
196         temp.resize(m_size);
197         sz = m_Ns;
198         for (k = 0; k < sz; k += m_sys_size)
199         {
200             auto bound = m_CoarseGrid->GetBoundary(k);
201             nk = bound->GetNeighbour(0);
202             ne = bound->GetNeighbour(1);
203             lorder = m_CoarseGrid->GetElement(nk)->GetDoFs();
204             if (ne != -1)
205             {
206                 rorder = m_CoarseGrid->GetElement(ne)->GetDoFs();
207                 for (i = 0; i < lorder; ++i)
208                     for (j = 0; j < rorder; ++j)
209                         temp[m_nums[nk] + j].insert(m_nums[nk] + i);
210                 for (i = 0; i < lorder; ++i)
211                     for (j = i + 1; j < lorder; ++j)
212                         temp[m_nums[nk] + j].insert(m_nums[nk] + i);
213             }
214             else
215             {
216                 for (i = 0; i < lorder; ++i)
217                     for (j = i + 1; j < lorder; ++j)
218                         temp[m_nums[nk] + j].insert(m_nums[nk] + i);
219             }
220         }
221
222         /*temp.resize(m_CoarseGrid->GetNumberOfNodes());
223         m_nums.resize(m_CoarseGrid->GetNumberOfNodes());
224         lorder = m_CoarseGrid->GetElement(0)->GetDoFs();
225         for (k = 0; k < m_CoarseGrid->GetNumberOfNodes(); ++k)
226             m_nums[k] = k;
227         //for (auto elem : m_CoarseGrid->GetElements())
228         for(k = 0; k < m_CoarseGrid->GetNumberOfElements(); ++k)
229         {
230             auto elem{ m_CoarseGrid->GetElement(k) };
231             auto order{ elem->GetDoFs() };
232             for (i = 0; i < order; ++i)
233                 for (j = 0; j < order; ++j)

```

```

234         if (elem->GetNode(j) > elem->GetNode(i))
235             temp[elem->GetNode(j)].insert(elem->GetNode(i));
236     }*/
237     m_GlobalMatrix->Create(temp.size(), temp);
238     m_rhsvector->resize(temp.size());
239     //m_vec.resize(temp.size());
240     return 0;
241 }
242
243 template<class Problem, class Grid, class Matrix>
244 const int system_dg_method<Problem, Grid, Matrix>::AssembleLocalMatrix(const int l)
245 {
246     int i, j, k, nodes;
247     double mij;
248     const auto& elem{ m_CoarseGrid->GetElement(l) };
249     const auto& dofs{ elem->GetDoFs() };
250     nodes = elem->GetNumberOfNodes();
251     std::vector<Mesh::Point> points(nodes);
252     for (i = 0; i < nodes; ++i)
253         points[i] = m_CoarseGrid->GetNode(elem->GetNode(i));
254     for (k = 0; k < m_problem->getNumberOfTerms(); ++k)
255     {
256         switch (m_problem->getTerm(k))
257         {
258             case Terms::EUV:
259                 //for (i = 0; i < dofs; ++i)
260                 //{
261                     // for (j = 0; j < dofs; ++j)
262                     // {
263                         auto M = [&](const Mesh::Point& p)
264                         {
265                             return elem->GetShapeFunction(i, p) * elem->GetShapeFunction(j, p);
266                         };
267                         mij = elem->Integrate(M, points);
268                         m_rhsvector->operator[] (m_nums[l] + i) +=
269                         m_CoarseGrid->getParameter(Parameters::MASS, l, j) * m_CoarseGrid->getSolution(l, j) * mij;
270                     // }
271                 //}
272                 for (size_t j = 0; j < m_sys_size; ++j)
273                     m_rhsvector->operator[] (m_nums[l] + j) += m_problem->get_solution(j, l,
274                     elem->GetType(), points[l]);
275                 break;
276             default:
277                 break;
278         }
279     }
280     return 0;
281 }
282
283 template<class Problem, class Grid, class Matrix>
284 const int system_dg_method<Problem, Grid, Matrix>::AssembleGlobal()
285 {
286     for (int l = 0; l < m_N; ++l)
287         AssembleLocalMatrix(l);
288     return 0;
289 }
290
291 template<class Problem, class Grid, class Matrix>
292 const int system_dg_method<Problem, Grid, Matrix>::AssembleFluxMatrix()
293 {
294     auto Nb{ m_CoarseGrid->GetNumberOfBoundaries() };
295     unsigned int l;
296     switch (m_fluxtype)
297     {
298         case corenc::method::DGFlux::ELaxFriedrichs:
299             for (l = 0; l < Nb; ++l)
300             {
301                 const auto& bound{ m_CoarseGrid->GetBoundary(l) };
302                 const auto& nk{ bound->GetNeighbour(0) };
303                 const auto& ne{ bound->GetNeighbour(1) };
304                 const auto& elemk{ m_CoarseGrid->GetElement(nk) };
305                 const auto& dofs{ bound->GetDoFs() };
306                 const auto& dofsk{ elemk->GetDoFs() };
307                 double C{ 0 };
308                 unsigned int i, j;
309                 std::vector<Mesh::Point> points(dofs);
310                 for (i = 0; i < dofs; ++i)
311                     points[i] = m_CoarseGrid->GetNode(bound->GetNode(i));
312                 if (ne > -1)
313                 {
314                     const auto& eleme{ m_CoarseGrid->GetElement(ne) };
315                     const auto& dofse{ eleme->GetDoFs() };
316                     for (i = 0; i < dofsk; ++i)
317                     {
318                         for (j = 0; j < dofsk; ++j)

```

```

319         auto Mkk = [&](const Mesh::Point& p)
320         {
321             return elemk->GetShapeFunction(j, p) * elemk->GetShapeFunction(i,
322 p);
323         };
324         auto temp{ bound->Integrate(Mkk, points) };
325         C = std::max(fabs(m_CoarseGrid->getSolution(ne, i)),
326 fabs(m_CoarseGrid->getSolution(nk, j)));
327         //m_rhsvector->operator[] (m_nums[nk] + i) +=
328 -0.5*(m_flux(m_CoarseGrid->getSolution(nk, j)) * temp - C * m_CoarseGrid->getSolution(nk, j) * temp);
329         auto val{ -0.5*(m_flux(m_CoarseGrid->getSolution(nk, j)) + C *
330 m_CoarseGrid->getSolution(nk, j)) * temp };
331         m_rhsvector->operator[] (m_nums[nk] + i) += val;
332         //lv[m_nums[nk] + i] += val;
333     }
334     for (i = 0; i < dofsk; ++i)
335     {
336         for (j = 0; j < dofse; ++j)
337         {
338             auto Mke = [&](const Mesh::Point& p)
339             {
340                 return eleme->GetShapeFunction(j, p) * elemk->GetShapeFunction(i,
341 p);
342             };
343             auto temp{ bound->Integrate(Mke, points) };
344             C = std::max(fabs(m_CoarseGrid->getSolution(nk, i)),
345 fabs(m_CoarseGrid->getSolution(ne, j)));
346             //m_rhsvector->operator[] (m_nums[nk] + 7 i) +=
347 -0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) * temp - C * m_CoarseGrid->getSolution(ne, j) * temp);
348             auto val{ -0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) - C *
349 m_CoarseGrid->getSolution(ne, j)) * temp };
350             m_rhsvector->operator[] (m_nums[nk] + i) += val;
351             //ke[m_nums[nk] + i] += val;
352             //lv[m_nums[nk] + i] += val;
353         }
354         for (i = 0; i < dofse; ++i)
355         {
356             for (j = 0; j < dofsk; ++j)
357             {
358                 auto Mek = [&](const Mesh::Point& p)
359                 {
360                     return eleme->GetShapeFunction(i, p) * elemk->GetShapeFunction(j,
361 p);
362                 };
363                 auto temp{ bound->Integrate(Mek, points) };
364                 C = std::max(fabs(m_CoarseGrid->getSolution(nk, j)),
365 fabs(m_CoarseGrid->getSolution(ne, i)));
366                 //m_rhsvector->operator[] (m_nums[ne] + i) +=
367 0.5*(m_flux(m_CoarseGrid->getSolution(nk, j)) * temp - C * m_CoarseGrid->getSolution(nk, j) * temp);
368                 auto val{ 0.5*(m_flux(m_CoarseGrid->getSolution(nk, j)) + C *
369 m_CoarseGrid->getSolution(nk, j)) * temp };
370                 m_rhsvector->operator[] (m_nums[ne] + i) += val;
371                 //ek[m_nums[ne] + i] += val;
372                 //rv[m_nums[ne] + i] += val;
373             }
374             for (i = 0; i < dofse; ++i)
375             {
376                 for (j = 0; j < dofse; ++j)
377                 {
378                     auto Mee = [&](const Mesh::Point& p)
379                     {
380                         return eleme->GetShapeFunction(j, p) * eleme->GetShapeFunction(i,
381 p);
382                     };
383                     auto temp{ bound->Integrate(Mee, points) };
384                     C = std::max(fabs(m_CoarseGrid->getSolution(nk, i)),
385 fabs(m_CoarseGrid->getSolution(ne, j)));
386                     //m_rhsvector->operator[] (m_nums[ne] + i) +=
387 0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) * temp - C * m_CoarseGrid->getSolution(ne, j) * temp);
388                     auto val{ 0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) - C *
389 m_CoarseGrid->getSolution(ne, j)) * temp };
390                     m_rhsvector->operator[] (m_nums[ne] + i) += val;
391                     //ee[m_nums[ne] + i] += val;
392                     //rv[m_nums[ne] + i] += val;
393                 }
394             }
395         }
396     }
397     else
398     {
399         //C = m_flux(m_CoarseGrid->getSolution(nk, 0));
400         //m_rhsvector->operator[] (m_nums[nk]) = C;
401         if (l == 0)
402         {
403             for (i = 0; i < dofsk; ++i)

```

```

391         {
392             for (j = 0; j < dofsk; ++j)
393             {
394                 auto Mkk = [&](const Mesh::Point& p)
395                 {
396                     return elemk->GetShapeFunction(j, p) *
elemk->GetShapeFunction(i, p);
397                 };
398                 auto temp{ bound->Integrate(Mkk, points) };
399                 //m_rhsvector->operator[] (m_nums[nk]+i) -=
((ne+int(l))>0?-1:1)*(m_flux(m_CoarseGrid->getSolution(nk, j)) * temp - C *
m_CoarseGrid->getSolution(nk, j) * temp);
400                 auto fl = m_flux(m_CoarseGrid->getSolution(nk, j)) * temp;
401                 m_rhsvector->operator[] (m_nums[nk] + i) += fl * temp;
402                 //if(C >= 0)
403                 //m_rhsvector->operator[] (m_nums[nk] + i) += ((ne + int(l))>0 ? 1 :
0) * C * temp;
404                 //m_rhsvector->operator[] (m_nums[nk] + i) += 1e10 * C * temp;
405             }
406         }
407     }
408     else
409     {
410         for (i = 0; i < dofsk; ++i)
411         {
412             for (j = 0; j < dofsk; ++j)
413             {
414                 auto Mkk = [&](const Mesh::Point& p)
415                 {
416                     return elemk->GetShapeFunction(j, p) *
elemk->GetShapeFunction(i, p);
417                 };
418                 auto temp{ bound->Integrate(Mkk, points) };
419                 auto fl = m_flux(m_CoarseGrid->getSolution(nk, j)) * temp;
420                 m_rhsvector->operator[] (m_nums[nk] + i) -= fl * temp;
421             }
422         }
423     }
424 }
425 }
426
427 // explicit LF flux
428 break;
429 default:
430     break;
431 }
432 return 0;
433 }
434
435 template<class Problem, class Grid, class Matrix>
436 void system_dg_method<Problem, Grid, Matrix>::assemble_flux(const unsigned l)
437 {
438     const auto& bound{ m_CoarseGrid->GetBoundary(l) };
439     const auto& nk{ bound->GetNeighbour(0) };
440     const auto& ne{ bound->GetNeighbour(1) };
441     const auto& elemk{ m_CoarseGrid->GetElement(nk) };
442     const auto& dofs{ bound->GetDoFs() };
443     const auto& dofsk{ elemk->GetDoFs() };
444     double C{ 0 };
445     unsigned int i, j;
446     std::vector<Mesh::Point> points(dofs);
447     for (i = 0; i < dofs; ++i)
448         points[i] = m_CoarseGrid->GetNode(bound->GetNode(i));
449     C = 2;
450     if (ne > -1)
451     {
452         const auto& eleme{ m_CoarseGrid->GetElement(ne) };
453         const auto& dofse{ eleme->GetDoFs() };
454         for (i = 0; i < dofsk; ++i)
455         {
456             for (j = 0; j < dofsk; ++j)
457             {
458                 auto Mkk = [&](const Mesh::Point& p)
459                 {
460                     return elemk->GetShapeFunction(j, p) * elemk->GetShapeFunction(i, p);
461                 };
462                 auto temp{ bound->Integrate(Mkk, points) };
463                 C = std::max(m_CoarseGrid->getSolution(nk, i), m_CoarseGrid->getSolution(nk,
j));
464                 auto val{ -0.5*(m_flux(m_CoarseGrid->getSolution(nk, j)) + C *
m_CoarseGrid->getSolution(nk, j)) * temp };
465                 m_rhsvector->operator[] (m_nums[nk] + i) += val;
466                 //kk[m_nums[nk] + i] += val;
467                 //lv[m_nums[nk] + i] += val;
468             }
469         }
470         for (i = 0; i < dofsk; ++i)

```

```

471         {
472             for (j = 0; j < dofse; ++j)
473             {
474                 auto Mke = [&](const Mesh::Point& p)
475                 {
476                     return eleme->GetShapeFunction(j, p) * elemk->GetShapeFunction(i, p);
477                 };
478                 auto temp{ bound->Integrate(Mke, points) };
479                 C = std::max(m_CoarseGrid->getSolution(nk, i), m_CoarseGrid->getSolution(ne,
j));
480                 //m_rhsvector->operator[](m_nums[nk] + 7 i) +=
-0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) * temp - C * m_CoarseGrid->getSolution(ne, j) * temp);
481                 auto val{ -0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) - C *
m_CoarseGrid->getSolution(ne, j)) * temp };
482                 m_rhsvector->operator[](m_nums[nk] + i) += val;
483                 //ke[m_nums[nk] + i] += val;
484                 //rv[m_nums[nk] + i] += val;
485             }
486         }
487         for (i = 0; i < dofse; ++i)
488         {
489             for (j = 0; j < dofsk; ++j)
490             {
491                 auto Mek = [&](const Mesh::Point& p)
492                 {
493                     return eleme->GetShapeFunction(i, p) * elemk->GetShapeFunction(j, p);
494                 };
495                 auto temp{ bound->Integrate(Mek, points) };
496                 C = std::max(m_CoarseGrid->getSolution(nk, j), m_CoarseGrid->getSolution(ne,
i));
497                 //m_rhsvector->operator[](m_nums[ne] + i) +=
0.5*(m_flux(m_CoarseGrid->getSolution(nk, j)) * temp - C * m_CoarseGrid->getSolution(nk, j) * temp);
498                 auto val{ 0.5*(m_flux(m_CoarseGrid->getSolution(nk, j)) + C *
m_CoarseGrid->getSolution(nk, j)) * temp };
499                 m_rhsvector->operator[](m_nums[ne] + i) += val;
500                 //ek[m_nums[ne] + i] += val;
501                 //lv[m_nums[ne] + i] += val;
502             }
503         }
504         for (i = 0; i < dofse; ++i)
505         {
506             for (j = 0; j < dofse; ++j)
507             {
508                 auto Mee = [&](const Mesh::Point& p)
509                 {
510                     return eleme->GetShapeFunction(j, p) * eleme->GetShapeFunction(i, p);
511                 };
512                 auto temp{ bound->Integrate(Mee, points) };
513                 C = std::max(m_CoarseGrid->getSolution(ne, i), m_CoarseGrid->getSolution(ne,
j));
514                 //m_rhsvector->operator[](m_nums[ne] + i) +=
0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) * temp - C * m_CoarseGrid->getSolution(ne, j) * temp);
515                 auto val{ 0.5*(m_flux(m_CoarseGrid->getSolution(ne, j)) - C *
m_CoarseGrid->getSolution(ne, j)) * temp };
516                 m_rhsvector->operator[](m_nums[ne] + i) += val;
517                 //ee[m_nums[ne] + i] += val;
518                 //rv[m_nums[ne] + i] += val;
519             }
520         }
521     }
522     else
523     {
524         for (i = 0; i < dofsk; ++i)
525         {
526             for (j = 0; j < dofsk; ++j)
527             {
528                 auto Mkk = [&](const Mesh::Point& p)
529                 {
530                     return elemk->GetShapeFunction(j, p) * elemk->GetShapeFunction(i, p);
531                 };
532                 auto temp{ bound->Integrate(Mkk, points) };
533                 auto fl = m_flux(m_CoarseGrid->getSolution(nk, j));
534                 C = m_CoarseGrid->getSolution(nk, j);
535                 m_rhsvector->operator[](m_nums[nk] + i) += ((ne + int(1))>0 ? 0 : 1) * fl * temp;
536             }
537         }
538     }
539 }
540 template<class Problem, class Grid, class Matrix>
541 void system_dg_method<Problem, Grid, Matrix>::MainConditions()
542 {
543 }
544 }
545
546 template<class Problem, class Grid, class Matrix>
547 void system_dg_method<Problem, Grid, Matrix>::SecondConditions()
548 {

```

```

549     }
550 }
551
552 template<class Problem, class Grid, class Matrix>
553 void system_dg_method<Problem, Grid, Matrix>::ThirdConditions()
554 {
555 }
556 }
557
558 template<class Problem, class Grid, class Matrix>
559 const double system_dg_method<Problem, Grid, Matrix>::GetMaxSolution() const
560 {
561     return 0.;
562 }
563
564 template<class Problem, class Grid, class Matrix>
565 const double system_dg_method<Problem, Grid, Matrix>::GetMinSolution() const
566 {
567     return 0.;
568 }
569
570 template<class Problem, class Grid, class Matrix>
571 const double system_dg_method<Problem, Grid, Matrix>::GetSolution(const std::vector<double>&
point) const
572 {
573     return 0.;
574 }
575 }
576 }
577 #endif // !CORENC_METHODS_SYSTEM_DG_METHOD_H_

```

7.90 CoreNCFEM/multi_vector.h File Reference

```

#include <vector>
#include <cstdint>
#include <cstdint>

```

Classes

- class [corenc::multi_vector< T >](#)

Namespaces

- namespace [corenc](#)

Macros

- #define [CORENC_MULTI_VECTOR_H_](#)

7.90.1 Macro Definition Documentation

7.90.1.1 CORENC_MULTI_VECTOR_H_

```
#define CORENC_MULTI_VECTOR_H_
```


7.91 multi_vector.h

[Go to the documentation of this file.](#)

```

1  #pragma once
2  #ifndef CORENC_MULTI_VECTOR_H_
3  #define CORENC_MULTI_VECTOR_H_
4  #include <vector>
5  #include <cstdint>
6  #include <cstring>
7  namespace corenc
8  {
9      template<class T>
10         class multi_vector
11         {
12         public:
13             multi_vector();
14             // dim = 1 vector, dim = 2 matrix, etc
15             // block x ... x block; dim times
16             multi_vector(const size_t block, const size_t dim);
17             multi_vector(const size_t dim);
18             ~multi_vector();
19             const T          get(const size_t i...) const;
20             const T          get(const std::vector<size_t>& i) const;
21             const int         set(const T& element, const std::vector<size_t>& index);
22             //const int       set(const T& element, const size_t i...);
23             const int         fill_inc();
24             void              resize(const size_t block);
25             void              resize(const size_t block, const size_t dim);
26             const size_t      size() const;
27             const size_t      totalsize() const;
28         private:
29             std::vector<T>    m_vector;
30             size_t            m_dim;
31             size_t            m_block;
32             size_t            m_totalsize;
33         };
34
35         template<class T>
36         multi_vector<T>::multi_vector()
37         {
38         }
39
40         template<class T>
41         multi_vector<T>::multi_vector(const size_t block, const size_t dim)
42         {
43             m_block = block;
44             m_dim = dim;
45             m_totalsize = 1;
46             for (size_t i = 0; i < m_dim; ++i, m_totalsize *= block);
47             m_vector.resize(m_totalsize);
48         }
49
50         template<class T>
51         multi_vector<T>::multi_vector(const size_t dim)
52         {
53             m_block = 0;
54             m_dim = dim;
55             m_totalsize = 0;
56         }
57
58         template<class T>
59         multi_vector<T>::~multi_vector()
60         {
61         }
62
63         template<class T>
64         const size_t multi_vector<T>::size() const
65         {
66             return m_block;
67         }
68
69         template<class T>
70         const size_t multi_vector<T>::totalsize() const
71         {
72             return m_totalsize;
73         }
74
75         template<class T>
76         const T multi_vector<T>::get(const size_t i...) const
77         {
78             va_list args;
79             va_start(args, i);
80
81             va_end(args);
82             return m_vector[i];
83         }
84
85         template<class T>
86         const T multi_vector<T>::get(const std::vector<size_t>& i) const
87         {
88         }

```

```

83         if (i.size() != m_dim)
84             return T(0);
85         size_t ind = 0;
86         for (size_t j = 0; j < m_dim; ++j)
87         {
88             size_t l = 1;
89             const int lim = m_dim - j - 1;
90             for (int k = 0; k < lim; ++k, l *= m_block);
91             ind += i[j] * l;
92         }
93         return m_vector[ind];
94     }
95     template<class T>
96     void multi_vector<T>::resize(const size_t block)
97     {
98         m_block = block;
99         m_totalsize = 1;
100        for (size_t i = 0; i < m_dim; ++i, m_totalsize *= block);
101        m_vector.resize(m_totalsize);
102    }
103    template<class T>
104    void multi_vector<T>::resize(const size_t block, const size_t dim)
105    {
106        m_block = block;
107        m_dim = dim;
108        m_totalsize = 1;
109        for (size_t i = 0; i < m_dim; ++i, m_totalsize *= block);
110        m_vector.resize(m_totalsize);
111    }
112    template<class T>
113    const int multi_vector<T>::fill_inc()
114    {
115        for (size_t i = 0; i < m_totalsize; ++i)
116            m_vector[i] = i;
117        return 0;
118    }
119    template<class T>
120    const int multi_vector<T>::set(const T& element, const std::vector<size_t>& i)
121    {
122        if (i.size() != m_dim)
123            return 1;
124        size_t ind = 0;
125        for (size_t j = 0; j < m_dim; ++j)
126        {
127            size_t l = 1;
128            const int lim = m_dim - j - 1;
129            for (int k = 0; k < lim; ++k, l *= m_block);
130            ind += i[j] * l;
131        }
132        m_vector[ind] = element;
133        return 0;
134    }
135 }
136 #endif // !CORENC_MULTI_VECTOR_H_

```

7.92 CoreNCFEM/Parameter.cpp File Reference

```
#include "Parameter.h"
```

7.93 CoreNCFEM/Parameter.h File Reference

```
#include "Point.h"
#include <functional>
```

Classes

- class [corenc::Mesh::parameter< T >](#)
- class [corenc::Mesh::point_source< T >](#)
- class [corenc::Mesh::CParameter](#)

Namespaces

- namespace `corenc`
- namespace `corenc::Mesh`

Macros

- `#define CORENC_MESH_PARAMETER_H_`

7.93.1 Macro Definition Documentation

7.93.1.1 CORENC_MESH_PARAMETER_H_

```
#define CORENC_MESH_PARAMETER_H_
```

7.94 Parameter.h

[Go to the documentation of this file.](#)

```
1 // OK. DESCRIPTION.
2 // Here the known parameters are described. it is used then with meshes and problems etc.
3 #pragma once
4 #ifndef CORENC_MESH_PARAMETER_H_
5 #define CORENC_MESH_PARAMETER_H_
6
7 #include "Point.h"
8 #include <functional>
9 namespace corenc
10 {
11     namespace Mesh
12     {
13         template<class T>
14         class parameter
15         {
16         public:
17             using cfunc = std::function<const T(const int, const int, const Point&)>;
18             using cfunc_old = std::function<const T(const int, const Point&)>;
19             parameter() :m_func{ [=](const int, const int, const Point&) {return T(); } } {};
20             parameter(const cfunc& func):m_func{func}{}
21             parameter(const cfunc_old& func)
22             {
23                 cfunc f = [=](const int, const int n, const Point& p) {return func(n, p); };
24                 m_func = f;
25             }
26             parameter(const double _p) :m_func{ [=](const int, const int, const Point&) {return _p; } }
27         {}
28         parameter(const Mesh::Point _p) :m_func{ [=](const int, const int, const Point&) {return _p;
29         } } {}
30         parameter(const parameter<T>& _p) :m_func{ _p.m_func } {}
31         ~parameter() {};
32         const T get(const Point& p) const { return m_func(0, 0, p); };
33         const T get(const int number, const Point& p) const { return m_func(0, number, p); };
34         const T get(const int element, const int node, const Point& p) const { return
35         m_func(element, node, p); };
36         void set(const cfunc& func) { m_func = func; };
37     private:
38         cfunc m_func;
39     };
40
41     template<class T>
42     class point_source
43     {
44     public:
45         point_source() : m_point(Mesh::Point(0,0,0)), m_value(T(0)) {};
46         point_source(const Mesh::Point& p, const T& val) : m_point(p), m_value(val) {};
```

```

44         const T          get_value() const { return m_value; };
45         const Mesh::Point get_point() const { return m_point; };
46         point_source<T>& operator=(const point_source<T>& ps)
47         {
48             m_point = ps.m_point;
49             m_value = ps.m_value;
50             return *this;
51         }
52     private:
53         Mesh::Point    m_point;
54         T              m_value;
55     };
56     class CParameter
57     {
58     public:
59         CParameter();
60         //CParameter(const double _diff, const double _adv, const double _mass);
61         CParameter(const parameter<double>& _diff, const parameter<double>& _adv, const
parameter<double>& _mass);
62         CParameter(const Parameters&, const parameter<double>&);
63         ~CParameter();
64         const double      GetDiffusion() const;
65         const double      GetAdvection() const;
66         const double      GetMass() const;
67         const double      GetDiffusion(const Point&) const;
68         const double      GetAdvection(const Point&) const;
69         const double      GetMass(const Point&) const;
70     private:
71         parameter<double> m_diffusion;
72         parameter<double> m_advection;
73         parameter<double> m_mass;
74     };
75 }
76 }
77
78 #endif // !CORENC_MESH_PARAMETER_H_

```

7.95 CoreNCFEM/Point.cpp File Reference

```

#include "Point.h"
#include <cmath>

```

7.96 CoreNCFEM/Point.h File Reference

```

#include <cmath>
#include <vector>

```

Classes

- class [corenc::Mesh::Point](#)
- struct [corenc::Mesh::GaussTriangle](#)
- struct [corenc::Mesh::GaussRectangular](#)
- struct [corenc::Mesh::Gauss1dim](#)
- struct [corenc::Mesh::Gauss1dimN< N >](#)
- struct [corenc::Mesh::GaussTetrahedron](#)
- struct [corenc::Mesh::GaussRectangularCubic](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::Mesh](#)

Macros

- `#define CORENC_MESH_Point_h`

Enumerations

- enum class `corenc::Terms` {
`corenc::IUV` , `corenc::IDUDV` , `corenc::IDUV` , `corenc::IUDV` ,
`corenc::EUV` , `corenc::EDUDV` , `corenc::EDUV` , `corenc::EUDV` ,
`corenc::EFV` , `corenc::RUV` , `corenc::SUPG` }
- enum class `corenc::Parameters` { `corenc::DIFFUSION` , `corenc::MASS` , `corenc::ADVECTION` }

7.96.1 Macro Definition Documentation

7.96.1.1 CORENC_MESH_Point_h

```
#define CORENC_MESH_Point_h
```

7.97 Point.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef CORENC_MESH_Point_h
3 #define CORENC_MESH_Point_h
4 #include <cmath>
5 #include <vector>
6 namespace corenc
7 {
8     enum class Terms
9     {
10         // left-side
11         // uv
12         IUV,
13         // grad u grad v
14         IDUDV,
15         // grad u v
16         IDUV,
17         // u grad v
18         IUDV,
19         // right-side
20         EUV,
21         EDUDV,
22         EDUV,
23         EUDV,
24         EFV,
25         // right-side matrix
26         RUV,
27         SUPG,
28     };
29
30     enum class Parameters
31     {
32         DIFFUSION,
33         MASS,
34         ADVECTION
35     };
36
37     namespace Mesh
38     {
39         class Point
40         {
41         public:
42             Point() :x{ 0 }, y{ 0 }, z{ 0 } {}
```

```

43     Point(const double _x, const double _y) :
44         x{ _x }, y{ _y }, z{ 0 } {}
45     Point(const double _x, const double _y, const double _z) :
46         x{ _x }, y{ _y }, z{ _z } {}
47     Point(const Point& p) :
48         x{ p.x }, y{ p.y }, z{ p.z } {}
49     double x, y, z;
50     const double Jacobian() const { return 1; }
51     Point& operator=(const Point& p)
52     {
53         x = p.x;
54         y = p.y;
55         z = p.z;
56         return *this;
57     }
58     const bool operator==(const Point& p)
59     {
60         const double eps{ 1e-13 };
61         if (fabs(x - p.x) < eps)
62             if (fabs(y - p.y) < eps)
63                 if (fabs(z - p.z) < eps)
64                     return true;
65         return false;
66     }
67     friend const bool operator!=(const Point& p1, const Point& p2)
68     {
69         const double eps{ 1e-13 };
70         if (fabs(p1.x - p2.x) < eps)
71             if (fabs(p1.y - p2.y) < eps)
72                 if (fabs(p1.z - p2.z) < eps)
73                     return false;
74         return true;
75     }
76     const bool operator<(const Point& p2)
77     {
78         return (x < p2.x);
79     }
80     friend const double operator*(const Point& lhs, const Point& rhs)
81     {
82         return lhs.x * rhs.x + lhs.y * rhs.y + lhs.z * rhs.z;
83     }
84     const Point operator*(const double rhs)
85     {
86         return Point{ x * rhs, y * rhs, z * rhs };
87     }
88     Point& operator+=(const Point& rhs)
89     {
90         x += rhs.x;
91         y += rhs.y;
92         z += rhs.z;
93         return *this;
94     }
95     Point& operator*=(const double rhs)
96     {
97         x *= rhs;
98         y *= rhs;
99         z *= rhs;
100        return *this;
101    }
102    friend const Point operator*(const Point& lhs, const double rhs)
103    {
104        return Point{ rhs * lhs.x, rhs * lhs.y, rhs * lhs.z };
105    }
106    friend const Point operator*(const double lhs, const Point& rhs)
107    {
108        return Point{ lhs * rhs.x, lhs * rhs.y, lhs * rhs.z };
109    }
110    friend const Point operator+(const Point& lhs, const Point& rhs)
111    {
112        return Point{ lhs.x + rhs.x, lhs.y + rhs.y, lhs.z + rhs.z };
113    }
114    friend const Point operator-(const Point& lhs, const Point& rhs)
115    {
116        return Point{ lhs.x - rhs.x, lhs.y - rhs.y, lhs.z - rhs.z };
117    }
118 };
119
120 struct GaussTriangle
121 {
122     const static double m_tra[];
123     const static double m_trb[];
124     const static double m_sqrt15;
125     const static double m_trw[];
126     const static int    m_order;
127 };
128
129 struct GaussRectangular

```

```

130     {
131         const static double m_ra[];
132         const static double m_rb[];
133         const static double m_rw[];
134         const static double m_a;
135         const static double m_b;
136         const static double m_c;
137         const static double m_wa;
138         const static double m_wb;
139         const static double m_wc;
140     };
141     struct Gaussldim
142     {
143         const static int m_order;
144         const static double m_a[];
145         const static double m_sqrt35;
146         const static double m_w[];
147     };
148
149     template<int N>
150     struct GaussldimN
151     {
152         const static int m_order;
153         const static double m_a[];
154         const static double m_w[];
155     };
156
157
158     struct GaussTetrahedron
159     {
160         const static double m_la[];
161         const static double m_lb[];
162         const static double m_lc[];
163         const static double m_ld[];
164         const static double m_w[];
165         const static double m_psq, m_msq;
166     };
167
168     struct GaussRectangularCubic
169     {
170         const static double m_ra[];
171         const static double m_rb[];
172         const static double m_rc[];
173         const static double m_rw[];
174         const static double m_a;
175         const static double m_b;
176         const static double m_c;
177         const static double m_w1;
178         const static double m_w2;
179         const static double m_w3;
180         const static double m_w4;
181         const static int m_s{ 34 };
182     };
183 }
184 }
185 #endif /* CORENC_MESH_Point_h */

```

7.98 main.cpp File Reference

```

#include <iostream>
#include "colors.h"
#include "Tests/test_cases.h"

```

Functions

- int [main](#) (int argc, char *argv[])

7.98.1 Function Documentation

7.98.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

7.99 Problems/BurgersScalar.cpp File Reference

```
#include "BurgersScalar.h"
#include <vector>
```

7.100 Problems/BurgersScalar.h File Reference

```
#include "Problems.h"
#include <vector>
```

Classes

- class [corenc::CBurgersScalar](#)

Namespaces

- namespace [corenc](#)

7.101 BurgersScalar.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_PROBLEMS_BURGERS_H_
2 #define CORENC_PROBLEMS_BURGERS_H_
3
4 #include "Problems.h"
5 #include <vector>
6 namespace corenc
7 {
8     class CBurgersScalar : public CProblem
9     {
10     public:
11         CBurgersScalar();
12         ~CBurgersScalar();
13         Terms          getTerm(const unsigned int) const;
14         const unsigned int getNumberOfTerms() const;
15         const int          setTerm(const unsigned int, const Terms&);
16         const int          addTerm(const Terms&);
17         const double       getFlux(const double) const;
18         const int          removeTerm(const Terms&);
19         const int          load_parameters(const std::string& file_name);
20     private:
21         std::vector<Terms> m_terms;
22     };
23 }
24 #endif // !CORENC_PROBLEMS_BURGERS_H_
```


7.102 Problems/DiffusionScalar.cpp File Reference

```
#include "DiffusionScalar.h"
#include <vector>
#include <istream>
#include <iostream>
#include <fstream>
```

7.103 Problems/DiffusionScalar.h File Reference

```
#include "Problems.h"
#include <vector>
#include "../CoreNCFEM/Parameter.h"
#include <map>
#include <tuple>
```

Classes

- class [corenc::CDiffusionScalar](#)

Namespaces

- namespace [corenc](#)

7.104 DiffusionScalar.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_PROBLEMS_DIFFUSIONSCALAR_H_
2 #define CORENC_PROBLEMS_DIFFUSIONSCALAR_H_
3
4 #include "Problems.h"
5 #include <vector>
6 #include "../CoreNCFEM/Parameter.h"
7 #include <map>
8 #include <tuple>
9 namespace corenc
10 {
11     class CDiffusionScalar : public CProblem
12     {
13     public:
14         using boundary = std::tuple<int, Mesh::parameter<double>, Mesh::parameter<double>>;
15         CDiffusionScalar();
16         ~CDiffusionScalar();
17         Terms getTerm(const unsigned int) const;
18         const unsigned int getNumberOfTerms() const;
19         const int findTerm(const Terms&) const;
20         const int setTerm(const unsigned int, const Terms&);
21         const int addTerm(const Terms&);
22         const int removeTerm(const Terms&);
23         const int load_parameters(const std::string& file_name);
24         const double get_parameter(const Terms&, const int element_type, const Mesh::Point&) const;
25         const double get_parameter(const Terms&, const int element_number, const int element_type, const Mesh::Point&) const;
26         const Mesh::Point get_parameter(const Terms&, const int element_number, const int element_type, const Mesh::Point&, const int) const;
27         const double get_parameter(const Terms&, const int element_type, const int element_number, const int node, const Mesh::Point&) const;
```

```

28     const Mesh::Point                get_parameter(const Terms&, const int element_type,
const int element_number, const int node, const Mesh::Point&, const int v) const;
29     const double                    get_boundary_parameter(const int type, const int
element_type, const Mesh::Point&) const;
30     const double                    get_boundary_parameter(const int type, const int
element_type, const int element_number, const Mesh::Point&) const;
31     const double                    get_boundary_parameter(const int type, const int
element_type, const int element_number, const int node, const Mesh::Point&) const;
32     const int                      get_number_of_boundaries() const;
33     const int                      get_boundary_type(const int number) const;
34     const int                      add_parameter(const Terms&, const int element_type, const
double& value);
35     const int                      add_parameter(const Terms&, const int element_type, const
Mesh::parameter<double>& value);
36     const int                      add_parameter(const Terms&, const int element_type, const
Mesh::parameter<Mesh::Point>& value);
37     const int                      set_parameter(const Terms&, const int element_type, const
Mesh::parameter<double>& value);
38     const int                      set_parameter(const Terms&, const int element_type, const
Mesh::parameter<Mesh::Point>& value);
39     const int                      set_boundary_parameter(const int type, const int
element_type, const boundary& value);
40                                     // 1st and 2nd types of boundary conditions
41     const int                      add_boundary_parameter(const int type, const int
element_type, const Mesh::parameter<double>& value);
42                                     // 3rd type of boundary conditions
43     const int                      add_boundary_parameter(const int element_type, const
Mesh::parameter<double>& value, const Mesh::parameter<double>& value2);
44     const Mesh::point_source<double> get_point_source(const int number) const;
45     void                          set_point_source(const int number, const
Mesh::point_source<double>&);
46     const int                      get_total_sources() const;
47 private:
48     std::vector<Terms>              m_terms;
49     std::map<int, Mesh::parameter<double>> m_params;
50     std::map<int, Mesh::parameter<Mesh::Point>> m_vels;
51     std::map<int, Mesh::parameter<double>> m_srcs;
52     std::map<int, Mesh::parameter<double>> m_gams;
53     std::map<int, boundary>          m_bounds;
54     //std::map<int, Mesh::point_source<double>> m_pointsrcs;
55     std::vector<Mesh::point_source<double>> m_pointsrcs;
56     int                              m_total_params;
57     int                              m_total_srcs;
58     int                              m_total_gams;
59     int                              m_total_bounds;
60 };
61 }
62 #endif // !CORENC_PROBLEMS_DIFFUSIONSCALAR_H_

```

7.105 Problems/Problems.h File Reference

```

#include "../CoreNCFEM/Point.h"
#include <string>

```

Classes

- class [corenc::CProblem](#)

Namespaces

- namespace [corenc](#)

Macros

- #define [CORENC_PROBLEMS_PROBLEMS_H_](#)

7.105.1 Macro Definition Documentation

7.105.1.1 CORENC_PROBLEMS_PROBLEMS_H_

```
#define CORENC_PROBLEMS_PROBLEMS_H_
```

7.106 Problems.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef CORENC_PROBLEMS_PROBLEMS_H_
3 #define CORENC_PROBLEMS_PROBLEMS_H_
4 #include "../CoreNCFEM/Point.h"
5 #include <string>
6
7 namespace corenc
8 {
9     class CProblem
10     {
11     public:
12         CProblem() {}
13         virtual ~CProblem() {}
14         virtual Terms          getTerm(const unsigned int) const = 0;
15         virtual const unsigned int getNumberOfTerms() const = 0;
16         virtual const int          setTerm(const unsigned int, const Terms&) = 0;
17         virtual const int          addTerm(const Terms&) = 0;
18         virtual const int          load_parameters(const std::string& file_name) = 0;
19     };
20 }
21
22 #endif // !CORENC_PROBLEMS_PROBLEMS_H_
```

7.107 Problems/ShallowWater.cpp File Reference

```
#include "ShallowWater.h"
#include <vector>
#include <istream>
#include <iostream>
#include <fstream>
```

7.108 Problems/ShallowWater.h File Reference

```
#include "Problems.h"
#include <vector>
#include "../CoreNCFEM/Parameter.h"
#include <map>
#include <tuple>
```

Classes

- class [corenc::CShallowWater](#)

Namespaces

- namespace `corenc`

7.109 ShallowWater.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_PROBLEMS_SHALLOWWATER_H_
2 #define CORENC_PROBLEMS_SHALLOWWATER_H_
3
4 #include "Problems.h"
5 #include <vector>
6 #include "../CoreNCFEM/Parameter.h"
7 #include <map>
8 #include <tuple>
9 namespace corenc
10 {
11     class CShallowWater : public CProblem
12     {
13     public:
14         using boundary = std::tuple<int, Mesh::parameter<double>, Mesh::parameter<double>>;
15         CShallowWater();
16         ~CShallowWater();
17         Terms
18         const unsigned int
19         const int
20         const int
21         const int
22         const int
23         const double
24         Mesh::Point& const;
25         const double
26         const int element_type, const Mesh::Point& const;
27         const double
28         element_type, const Mesh::Point& const;
29         const int
30         const int
31         Mesh::parameter<double>& value);
32         const int
33         Mesh::parameter<double>& value);
34         const int
35         element_type, const Mesh::parameter<double>& value);
36         // 1st and 2nd types of boundary conditions
37         const int
38         element_type, const Mesh::parameter<double>& value);
39         // 3rd type of boundary conditions
40         const int
41         Mesh::parameter<double>& value, const Mesh::parameter<double>& value2);
42     private:
43         std::vector<Terms>
44         std::map<int, Mesh::parameter<double>>
45         std::map<int, Mesh::parameter<double>>
46         std::map<int, boundary>
47         int
48         int
49         int
50         int
51         int
52         int
53         int
54         int
55         int
56         int
57         int
58         int
59         int
60         int
61         int
62         int
63         int
64         int
65         int
66         int
67         int
68         int
69         int
70         int
71         int
72         int
73         int
74         int
75         int
76         int
77         int
78         int
79         int
80         int
81         int
82         int
83         int
84         int
85         int
86         int
87         int
88         int
89         int
90         int
91         int
92         int
93         int
94         int
95         int
96         int
97         int
98         int
99         int
100        int
101        int
102        int
103        int
104        int
105        int
106        int
107        int
108        int
109        int
110        int
111        int
112        int
113        int
114        int
115        int
116        int
117        int
118        int
119        int
120        int
121        int
122        int
123        int
124        int
125        int
126        int
127        int
128        int
129        int
130        int
131        int
132        int
133        int
134        int
135        int
136        int
137        int
138        int
139        int
140        int
141        int
142        int
143        int
144        int
145        int
146        int
147        int
148        int
149        int
150        int
151        int
152        int
153        int
154        int
155        int
156        int
157        int
158        int
159        int
160        int
161        int
162        int
163        int
164        int
165        int
166        int
167        int
168        int
169        int
170        int
171        int
172        int
173        int
174        int
175        int
176        int
177        int
178        int
179        int
180        int
181        int
182        int
183        int
184        int
185        int
186        int
187        int
188        int
189        int
190        int
191        int
192        int
193        int
194        int
195        int
196        int
197        int
198        int
199        int
200        int
201        int
202        int
203        int
204        int
205        int
206        int
207        int
208        int
209        int
210        int
211        int
212        int
213        int
214        int
215        int
216        int
217        int
218        int
219        int
220        int
221        int
222        int
223        int
224        int
225        int
226        int
227        int
228        int
229        int
230        int
231        int
232        int
233        int
234        int
235        int
236        int
237        int
238        int
239        int
240        int
241        int
242        int
243        int
244        int
245        int
246        int
247        int
248        int
249        int
250        int
251        int
252        int
253        int
254        int
255        int
256        int
257        int
258        int
259        int
260        int
261        int
262        int
263        int
264        int
265        int
266        int
267        int
268        int
269        int
270        int
271        int
272        int
273        int
274        int
275        int
276        int
277        int
278        int
279        int
280        int
281        int
282        int
283        int
284        int
285        int
286        int
287        int
288        int
289        int
290        int
291        int
292        int
293        int
294        int
295        int
296        int
297        int
298        int
299        int
300        int
301        int
302        int
303        int
304        int
305        int
306        int
307        int
308        int
309        int
310        int
311        int
312        int
313        int
314        int
315        int
316        int
317        int
318        int
319        int
320        int
321        int
322        int
323        int
324        int
325        int
326        int
327        int
328        int
329        int
330        int
331        int
332        int
333        int
334        int
335        int
336        int
337        int
338        int
339        int
340        int
341        int
342        int
343        int
344        int
345        int
346        int
347        int
348        int
349        int
350        int
351        int
352        int
353        int
354        int
355        int
356        int
357        int
358        int
359        int
360        int
361        int
362        int
363        int
364        int
365        int
366        int
367        int
368        int
369        int
370        int
371        int
372        int
373        int
374        int
375        int
376        int
377        int
378        int
379        int
380        int
381        int
382        int
383        int
384        int
385        int
386        int
387        int
388        int
389        int
390        int
391        int
392        int
393        int
394        int
395        int
396        int
397        int
398        int
399        int
400        int
401        int
402        int
403        int
404        int
405        int
406        int
407        int
408        int
409        int
410        int
411        int
412        int
413        int
414        int
415        int
416        int
417        int
418        int
419        int
420        int
421        int
422        int
423        int
424        int
425        int
426        int
427        int
428        int
429        int
430        int
431        int
432        int
433        int
434        int
435        int
436        int
437        int
438        int
439        int
440        int
441        int
442        int
443        int
444        int
445        int
446        int
447        int
448        int
449        int
450        int
451        int
452        int
453        int
454        int
455        int
456        int
457        int
458        int
459        int
460        int
461        int
462        int
463        int
464        int
465        int
466        int
467        int
468        int
469        int
470        int
471        int
472        int
473        int
474        int
475        int
476        int
477        int
478        int
479        int
480        int
481        int
482        int
483        int
484        int
485        int
486        int
487        int
488        int
489        int
490        int
491        int
492        int
493        int
494        int
495        int
496        int
497        int
498        int
499        int
500        int
501        int
502        int
503        int
504        int
505        int
506        int
507        int
508        int
509        int
510        int
511        int
512        int
513        int
514        int
515        int
516        int
517        int
518        int
519        int
520        int
521        int
522        int
523        int
524        int
525        int
526        int
527        int
528        int
529        int
530        int
531        int
532        int
533        int
534        int
535        int
536        int
537        int
538        int
539        int
540        int
541        int
542        int
543        int
544        int
545        int
546        int
547        int
548        int
549        int
550        int
551        int
552        int
553        int
554        int
555        int
556        int
557        int
558        int
559        int
560        int
561        int
562        int
563        int
564        int
565        int
566        int
567        int
568        int
569        int
570        int
571        int
572        int
573        int
574        int
575        int
576        int
577        int
578        int
579        int
580        int
581        int
582        int
583        int
584        int
585        int
586        int
587        int
588        int
589        int
590        int
591        int
592        int
593        int
594        int
595        int
596        int
597        int
598        int
599        int
600        int
601        int
602        int
603        int
604        int
605        int
606        int
607        int
608        int
609        int
610        int
611        int
612        int
613        int
614        int
615        int
616        int
617        int
618        int
619        int
620        int
621        int
622        int
623        int
624        int
625        int
626        int
627        int
628        int
629        int
630        int
631        int
632        int
633        int
634        int
635        int
636        int
637        int
638        int
639        int
640        int
641        int
642        int
643        int
644        int
645        int
646        int
647        int
648        int
649        int
650        int
651        int
652        int
653        int
654        int
655        int
656        int
657        int
658        int
659        int
660        int
661        int
662        int
663        int
664        int
665        int
666        int
667        int
668        int
669        int
670        int
671        int
672        int
673        int
674        int
675        int
676        int
677        int
678        int
679        int
680        int
681        int
682        int
683        int
684        int
685        int
686        int
687        int
688        int
689        int
690        int
691        int
692        int
693        int
694        int
695        int
696        int
697        int
698        int
699        int
700        int
701        int
702        int
703        int
704        int
705        int
706        int
707        int
708        int
709        int
710        int
711        int
712        int
713        int
714        int
715        int
716        int
717        int
718        int
719        int
720        int
721        int
722        int
723        int
724        int
725        int
726        int
727        int
728        int
729        int
730        int
731        int
732        int
733        int
734        int
735        int
736        int
737        int
738        int
739        int
740        int
741        int
742        int
743        int
744        int
745        int
746        int
747        int
748        int
749        int
750        int
751        int
752        int
753        int
754        int
755        int
756        int
757        int
758        int
759        int
760        int
761        int
762        int
763        int
764        int
765        int
766        int
767        int
768        int
769        int
770        int
771        int
772        int
773        int
774        int
775        int
776        int
777        int
778        int
779        int
780        int
781        int
782        int
783        int
784        int
785        int
786        int
787        int
788        int
789        int
790        int
791        int
792        int
793        int
794        int
795        int
796        int
797        int
798        int
799        int
800        int
801        int
802        int
803        int
804        int
805        int
806        int
807        int
808        int
809        int
810        int
811        int
812        int
813        int
814        int
815        int
816        int
817        int
818        int
819        int
820        int
821        int
822        int
823        int
824        int
825        int
826        int
827        int
828        int
829        int
830        int
831        int
832        int
833        int
834        int
835        int
836        int
837        int
838        int
839        int
840        int
841        int
842        int
843        int
844        int
845        int
846        int
847        int
848        int
849        int
850        int
851        int
852        int
853        int
854        int
855        int
856        int
857        int
858        int
859        int
860        int
861        int
862        int
863        int
864        int
865        int
866        int
867        int
868        int
869        int
870        int
871        int
872        int
873        int
874        int
875        int
876        int
877        int
878        int
879        int
880        int
881        int
882        int
883        int
884        int
885        int
886        int
887        int
888        int
889        int
890        int
891        int
892        int
893        int
894        int
895        int
896        int
897        int
898        int
899        int
900        int
901        int
902        int
903        int
904        int
905        int
906        int
907        int
908        int
909        int
910        int
911        int
912        int
913        int
914        int
915        int
916        int
917        int
918        int
919        int
920        int
921        int
922        int
923        int
924        int
925        int
926        int
927        int
928        int
929        int
930        int
931        int
932        int
933        int
934        int
935        int
936        int
937        int
938        int
939        int
940        int
941        int
942        int
943        int
944        int
945        int
946        int
947        int
948        int
949        int
950        int
951        int
952        int
953        int
954        int
955        int
956        int
957        int
958        int
959        int
960        int
961        int
962        int
963        int
964        int
965        int
966        int
967        int
968        int
969        int
970        int
971        int
972        int
973        int
974        int
975        int
976        int
977        int
978        int
979        int
980        int
981        int
982        int
983        int
984        int
985        int
986        int
987        int
988        int
989        int
990        int
991        int
992        int
993        int
994        int
995        int
996        int
997        int
998        int
999        int
1000       int

```

7.110 Solvers/dg_solver.h File Reference

```

#include "../CoreNCFEM/Grids/TriangularMesh.h"
#include "../CoreNCFEM/Methods/DGMethod.h"
#include "../Problems/DiffusionScalar.h"
#include "../CoreNCA/MatrixSkyline.h"
#include "../CoreNCFEM/Methods/FEAnalysis.h"

```

Classes

- class `corenc::solvers::dg_solver< _Problem, _Mesh, _Result >`

Namespaces

- namespace `corenc`
- namespace `corenc::solvers`

7.111 dg_solver.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_SOLVERS_DG_SOLVER_H_
2 #define CORENC_SOLVERS_DG_SOLVER_H_
3
4 #include "../CoreNCFEM/Grids/TriangularMesh.h"
5 #include "../CoreNCFEM/Methods/DGMethod.h"
6 #include "../Problems/DiffusionScalar.h"
7 #include "../CoreNCA/MatrixSkyline.h"
8 #include "../CoreNCFEM/Methods/FEAnalysis.h"
9
10 namespace corenc
11 {
12     namespace solvers
13     {
14         template<class _Problem, class _Mesh, class _Result>
15         class dg_solver
16         {
17             using _Method = method::DGMethod<_Problem, _Mesh, Algebra::MatrixSkyline>;
18         public:
19             dg_solver() :m_method{ nullptr } {}
20             ~dg_solver()
21             {
22                 if (m_method != nullptr)
23                     delete m_method;
24             }
25             // terms, method, mesh, solver, result
26             const int elliptic_solver(_Problem*, _Mesh*, _Result*);
27             const double get_value(const _Mesh&, const _Result&, const Mesh::Point& p) const;
28             const double get_value(const _Method*, const _Mesh&, const _Result&, const
Mesh::Point& p) const;
29             const double get_value(const _Mesh&, const _Result&, const Mesh::Point& p, const
int i) const;
30             const Mesh::Point get_gradvalue(const _Mesh&, const _Result&, const Mesh::Point& p)
const;
31             const Mesh::Point get_gradvalue(const _Mesh&, const _Result&, const Mesh::Point& p,
const int i) const;
32         private:
33             _Method * m_method;
34         };
35
36         template<class _Problem, class _Mesh, class _Result>
37         const int dg_solver<_Problem, _Mesh, _Result>::elliptic_solver(_Problem* problem, _Mesh* mesh,
_Result* result)
38         {
39             std::vector<double> res;
40             //std::shared_ptr<Algebra::MatrixSkyline> matrix{ new Algebra::MatrixSkyline() };
41             Algebra::MatrixSkyline* matrix{ new Algebra::MatrixSkyline() };
42             std::vector<double> rhs;
43             if (m_method != nullptr)
44                 delete m_method;
45             m_method = new _Method{ problem, mesh, matrix, &rhs };
46             m_method->Discretization();
47             Algebra::ESolver esl{ Algebra::Solvers::GMRES };
48             *result = esl.Solve(*matrix, rhs, *result, res, 100000, 1e-13);
49             delete matrix;
50             return 0;
51         }
52
53         template<class _Problem, class _Mesh, class _Result>
54         const double dg_solver<_Problem, _Mesh, _Result>::get_value(const _Mesh& mesh, const _Result&
res, const Mesh::Point& p) const
55         {
56             if (m_method != nullptr)
57                 return m_method->GetSolution(mesh, res, p);
58             return 0.;
59         }
60     }
61 }

```

```

59     }
60
61     template<class _Problem, class _Mesh, class _Result>
62     const Mesh::Point dg_solver<_Problem, _Mesh, _Result>::get_gradvalue(const _Mesh& mesh, const
        _Result& res, const Mesh::Point& p) const
63     {
64         if (m_method != nullptr)
65             return m_method->GetGradSolution(mesh, res, p);
66         return Mesh::Point(0, 0, 0);
67     }
68
69     template<class _Problem, class _Mesh, class _Result>
70     const double dg_solver<_Problem, _Mesh, _Result>::get_value(const _Method* method2, const _Mesh&
        mesh, const _Result& res, const Mesh::Point& p) const
71     {
72         if (method2 != nullptr)
73             return method2->GetSolution(mesh, res, p);
74         return 0.;
75     }
76
77     template<class _Problem, class _Mesh, class _Result>
78     const double dg_solver<_Problem, _Mesh, _Result>::get_value(const _Mesh& mesh, const _Result&
        res, const Mesh::Point& p, const int i) const
79     {
80         if (m_method != nullptr)
81             return m_method->GetSolution(mesh, res, p, i);
82         return 0.;
83     }
84     template<class _Problem, class _Mesh, class _Result>
85     const Mesh::Point dg_solver<_Problem, _Mesh, _Result>::get_gradvalue(const _Mesh& mesh, const
        _Result& res, const Mesh::Point& p, const int i) const
86     {
87         if (m_method != nullptr)
88             return m_method->GetGradSolution(mesh, res, p, i);
89         return Mesh::Point(0, 0, 0);
90     }
91 }
92 }
93 }
94 #endif // !CORENC_SOLVERS_dg_solver_H_

```

7.112 Solvers/dg_solver_shallow_water.cpp File Reference

```

#include "dg_solver_shallow_water.h"
#include <vector>
#include "../CoreNCFEM/Grids/RegularMesh.h"
#include "../CoreNCFEM/Parameter.h"
#include <algorithm>
#include <functional>

```

7.113 Solvers/dg_solver_shallow_water.h File Reference

```

#include <vector>
#include <functional>
#include <istream>
#include <iostream>
#include <fstream>
#include <algorithm>
#include "../CoreNCFEM/Point.h"

```

Classes

- struct `corenc::solvers::vector_solution`
- class `corenc::solvers::dg_solver_shallow_water`
- class `corenc::solvers::dg_shallow_water< Mesh >`

Namespaces

- namespace `corenc`
- namespace `corenc::solvers`

7.114 dg_solver_shallow_water.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_SOLVERS_DG_SOLVER_SHALLOW_WATER_H_
2 #define CORENC_SOLVERS_DG_SOLVER_SHALLOW_WATER_H_
3
4 #include <vector>
5 #include <functional>
6 #include <istream>
7 #include <iostream>
8 #include <fstream>
9 #include <algorithm>
10 #include "../CoreNCFEM/Point.h"
11 namespace corenc
12 {
13     namespace solvers
14     {
15         struct vector_solution
16         {
17             std::vector<double> S[3];
18             vector_solution() {}
19             vector_solution(const int _size)
20             {
21                 S[0].resize(_size);
22                 S[1].resize(_size);
23                 S[2].resize(_size);
24             }
25         };
26         class dg_solver_shallow_water
27         {
28         public:
29             dg_solver_shallow_water();
30             ~dg_solver_shallow_water();
31             const int solve() const;
32             const int solve(
33                 const double t0,
34                 const double t1,
35                 const size_t nx,
36                 const size_t ny,
37                 const double x0,
38                 const double x1,
39                 const double y0,
40                 const double y1,
41                 const double g,
42                 const double H,
43                 const std::function<const std::vector<double>>(const std::vector<double>&)>&,
44                 const std::function<const std::vector<double>>(const std::vector<double>&)>&,
45                 const std::function<const std::vector<double>>(const std::vector<double>&)>&) const;
46         };
47
48         template<class Mesh>
49         class dg_shallow_water
50         {
51         public:
52             dg_shallow_water();
53             ~dg_shallow_water();
54             const int solve(
55                 const double t0,
56                 const double t1,
57                 const Mesh& mesh,
58                 vector_solution& sol,
59                 const std::function<const std::vector<double>>(const std::vector<double>&)>&,
60                 const std::function<const std::vector<double>>(const std::vector<double>&)>&,
61                 const std::function<const std::vector<double>>(const std::vector<double>&)>&) const;
62             const int solve(
63                 const double t0,
64                 const double t1,
65                 const Mesh& mesh,
66                 vector_solution& sol,
67                 std::vector<double>& bath,
68                 std::vector<double>& ze,
69                 std::vector<double>& dzx,
70                 std::vector<double>& dzy,
71                 std::vector<double>& dbx,
72                 std::vector<double>& dby,

```

```

73         const std::function<const std::vector<double>(const std::vector<double>&, const int)>&,
74         const std::function<const std::vector<double>(const std::vector<double>&, const int)>&,
75         const std::function<const std::vector<double>(const std::vector<double>&, const int)>&,
76         const bool WRITE_FILE) const;
77     };
78
79     template<class Mesh>
80     dg_shallow_water<Mesh>::dg_shallow_water()
81     {
82     }
83
84     template<class Mesh>
85     dg_shallow_water<Mesh>::~dg_shallow_water()
86     {
87     }
88
89
90     template<class Mesh>
91     const int dg_shallow_water<Mesh>::solve(
92         const double t0,
93         const double t1,
94         const Mesh& mesh,
95         vector_solution& sol,
96         const std::function< const std::vector<double>(const std::vector<double>&)>&R,
97         const std::function< const std::vector<double>(const std::vector<double>&)>&G,
98         const std::function< const std::vector<double>(const std::vector<double>&)>&F) const
99     {
100         std::vector<double> Ut[3];
101         const int max_iter = 30000;
102         const double dx = mesh.GetNode(mesh.GetNumberOfNodes() - 1).x - mesh.GetNode(0).x;
103         const double dy = mesh.GetNode(mesh.GetNumberOfNodes() - 1).y - mesh.GetNode(0).y;
104         //const double dx = (x1 - x0) / nx;
105         //const double dy = (y1 - y0) / ny;
106         const int size = mesh.GetNumberOfElements();
107         const int bsize = mesh.GetNumberOfBoundaries();
108
109         std::vector<vector_solution> U(2);
110         std::vector<vector_solution> W(2);
111         U[0].S[0].resize(size);
112         U[0].S[1].resize(size);
113         U[0].S[2].resize(size);
114         U[1].S[0].resize(size);
115         U[1].S[1].resize(size);
116         U[1].S[2].resize(size);
117
118         W[0].S[0].resize(size);
119         W[0].S[1].resize(size);
120         W[0].S[2].resize(size);
121         W[1].S[0].resize(size);
122         W[1].S[1].resize(size);
123         W[1].S[2].resize(size);
124         for (size_t i = 0; i < size; ++i)
125         {
126             W[0].S[0][i] = sol.S[0][i];
127             W[0].S[1][i] = sol.S[1][i];
128             W[0].S[2][i] = sol.S[2][i];
129
130             U[0].S[0][i] = sol.S[0][i];
131             U[0].S[1][i] = sol.S[1][i] / sol.S[0][i];
132             U[0].S[2][i] = sol.S[2][i] / sol.S[0][i];
133         }
134
135         double t_step = 0.1;
136         const double cfl = 0.5;
137         // W = [h hu hv]
138         double lambda_x = 0;
139         double lambda_y = 0;
140         double lambda_dx = 0;
141         double lambda_dy = 0;
142         double lambda = 0;
143         double t_curr = 0;
144         double g = 1;
145         size_t iter_max = 10000;
146         for (size_t t = 0; t < iter_max && t_curr < t1; ++t, t_curr += t_step)
147         {
148             lambda_x = 0;
149             lambda_y = 0;
150             for (size_t i = 0; i < size; ++i)
151             {
152                 const auto& elem = mesh.GetElement(i);
153                 const auto& res = F(std::vector<double>{W[t].S[0][i], W[t].S[1][i], W[t].S[2][i]});
154                 W[t + 1].S[0][i] = W[t].S[0][i] + res[0];
155                 W[t + 1].S[1][i] = W[t].S[1][i] + res[1];
156                 W[t + 1].S[2][i] = W[t].S[2][i] + res[2];
157
158                 lambda_x = std::max(fabs(W[t].S[1][i]), lambda_x);
159                 lambda_y = std::max(fabs(W[t].S[2][i]), lambda_y);

```



```

160         }
161         t_step = cfl / 2 * std::min(dx / lambda_x, dy / lambda_y);
162         if (t_curr + t_step > t1)
163             t_step = t1 - t_curr;
164         for (size_t i = 0; i < bsize; ++i)
165         {
166             const auto& bound = mesh.GetBoundary(i);
167             const int nk = bound->GetNeighbour(0);
168             const int ne = bound->GetNeighbour(1);
169             const auto& normal = bound->GetNormal();
170             if (ne > -1)
171             {
172                 const auto& normal = bound->GetNormal();
173                 std::vector<double> wk(3);
174                 wk[0] = U[t].S[0][nk];
175                 wk[1] = U[t].S[1][nk] * U[t].S[0][nk];
176                 wk[2] = U[t].S[2][nk] * U[t].S[0][nk];
177
178                 std::vector<double> we(3);
179                 we[0] = U[t].S[0][ne];
180                 we[1] = U[t].S[1][ne] * U[t].S[0][ne];
181                 we[2] = U[t].S[2][ne] * U[t].S[0][ne];
182
183                 //lambda_x = std::max(fabs(U[t].S[1][nk]) + sqrt(g * U[t].S[0][nk]),
184                 fabs(U[t].S[1][ne]) + sqrt(g * U[t].S[0][ne]));
185                 //lambda_y = std::max(fabs(U[t].S[2][nk]) + sqrt(g * U[t].S[0][nk]),
186                 fabs(U[t].S[2][ne]) + sqrt(g * U[t].S[0][ne]));
187
188                 lambda_x = std::max(fabs(U[t].S[1][nk]), fabs(U[t].S[1][ne]));
189                 lambda_y = std::max(fabs(U[t].S[2][nk]), fabs(U[t].S[2][ne]));
190
191                 lambdax = std::max(lambdax, lambda_x);
192                 lambday = std::max(lambday, lambda_y);
193                 double ll = std::max(lambda_x, lambda_y);
194                 //cout << "max:\t" << ll << endl;
195                 std::vector<double> uk(3);
196                 uk[0] = U[t].S[0][nk];
197                 uk[1] = U[t].S[1][nk];
198                 uk[2] = U[t].S[2][nk];
199
200                 std::vector<double> ue(3);
201                 ue[0] = U[t].S[0][ne];
202                 ue[1] = U[t].S[1][ne];
203                 ue[2] = U[t].S[2][ne];
204
205                 const auto rk = R(uk);
206                 const auto re = R(ue);
207                 const auto gk = G(uk);
208                 const auto ge = G(ue);
209
210                 std::vector<double> uu(3);
211                 uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
212                 (normal.x * (rk[0] + re[0]) / 2 + normal.y * (gk[0] + ge[0]) / 2 - (lambda_x * normal.x / 2 * (ue[0]
213                 - uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
214                 uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
215                 (normal.x * (rk[1] + re[1]) / 2 + normal.y * (gk[1] + ge[1]) / 2 - (lambda_x * normal.x / 2 * (ue[1]
216                 - uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
217                 uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
218                 (normal.x * (rk[2] + re[2]) / 2 + normal.y * (gk[2] + ge[2]) / 2 - (lambda_x * normal.x / 2 * (ue[2]
219                 - uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
220                 W[t + 1].S[0][nk] -= uu[0];
221                 W[t + 1].S[1][nk] -= uu[1];
222                 W[t + 1].S[2][nk] -= uu[2];
223
224                 uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
225                 (-normal.x * (rk[0] + re[0]) / 2 - normal.y * (gk[0] + ge[0]) / 2 + (lambda_x * normal.x / 2 * (ue[0]
226                 - uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
227                 uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
228                 (-normal.x * (rk[1] + re[1]) / 2 - normal.y * (gk[1] + ge[1]) / 2 + (lambda_x * normal.x / 2 * (ue[1]
229                 - uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
230                 uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
231                 (-normal.x * (rk[2] + re[2]) / 2 - normal.y * (gk[2] + ge[2]) / 2 + (lambda_x * normal.x / 2 * (ue[2]
232                 - uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
233                 W[t + 1].S[0][ne] -= uu[0];
234                 W[t + 1].S[1][ne] -= uu[1];
235                 W[t + 1].S[2][ne] -= uu[2];
236             }
237         }
238         for (size_t i = 0; i < bsize; ++i)
239         {
240             const auto& bound = mesh.GetBoundary(i);
241             const int nk = bound->GetNeighbour(0);
242             const int ne = bound->GetNeighbour(1);
243             if (ne == -1)
244             {
245                 auto normal = bound->GetNormal();

```

```

233
234         std::vector<double> u(3);
235         u[0] = U[t].S[0][nk];
236         u[1] = -U[t].S[1][nk];
237         u[2] = -U[t].S[2][nk];
238
239         lambdax = std::max(lambdax, lambda_x);
240         lambday = std::max(lambday, lambda_y);
241
242         std::vector<double> uk(3);
243         uk[0] = U[t].S[0][nk];
244         uk[1] = U[t].S[1][nk];
245         uk[2] = U[t].S[2][nk];
246
247         std::vector<double> ue(3);
248         ue[0] = u[0];
249         ue[1] = u[1];
250         ue[2] = u[2];
251
252         const auto rk = R(uk);
253         const auto re = R(ue);
254         const auto gk = G(uk);
255         const auto ge = G(ue);
256         lambda_x = std::max(fabs(uk[1]), fabs(ue[1]));
257         lambda_y = std::max(fabs(uk[2]), fabs(ue[2]));
258         std::vector<double> uu(3);
259         if (normal.x > 0 || normal.y > 0)
260         {
261             uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
262             (normal.x * (rk[0] + re[0]) / 2 + normal.y * (gk[0] + ge[0]) / 2 - (lambda_x * normal.x / 2 * (ue[0]
263             - uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
264             uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
265             (normal.x * (rk[1] + re[1]) / 2 + normal.y * (gk[1] + ge[1]) / 2 - (lambda_x * normal.x / 2 * (ue[1]
266             - uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
267             uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
268             (normal.x * (rk[2] + re[2]) / 2 + normal.y * (gk[2] + ge[2]) / 2 - (lambda_x * normal.x / 2 * (ue[2]
269             - uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
270         }
271         else
272         {
273             uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
274             (normal.x * (rk[0] + re[0]) / 2 + normal.y * (gk[0] + ge[0]) / 2 + (lambda_x * normal.x / 2 * (ue[0]
275             - uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
276             uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
277             (normal.x * (rk[1] + re[1]) / 2 + normal.y * (gk[1] + ge[1]) / 2 + (lambda_x * normal.x / 2 * (ue[1]
278             - uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
279             uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
280             (normal.x * (rk[2] + re[2]) / 2 + normal.y * (gk[2] + ge[2]) / 2 + (lambda_x * normal.x / 2 * (ue[2]
281             - uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
282         }
283         W[t + 1].S[0][nk] -= uu[0];
284         W[t + 1].S[1][nk] -= uu[1];
285         W[t + 1].S[2][nk] -= uu[2];
286     }
287     }
288     for (size_t i = 0; i < size; ++i)
289     {
290         U[t + 1].S[0][i] = W[t + 1].S[0][i];
291         U[t + 1].S[1][i] = W[t + 1].S[1][i] / W[t + 1].S[0][i];
292         U[t + 1].S[2][i] = W[t + 1].S[2][i] / W[t + 1].S[0][i];
293     }
294     W.push_back(vector_solution(size));
295     U.push_back(vector_solution(size));
296 }
297
298 template<class Mesh>
299 const int dg_shallow_water<Mesh>::solve(
300     const double t0,
301     const double t1,
302     const Mesh& mesh,
303     vector_solution& sol,
304     std::vector<double>& bath,
305     std::vector<double>& ze,
306     std::vector<double>& dzx,
307     std::vector<double>& dzy,

```

```

308         std::vector<double>& dbx,
309         std::vector<double>& dby,
310         const std::function< const std::vector<double>(const std::vector<double>&, const int)>&R,
311         const std::function< const std::vector<double>(const std::vector<double>&, const int)>&G,
312         const std::function< const std::vector<double>(const std::vector<double>&, const int)>&F,
313         const bool WRITE_FILE) const
314     {
315         std::vector<double> Ut[3];
316         const int max_iter = 30000;
317         double dx = 100, dy = 100;
318         //const double dx = mesh.GetNode(mesh.GetNumberOfNodes() - 1).x - mesh.GetNode(0).x;
319         //const double dy = mesh.GetNode(mesh.GetNumberOfNodes() - 1).y - mesh.GetNode(0).y;
320         //const double dx = (x1 - x0) / nx;
321         //const double dy = (y1 - y0) / ny;
322         const int size = mesh.GetNumberOfElements();
323         const int bsize = mesh.GetNumberOfBoundaries();
324
325         std::vector<vector_solution> U(2);
326         std::vector<vector_solution> W(2);
327         U[0].S[0].resize(size);
328         U[0].S[1].resize(size);
329         U[0].S[2].resize(size);
330         U[1].S[0].resize(size);
331         U[1].S[1].resize(size);
332         U[1].S[2].resize(size);
333
334         W[0].S[0].resize(size);
335         W[0].S[1].resize(size);
336         W[0].S[2].resize(size);
337         W[1].S[0].resize(size);
338         W[1].S[1].resize(size);
339         W[1].S[2].resize(size);
340         for (size_t i = 0; i < size; ++i)
341         {
342             W[0].S[0][i] = sol.S[0][i];
343             W[0].S[1][i] = sol.S[1][i];
344             W[0].S[2][i] = sol.S[2][i];
345
346             U[0].S[0][i] = sol.S[0][i];
347             U[0].S[1][i] = sol.S[1][i] / sol.S[0][i];
348             U[0].S[2][i] = sol.S[2][i] / sol.S[0][i];
349         }
350         auto center = [=](const size_t i)
351         {
352             const auto& elem = mesh.GetElement(i);
353             std::vector<corenc::Mesh::Point> pts(4);
354             pts[0] = mesh.GetNode(elem->GetNode(0));
355             pts[1] = mesh.GetNode(elem->GetNode(1));
356             pts[2] = mesh.GetNode(elem->GetNode(2));
357             pts[3] = mesh.GetNode(elem->GetNode(3));
358             return corenc::Mesh::Point(pts[0].x + (pts[3].x - pts[0].x) / 2, pts[0].y + (pts[3].y -
359             pts[0].y) / 2);
360         };
361         double t_step = 0.1;
362         const double cfl = 0.1;
363         // W = [h hu hv]
364         double lambda_x = 0;
365         double lambda_y = 0;
366         double lambdax = 0;
367         double lambday = 0;
368         double lambda = 0;
369         double t_curr = 0;
370         double g = 1;
371         size_t iter_max = 100;
372         for (size_t t = 0; t < iter_max && t_curr < t1; ++t, t_curr += t_step)
373         {
374             lambda_x = 0;
375             lambda_y = 0;
376             for (size_t i = 0; i < size; ++i)
377             {
378                 const auto& elem = mesh.GetElement(i);
379                 const auto& res = F(std::vector<double>{W[t].S[0][i], W[t].S[1][i], W[t].S[2][i]},
380                 i);
381                 W[t + 1].S[0][i] = W[t].S[0][i] + res[0];
382                 W[t + 1].S[1][i] = W[t].S[1][i] + res[1];
383                 W[t + 1].S[2][i] = W[t].S[2][i] + res[2];
384
385                 lambda_x = std::max(fabs(U[t].S[1][i]) + sqrt(g*U[t].S[0][i]), lambda_x);
386                 lambda_y = std::max(fabs(U[t].S[2][i]) + sqrt(g*U[t].S[0][i]), lambda_y);
387                 dx = std::min(mesh.GetNode(elem->GetNode(3)).x - mesh.GetNode(elem->GetNode(0)).x,
388                 dx);
389                 dy = std::min(mesh.GetNode(elem->GetNode(3)).y - mesh.GetNode(elem->GetNode(0)).y,
390                 dy);
391
392                 //lambda_x = std::min(U[t].S[1][i])
393                 //lambda_x = std::max(fabs(U[t].S[1][i]), lambda_x);
394                 //lambda_y = std::max(fabs(U[t].S[2][i]), lambda_y);
395             }
396         }

```

```

391         t_step = cfl / 2 * std::min(dx / lambda_x, dy / lambda_y);
392         //std::cout << t_step << std::endl;
393         if (t_curr + t_step > t1)
394             t_step = t1 - t_curr;
395         for (size_t i = 0; i < bsize; ++i)
396         {
397             const auto& bound = mesh.GetBoundary(i);
398             const int nk = bound->GetNeighbour(0);
399             const int ne = bound->GetNeighbour(1);
400             const auto& normal = bound->GetNormal();
401             if (ne > -1)
402             {
403                 const auto& normal = bound->GetNormal();
404                 std::vector<double> wk(3);
405                 wk[0] = U[t].S[0][nk];
406                 wk[1] = U[t].S[1][nk] * U[t].S[0][nk];
407                 wk[2] = U[t].S[2][nk] * U[t].S[0][nk];
408
409                 std::vector<double> we(3);
410                 we[0] = U[t].S[0][ne];
411                 we[1] = U[t].S[1][ne] * U[t].S[0][ne];
412                 we[2] = U[t].S[2][ne] * U[t].S[0][ne];
413
414                 lambda_x = std::max(fabs(U[t].S[1][nk]) + sqrt(g * U[t].S[0][nk]),
fabs(U[t].S[1][ne]) + sqrt(g * U[t].S[0][ne]));
415                 lambda_y = std::max(fabs(U[t].S[2][nk]) + sqrt(g * U[t].S[0][nk]),
fabs(U[t].S[2][ne]) + sqrt(g * U[t].S[0][ne]));
416
417                 //lambda_x = std::max(fabs(U[t].S[1][nk]), fabs(U[t].S[1][ne]));
418                 //lambda_y = std::max(fabs(U[t].S[2][nk]), fabs(U[t].S[2][ne]));
419
420
421                 lambdax = std::max(lambdax, lambda_x);
422                 lambday = std::max(lambday, lambda_y);
423                 double ll = std::max(lambda_x, lambda_y);
424                 //cout << "max:\t" << ll << endl;
425                 std::vector<double> uk(3);
426                 uk[0] = U[t].S[0][nk];
427                 uk[1] = U[t].S[1][nk];
428                 uk[2] = U[t].S[2][nk];
429
430                 std::vector<double> ue(3);
431                 ue[0] = U[t].S[0][ne];
432                 ue[1] = U[t].S[1][ne];
433                 ue[2] = U[t].S[2][ne];
434
435                 const auto rk = R(uk, nk);
436                 const auto re = R(ue, ne);
437                 const auto gk = G(uk, nk);
438                 const auto ge = G(ue, ne);
439
440                 std::vector<double> uu(3);
441                 uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
(normal.x * (rk[0] + re[0]) / 2 + normal.y * (gk[0] + ge[0]) / 2 - (lambda_x * normal.x / 2 * (ue[0]
- uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
442                 uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
(normal.x * (rk[1] + re[1]) / 2 + normal.y * (gk[1] + ge[1]) / 2 - (lambda_x * normal.x / 2 * (ue[1]
- uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
443                 uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
(normal.x * (rk[2] + re[2]) / 2 + normal.y * (gk[2] + ge[2]) / 2 - (lambda_x * normal.x / 2 * (ue[2]
- uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
444                 W[t + 1].S[0][nk] -= uu[0];
445                 W[t + 1].S[1][nk] -= uu[1];
446                 W[t + 1].S[2][nk] -= uu[2];
447
448                 uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
(-normal.x * (rk[0] + re[0]) / 2 - normal.y * (gk[0] + ge[0]) / 2 + (lambda_x * normal.x / 2 * (ue[0]
- uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
449                 uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
(-normal.x * (rk[1] + re[1]) / 2 - normal.y * (gk[1] + ge[1]) / 2 + (lambda_x * normal.x / 2 * (ue[1]
- uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
450                 uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
(-normal.x * (rk[2] + re[2]) / 2 - normal.y * (gk[2] + ge[2]) / 2 + (lambda_x * normal.x / 2 * (ue[2]
- uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
451                 W[t + 1].S[0][ne] -= uu[0];
452                 W[t + 1].S[1][ne] -= uu[1];
453                 W[t + 1].S[2][ne] -= uu[2];
454
455             }
456         }
457     }
458     for (size_t i = 0; i < bsize; ++i)
459     {
460         const auto& bound = mesh.GetBoundary(i);
461         const int nk = bound->GetNeighbour(0);
462         const int ne = bound->GetNeighbour(1);
463         if (ne == -1)

```

```

464         {
465             auto normal = bound->GetNormal();
466
467             std::vector<double> u(3);
468             u[0] = U[t].S[0][nk];
469             u[1] = -U[t].S[1][nk];
470             u[2] = -U[t].S[2][nk];
471
472             //u[0] = U[t].S[0][nk];
473             //u[1] = U[t].S[1][nk];
474             //u[2] = U[t].S[2][nk];
475
476             lambdax = std::max(lambdax, lambda_x);
477             lambday = std::max(lambday, lambda_y);
478
479             std::vector<double> uk(3);
480             uk[0] = U[t].S[0][nk];
481             uk[1] = U[t].S[1][nk];
482             uk[2] = U[t].S[2][nk];
483
484             std::vector<double> ue(3);
485             ue[0] = u[0];
486             ue[1] = u[1];
487             ue[2] = u[2];
488
489             const auto rk = R(uk, nk);
490             const auto re = R(ue, nk);
491             const auto gk = G(uk, nk);
492             const auto ge = G(ue, nk);
493             lambda_x = std::max(fabs(uk[1]) + sqrt(g*uk[0]), fabs(ue[1]) + sqrt(g*ue[0]));
494             lambda_y = std::max(fabs(uk[2]) + sqrt(g*uk[0]), fabs(ue[2]) + sqrt(g*ue[0]));
495             std::vector<double> uu(3);
496             if (normal.x > 0 || normal.y > 0)
497             {
498                 uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
499                 (normal.x * (rk[0] + re[0]) / 2 + normal.y * (gk[0] + ge[0]) / 2 - (lambda_x * normal.x / 2 * (ue[0]
500                 - uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
501                 uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
502                 (normal.x * (rk[1] + re[1]) / 2 + normal.y * (gk[1] + ge[1]) / 2 - (lambda_x * normal.x / 2 * (ue[1]
503                 - uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
504                 uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
505                 (normal.x * (rk[2] + re[2]) / 2 + normal.y * (gk[2] + ge[2]) / 2 - (lambda_x * normal.x / 2 * (ue[2]
506                 - uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
507             }
508             else
509             {
510                 uu[0] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
511                 (normal.x * (rk[0] + re[0]) / 2 + normal.y * (gk[0] + ge[0]) / 2 + (lambda_x * normal.x / 2 * (ue[0]
512                 - uk[0]) + lambda_y * normal.y / 2 * (ue[0] - uk[0])));
513                 uu[1] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
514                 (normal.x * (rk[1] + re[1]) / 2 + normal.y * (gk[1] + ge[1]) / 2 + (lambda_x * normal.x / 2 * (ue[1]
515                 - uk[1]) + lambda_y * normal.y / 2 * (ue[1] - uk[1])));
516                 uu[2] = t_step / mesh.GetElement(nk)->GetMeasure() * bound->GetMeasure() *
517                 (normal.x * (rk[2] + re[2]) / 2 + normal.y * (gk[2] + ge[2]) / 2 + (lambda_x * normal.x / 2 * (ue[2]
518                 - uk[2]) + lambda_y * normal.y / 2 * (ue[2] - uk[2])));
519             }
520
521             W[t + 1].S[0][nk] -= uu[0];
522             W[t + 1].S[1][nk] -= uu[1];
523             W[t + 1].S[2][nk] -= uu[2];
524         }
525     }
526
527     for (size_t i = 0; i < size; ++i)
528     {
529         U[t + 1].S[0][i] = W[t + 1].S[0][i];
530         U[t + 1].S[1][i] = W[t + 1].S[1][i] / W[t + 1].S[0][i];
531         U[t + 1].S[2][i] = W[t + 1].S[2][i] / W[t + 1].S[0][i];
532     }
533     W.push_back(vector_solution(size));
534     U.push_back(vector_solution(size));
535
536     for (size_t k = 0; k < bsize; ++k)
537     {
538         const auto& bound = mesh.GetBoundary(k);
539         const auto nk = bound->GetNeighbour(0);
540         const auto ne = bound->GetNeighbour(1);
541         ze[nk] = W[t + 1].S[0][nk] - bath[nk];
542         if (ne > -1)
543         {
544             ze[ne] = W[t + 1].S[0][ne] - bath[ne];
545             const auto ce = center(ne);
546             const auto ck = center(nk);
547             const double cx = ce.x - ck.x;
548             const double cy = ce.y - ck.y;
549             if (fabs(cy) < 1e-13)
550             {

```

```

539             dzx[nk] = (ze[ne] - ze[nk]) / cx;
540             dzx[ne] = dzx[nk];
541             dbx[nk] = (bath[ne] - bath[nk]) / cx;
542             dbx[ne] = dbx[nk];
543         }
544         else
545         {
546             dzy[nk] = (ze[ne] - ze[nk]) / cy;
547             dzy[ne] = dzy[nk];
548             dby[nk] = (bath[ne] - bath[nk]) / cy;
549             dby[ne] = dby[nk];
550         }
551     }
552 }
553 }
554 /*const auto ut = W.size() - 2;
555 for (size_t i = 0; i < size; ++i)
556 {
557     sol.S[0][i] = W[ut].S[0][i];
558     sol.S[1][i] = W[ut].S[1][i];
559     sol.S[2][i] = W[ut].S[2][i];
560 }
561 std::ofstream ofs;
562 ofs.open("meshU.txt");
563 const size_t t_r = U.size() - 1;
564 ofs << t_r << std::endl;
565 for (size_t i = 0; i < t_r; ++i)
566     for (size_t j = 0; j < size; ++j)
567         ofs << U[i].S[0][j] - bath[j] << std::endl;
568 ofs.close();*/
569 return 0;
570 }
571 }
572 }
573
574
575 #endif // !CORENC_SOLVERS_DG_SOLVER_SHALLOW_WATER_H_

```

7.115 Solvers/eigen_solver.h File Reference

```

#include <vector>
#include <complex>

```

Classes

- class [corenc::solvers::eigen_solver< Matrix, Solver >](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::solvers](#)

7.116 eigen_solver.h

[Go to the documentation of this file.](#)

```

1 #ifndef EIGEN_SOLVER_H
2 #define EIGEN_SOLVER_H
3 #include <vector>
4 #include <complex>
5 namespace corenc
6 {
7     namespace solvers
8     {
9         template<class Matrix, class Solver>

```

```

10     class eigen_solver
11     {
12     public:
13         eigen_solver() {}
14         ~eigen_solver() {}
15         void rayleigh(Matrix* A, Matrix* B, Solver* esl, std::complex<double>* mu0, double*
x0, const int n) const
16         {
17             std::vector<std::complex<double>> x(n);
18             std::vector<std::complex<double>> y(n);
19             std::vector<std::complex<double>> lam(n);
20             double norm_mu = 0;
21             double norm_x = 0;
22             for (int i = 0; i < n; ++i)
23             {
24                 norm_mu += std::norm(mu0[i]) * std::norm(mu0[i]);
25                 norm_x += std::norm(x0[i]) * std::norm(x0[i]);
26             }
27             norm_mu = sqrt(norm_mu);
28             norm_x = sqrt(norm_x);
29             for (int i = 0; i < n; ++i)
30             {
31                 x[i] = x0[i] / norm_x;
32                 y[i] = mu0[i] / norm_mu;
33             }
34             std::complex<double> temp(0, 0);
35             temp =
36         }
37     };
38 }
39 }
40 #endif // EIGEN_SOLVER_H

```

7.117 Solvers/fem_solver.h File Reference

```

#include "../CoreNCFEM/Grids/TriangularMesh.h"
#include "../CoreNCFEM/Methods/FEMethod.h"
#include "../Problems/DiffusionScalar.h"
#include "../CoreNCA/MatrixSkyline.h"
#include "../CoreNCFEM/Methods/FEAnalysis.h"

```

Classes

- class [corenc::solvers::fem_solver<_Problem, _Mesh, _Result>](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::solvers](#)

7.118 fem_solver.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_SOLVERS_FEM_SOLVER_H_
2 #define CORENC_SOLVERS_FEM_SOLVER_H_
3
4 #include "../CoreNCFEM/Grids/TriangularMesh.h"
5 #include "../CoreNCFEM/Methods/FEMethod.h"
6 #include "../Problems/DiffusionScalar.h"
7 #include "../CoreNCA/MatrixSkyline.h"
8 #include "../CoreNCFEM/Methods/FEAnalysis.h"
9
10 // FINITE ELEMENT METHOD SOLVER ONLY IN SPACE

```

```

11
12 namespace corenc
13 {
14     namespace solvers
15     {
16         template<class _Problem, class _Mesh, class _Result>
17         class fem_solver
18         {
19             using _Method = method::FEMethod<_Problem, _Mesh, Algebra::MatrixSkyline>;
20             using _Method2 = method::FEMethod<_Problem, _Mesh, Algebra::Matrix>;
21         public:
22             fem_solver() :m_method2{ nullptr }, m_method{nullptr}{}
23             ~fem_solver()
24             {
25                 if(m_method2 != nullptr)
26                     delete m_method2;
27                 if(m_method != nullptr)
28                     delete m_method;
29             }
30             // terms, method, mesh, solver, result
31             const int      elliptic_solver(_Problem*, _Mesh*, _Result*);
32             const int      elliptic_solver_gauss(_Problem*, _Mesh*, _Result*);
33             const double   get_value(const _Mesh&, const _Result&, const Mesh::Point& p) const;
34             const double   get_value(const _Method2*, const _Mesh&, const _Result&, const
Mesh::Point& p) const;
35             const double   get_value(const _Method*, const _Mesh&, const _Result&, const
Mesh::Point& p) const;
36             const double   get_value(const _Mesh&, const _Result&, const Mesh::Point& p, const
int i) const;
37             const Mesh::Point get_gradvalue(const _Mesh&, const _Result&, const Mesh::Point& p)
const;
38             const Mesh::Point get_gradvalue(const _Mesh&, const _Result&, const Mesh::Point& p,
const int i) const;
39         private:
40             _Method*      m_method;
41             _Method*      m_method2;
42             //_Method2*    m_method2;
43         };
44
45         template<class _Problem, class _Mesh, class _Result>
46         const int fem_solver<_Problem, _Mesh, _Result>::elliptic_solver(_Problem* problem, _Mesh* mesh,
_Result* result)
47         {
48             std::vector<double> res;
49             std::vector<double> res2;
50             //std::shared_ptr<Algebra::MatrixSkyline> matrix{ new Algebra::MatrixSkyline() };
51             Algebra::MatrixSkyline* matrix{ new Algebra::MatrixSkyline() };
52             std::vector<double> rhs;
53             if (m_method != nullptr)
54                 delete m_method;
55
56             m_method = new _Method{ problem, mesh, matrix, &rhs };
57
58             m_method->Discretization();
59
60             Algebra::ESolver esl{ Algebra::Solvers::BiCGStab };
61             //std::cout << "Size:\t" << matrix->GetSize() << std::endl;
62
63             //std::cout << matrix->GetSize() << std::endl;
64             *result = esl.Solve(*matrix, rhs, *result, res, 100000, 1e-13);
65             //std::cout << matrix->GetSize() << std::endl;
66
67             //esl.Pardiso(*matrix, rhs, *result);
68             //res.resize(matrix2->GetSize());
69             //for (int i = 0; i < matrix2->GetSize(); ++i)
70             //{
71                 //for (int j = 0; j < matrix2->GetSize(); ++j)
72                 //{
73                     //res[i] += result->operator[](j) * (matrix2->GetElement(i, j));
74                 //}
75             //}
76             delete matrix;
77
78             return 0;
79         }
80
81         template<class _Problem, class _Mesh, class _Result>
82         const int fem_solver<_Problem, _Mesh, _Result>::elliptic_solver_gauss(_Problem* problem, _Mesh*
mesh, _Result* result)
83         {
84             std::vector<double> res;
85             std::vector<double> res2;
86             //std::shared_ptr<Algebra::MatrixSkyline> matrix{ new Algebra::MatrixSkyline() };
87             //Algebra::MatrixSkyline* matrix{ new Algebra::MatrixSkyline() };
88             Algebra::Matrix* matrix2{ new Algebra::Matrix() };
89             std::vector<double> rhs;
90             //if (m_method != nullptr)

```



```

91         // delete m_method;
92         if (m_method2 != nullptr)
93             delete m_method2;
94
95         //m_method = new _Method{ problem, mesh, matrix, &rhs };
96         m_method2 = new _Method2{ problem, mesh, matrix2, &rhs };
97         //m_method->Discretization();
98         m_method2->Discretization();
99         //Algebra::ESolver esl{ Algebra::Solvers::BiCGStab };
100        //std::cout << "Size:\t" << matrix->GetSize() << std::endl;
101        Algebra::ESolver esl{ Algebra::Solvers::Gauss };
102        //std::cout << matrix->GetSize() << std::endl;
103        //*result = esl.Solve(*matrix, rhs, *result, res, 100000, 1e-13);
104        //std::cout << matrix->GetSize() << std::endl;
105        esl.Gauss(*matrix2, rhs, *result);
106        //esl.Pardiso(*matrix, rhs, *result);
107        //res.resize(matrix2->GetSize());
108        //for (int i = 0; i < matrix2->GetSize(); ++i)
109        //{
110            //for (int j = 0; j < matrix2->GetSize(); ++j)
111            //{
112                //res[i] += result->operator[] (j) * (matrix2->GetElement(i, j));
113            //}
114        //}
115        //delete matrix;
116        delete matrix2;
117        return 0;
118    }
119    template<class _Problem, class _Mesh, class _Result>
120    const double fem_solver<_Problem, _Mesh, _Result>::get_value(const _Mesh& mesh, const _Result&
121    res, const Mesh::Point& p) const
122    {
123        if (m_method2 != nullptr)
124            return m_method2->GetSolution(mesh, res, p);
125        return 0.;
126    }
127    template<class _Problem, class _Mesh, class _Result>
128    const double fem_solver<_Problem, _Mesh, _Result>::get_value(const _Method2* method2, const
129    _Mesh& mesh, const _Result& res, const Mesh::Point& p) const
130    {
131        if (method2 != nullptr)
132            return method2->GetSolution(mesh, res, p);
133        return 0.;
134    }
135    template<class _Problem, class _Mesh, class _Result>
136    const double fem_solver<_Problem, _Mesh, _Result>::get_value(const _Method* method2, const
137    _Mesh& mesh, const _Result& res, const Mesh::Point& p) const
138    {
139        if (method2 != nullptr)
140            return method2->GetSolution(mesh, res, p);
141        return 0.;
142    }
143    template<class _Problem, class _Mesh, class _Result>
144    const double fem_solver<_Problem, _Mesh, _Result>::get_value(const _Mesh& mesh, const _Result&
145    res, const Mesh::Point& p, const int i) const
146    {
147        if (m_method2 != nullptr)
148            return m_method2->GetSolution(mesh, res, p, i);
149        return 0.;
150    }
151    template<class _Problem, class _Mesh, class _Result>
152    const Mesh::Point fem_solver<_Problem, _Mesh, _Result>::get_gradvalue(const _Mesh& mesh, const
153    _Result& res, const Mesh::Point& p) const
154    {
155        if (m_method2 != nullptr)
156            return m_method2->GetGradSolution(mesh, res, p);
157        return Mesh::Point(0, 0, 0);
158    }
159    template<class _Problem, class _Mesh, class _Result>
160    const Mesh::Point fem_solver<_Problem, _Mesh, _Result>::get_gradvalue(const _Mesh& mesh, const
161    _Result& res, const Mesh::Point& p, const int i) const
162    {
163        if (m_method2 != nullptr)
164            return m_method2->GetGradSolution(mesh, res, p, i);
165        return Mesh::Point(0, 0, 0);
166    }
167 }
168 #endif // !CORENC_SOLVERS_FEM_SOLVER_H_

```

7.119 Solvers/fem_solver_lib.h File Reference

```
#include "../CoreNCFEM/Grids/TriangularMesh.h"
#include "../CoreNCFEM/Methods/FEMethod.h"
#include "../Problems/DiffusionScalar.h"
#include "../CoreNCA/MatrixSkyline.h"
#include "../CoreNCFEM/Methods/FEAnalysis.h"
#include <chrono>
#include <iostream>
#include <fstream>
#include <eigen3/Eigen/SparseCore>
#include <cstdlib>
#include <string>
#include <eigen3/Eigen/Cholesky>
#include <eigen3/Eigen/Jacobi>
#include <eigen3/Eigen/Householder>
#include <eigen3/Eigen/IterativeLinearSolvers>
#include <eigen3/unsupported/Eigen/IterativeSolvers>
#include <eigen3/Eigen/LU>
#include <eigen3/unsupported/Eigen/SparseExtra>
#include <eigen3/Eigen/SparseLU>
#include <eigen3/Eigen/UmfPackSupport>
```

Classes

- class `corenc::solvers::fem_solver_lib<_Problem, _Mesh, _Result>`

Namespaces

- namespace `corenc`
- namespace `corenc::solvers`

7.120 fem_solver_lib.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_SOLVERS_fem_solver_lib_H_
2 #define CORENC_SOLVERS_fem_solver_lib_H_
3
4 #include "../CoreNCFEM/Grids/TriangularMesh.h"
5 #include "../CoreNCFEM/Methods/FEMethod.h"
6 #include "../Problems/DiffusionScalar.h"
7 #include "../CoreNCA/MatrixSkyline.h"
8 #include "../CoreNCFEM/Methods/FEAnalysis.h"
9 #include <chrono>
10
11 /**#include <eigen3/Eigen/SparseCore>
12 //#include <eigen3/Eigen/Sparse>
13 //#include <eigen3/Eigen/SparseLU>
14 //#include <eigen3/Eigen/SparseCholesky>
15 #include <eigen3/Eigen/Cholesky>
16 //#include <eigen3/Eigen/Dense>
17 #include <eigen3/Eigen/UmfPackSupport>
18 //#include <eigen3/Eigen/SparseCore>
19 #include <eigen3/unsupported/Eigen/SparseExtra>
20 #include <eigen3/Eigen/SparseLU>*/
21
22 #include <iostream>
23 #include <fstream>
24 #include <eigen3/Eigen/SparseCore>
```

```

25 #include <cstdlib>
26 #include <string>
27 #include <eigen3/Eigen/Cholesky>
28 #include <eigen3/Eigen/Jacobi>
29 #include <eigen3/Eigen/Householder>
30 #include <eigen3/Eigen/IterativeLinearSolvers>
31 #include <eigen3/unsupported/Eigen/IterativeSolvers>
32 #include <eigen3/Eigen/LU>
33 #include <eigen3/unsupported/Eigen/SparseExtra>
34 #include <eigen3/Eigen/SparseLU>
35 #include <eigen3/Eigen/UmfPackSupport>
36
37
38 // FINITE ELEMENT METHOD SOLVER ONLY IN SPACE
39
40 namespace corenc
41 {
42     namespace solvers
43     {
44         template<class _Problem, class _Mesh, class _Result>
45         class fem_solver_lib
46         {
47             using _Method = method::FEMethod<_Problem, _Mesh, Algebra::MatrixSkyline>;
48             using _Method2 = method::FEMethod<_Problem, _Mesh, Algebra::Matrix>;
49         public:
50             fem_solver_lib() :m_method2{ nullptr }, m_method{nullptr}{}
51             ~fem_solver_lib()
52             {
53                 if(m_method2 != nullptr)
54                     delete m_method2;
55                 if(m_method != nullptr)
56                     delete m_method;
57             }
58             // terms, method, mesh, solver, result
59             const int          elliptic_solver(_Problem*, _Mesh*, _Result*);
60             const int          elliptic_solver_gauss(_Problem*, _Mesh*, _Result*);
61             const double        get_value(const _Mesh&, const _Result&, const Mesh::Point& p) const;
62             const double        get_value(const _Method2*, const _Mesh&, const _Result&, const
Mesh::Point& p) const;
63             const double        get_value(const _Method*, const _Mesh&, const _Result&, const
Mesh::Point& p) const;
64             const double        get_value(const _Mesh&, const _Result&, const Mesh::Point& p, const
int i) const;
65             const Mesh::Point   get_gradvalue(const _Mesh&, const _Result&, const Mesh::Point& p)
const;
66             const Mesh::Point   get_gradvalue(const _Mesh&, const _Result&, const Mesh::Point& p,
const int i) const;
67         private:
68             _Method*            m_method;
69             _Method*            m_method2;
70             //_Method2*          m_method2;
71         };
72
73         template<class _Problem, class _Mesh, class _Result>
74         const int fem_solver_lib<_Problem, _Mesh, _Result>::elliptic_solver(_Problem* problem, _Mesh*
mesh, _Result* result)
75         {
76
77
78             std::vector<double> res;
79             std::vector<double> res2;
80             //std::shared_ptr<Algebra::MatrixSkyline> matrix{ new Algebra::MatrixSkyline() };
81             Algebra::MatrixSkyline* matrix{ new Algebra::MatrixSkyline() };
82
83
84
85             std::vector<double> rhs;
86             if (m_method != nullptr)
87                 delete m_method;
88
89             m_method = new _Method{ problem, mesh, matrix, &rhs };
90
91             m_method->Discretization();
92
93             int n = matrix->GetSize();
94             Eigen::SparseMatrix<double> eA(n, n);
95             for (int i = 0; i < n; ++i)
96             {
97                 for (int j = 0; j < n; ++j)
98                 {
99                     auto elem = matrix->GetElement(i, j);
100                     if (fabs(elem) > 1e-12)
101                         eA.insert(i, j) = elem;
102                 }
103             }
104             eA.makeCompressed();
105

```

```

106         //Algebra::ESolver esl{ Algebra::Solvers::BiCGStab };
107
108
109         //std::cout << "Size:\t" << matrix->GetSize() << std::endl;
110
111         //std::cout << matrix->GetSize() << std::endl;
112         //*result = esl.Solve(*matrix, rhs, *result, res, 100000, 1e-13);
113         Eigen::MatrixMarketIterator<double> it("matr");
114         Eigen::VectorXd xx(n);
115         for (int i = 0; i < n; ++i)
116             xx[i] = rhs[i];
117         std::chrono::steady_clock::time_point beg{ std::chrono::steady_clock::now() };
118
119         //Eigen::SparseLU<Eigen::SparseMatrix<double>> chol; // (eA);
120         //Eigen::BiCGSTAB<Eigen::SparseMatrix<double>> chol; // (eA);
121
122         Eigen::UmfPackLU<Eigen::SparseMatrix<double>> chol; // (eA);
123
124         //chol.analyzePattern(eA);
125         chol.compute(eA);
126         //chol.factorize(eA);
127         if (chol.info() != Eigen::Success)
128             std::cout << "oops" << std::endl;
129         Eigen::Matrix<double, Eigen::Dynamic, 1> bb;
130         //auto bb = chol.solve(xx);
131         bb = chol.solve(xx);
132         if (chol.info() != Eigen::Success)
133             std::cout << "oops xx" << std::endl;
134
135         //Eigen::saveMarket(eA, "matrix.mtx");
136         //Eigen::saveMarketVector(xx, "vector.mtx");
137         //Eigen::saveMarketVector(bb, "MatrixName_x.mtx");
138
139
140         std::chrono::steady_clock::time_point end{ std::chrono::steady_clock::now() };
141         auto dur = std::chrono::duration_cast<std::chrono::milliseconds>(end - beg).count();
142         std::cout << dur << std::endl;
143
144         result->resize(n);
145         for (int i = 0; i < n; ++i)
146             (*result)[i] = bb[i];
147
148         //std::cout << matrix->GetSize() << std::endl;
149
150         //esl.Pardiso(*matrix, rhs, *result);
151         //res.resize(matrix2->GetSize());
152         //for (int i = 0; i < matrix2->GetSize(); ++i)
153         //{
154             //for (int j = 0; j < matrix2->GetSize(); ++j)
155             //{
156                 //res[i] += result->operator[] (j) * (matrix2->GetElement(i, j));
157             //}
158         //}
159         delete matrix;
160
161         return 0;
162     }
163
164     template<class _Problem, class _Mesh, class _Result>
165     const int fem_solver_lib<_Problem, _Mesh, _Result>::elliptic_solver_gauss(_Problem* problem,
166     _Mesh* mesh, _Result* result)
167     {
168         std::vector<double> res;
169         std::vector<double> res2;
170         //std::shared_ptr<Algebra::MatrixSkyline> matrix{ new Algebra::MatrixSkyline() };
171         //Algebra::MatrixSkyline* matrix{ new Algebra::MatrixSkyline() };
172         Algebra::Matrix* matrix2{ new Algebra::Matrix() };
173         std::vector<double> rhs;
174         //if (m_method != nullptr)
175         //    delete m_method;
176         if (m_method2 != nullptr)
177             delete m_method2;
178
179         //m_method = new _Method{ problem, mesh, matrix, &rhs };
180         m_method2 = new _Method2{ problem, mesh, matrix2, &rhs };
181         //m_method->Discretization();
182         m_method2->Discretization();
183         //Algebra::ESolver esl{ Algebra::Solvers::BiCGStab };
184         //std::cout << "Size:\t" << matrix->GetSize() << std::endl;
185         Algebra::ESolver esl{ Algebra::Solvers::Gauss };
186         //std::cout << matrix->GetSize() << std::endl;
187         //*result = esl.Solve(*matrix, rhs, *result, res, 100000, 1e-13);
188         //std::cout << matrix->GetSize() << std::endl;
189         esl.Gauss(*matrix2, rhs, *result);
190         //esl.Pardiso(*matrix, rhs, *result);
191         //res.resize(matrix2->GetSize());
192         //for (int i = 0; i < matrix2->GetSize(); ++i)

```

```

192         //{
193         //for (int j = 0; j < matrix2->GetSize(); ++j)
194         //{
195             //res[i] += result->operator[] (j) * (matrix2->GetElement(i, j));
196         //}
197         //}
198         //delete matrix;
199         delete matrix2;
200         return 0;
201     }
202     template<class _Problem, class _Mesh, class _Result>
203     const double fem_solver_lib<_Problem, _Mesh, _Result>::get_value(const _Mesh& mesh, const
204     _Result& res, const Mesh::Point& p) const
205     {
206         if (m_method2 != nullptr)
207             return m_method2->GetSolution(mesh, res, p);
208         return 0.;
209     }
210     template<class _Problem, class _Mesh, class _Result>
211     const double fem_solver_lib<_Problem, _Mesh, _Result>::get_value(const _Method2* method2, const
212     _Mesh& mesh, const _Result& res, const Mesh::Point& p) const
213     {
214         if (method2 != nullptr)
215             return method2->GetSolution(mesh, res, p);
216         return 0.;
217     }
218     template<class _Problem, class _Mesh, class _Result>
219     const double fem_solver_lib<_Problem, _Mesh, _Result>::get_value(const _Method* method2, const
220     _Mesh& mesh, const _Result& res, const Mesh::Point& p) const
221     {
222         if (method2 != nullptr)
223             return method2->GetSolution(mesh, res, p);
224         return 0.;
225     }
226     template<class _Problem, class _Mesh, class _Result>
227     const double fem_solver_lib<_Problem, _Mesh, _Result>::get_value(const _Mesh& mesh, const
228     _Result& res, const Mesh::Point& p, const int i) const
229     {
230         if (m_method2 != nullptr)
231             return m_method2->GetSolution(mesh, res, p, i);
232         return 0.;
233     }
234     template<class _Problem, class _Mesh, class _Result>
235     const Mesh::Point fem_solver_lib<_Problem, _Mesh, _Result>::get_gradvalue(const _Mesh& mesh,
236     const _Result& res, const Mesh::Point& p) const
237     {
238         if (m_method2 != nullptr)
239             return m_method2->GetGradSolution(mesh, res, p);
240         return Mesh::Point(0, 0, 0);
241     }
242     template<class _Problem, class _Mesh, class _Result>
243     const Mesh::Point fem_solver_lib<_Problem, _Mesh, _Result>::get_gradvalue(const _Mesh& mesh,
244     const _Result& res, const Mesh::Point& p, const int i) const
245     {
246         if (m_method2 != nullptr)
247             return m_method2->GetGradSolution(mesh, res, p, i);
248         return Mesh::Point(0, 0, 0);
249     }
250 }
251 #endif // !CORENC_SOLVERS_fem_solver_lib_H_

```

7.121 Tests/FiniteElements/test_case_rectanglebasis.cpp File Reference

```

#include "test_case_rectanglebasis.h"
#include "../CoreNCFEM/FiniteElements/Rectangle.h"

```

7.122 Tests/FiniteElements/test_case_rectanglebasis.h File Reference

Classes

- class [corenc::tests::test_case_rectanglebasis](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::tests](#)

Macros

- `#define` [CORENC_TEST_CASE_RECTANGLEBASIS_H_](#)

7.122.1 Macro Definition Documentation

7.122.1.1 CORENC_TEST_CASE_RECTANGLEBASIS_H_

```
#define CORENC_TEST_CASE_RECTANGLEBASIS_H_
```

7.123 test_case_rectanglebasis.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef CORENC_TEST_CASE_RECTANGLEBASIS_H_
3 #define CORENC_TEST_CASE_RECTANGLEBASIS_H_
4 namespace corenc
5 {
6     namespace tests
7     {
8         class test_case_rectanglebasis
9         {
10         public:
11             test_case_rectanglebasis();
12             ~test_case_rectanglebasis();
13             const int      mass_matrix() const;
14             const int      stress_matrix() const;
15         };
16     }
17 }
18 #endif // !CORENC_TEST_CASE_RECTANGLEBASIS_H_
```

7.124 Tests/FiniteElements/test_case_trianglebasis.cpp File Reference

```
#include "test_case_trianglebasis.h"
#include "../../CoreNCFEM/FiniteElements/Triangle.h"
```

7.125 Tests/FiniteElements/test_case_trianglebasis.h File Reference

Classes

- class [corenc::tests::test_case_trianglebasis](#)

Namespaces

- namespace [corenc](#)
- namespace [corenc::tests](#)

Macros

- `#define` [CORENC_TEST_CASE_TRIANGLEBASIS_H_](#)

7.125.1 Macro Definition Documentation

7.125.1.1 CORENC_TEST_CASE_TRIANGLEBASIS_H_

```
#define CORENC_TEST_CASE_TRIANGLEBASIS_H_
```

7.126 test_case_trianglebasis.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef CORENC_TEST_CASE_TRIANGLEBASIS_H_
3 #define CORENC_TEST_CASE_TRIANGLEBASIS_H_
4 namespace corenc
5 {
6     namespace tests
7     {
8         class test_case_trianglebasis
9         {
10         public:
11             test_case_trianglebasis();
12             ~test_case_trianglebasis();
13             const int      mass_matrix() const;
14             const int      stress_matrix() const;
15         };
16     }
17 }
18 #endif // !CORENC_TEST_CASE_TRIANGLEBASIS_H_
```

7.127 Tests/test_case_elliptic_fem.cpp File Reference

```
#include "test_case_elliptic_fem.h"
#include "../CoreNCFEM/Grids/TriangularMesh.h"
#include "../CoreNCFEM/Grids/RegularMesh.h"
#include "../CoreNCFEM/Methods/FEMethod.h"
#include "../Problems/DiffusionScalar.h"
#include "../CoreNCA/MatrixSkyline.h"
#include "../CoreNCFEM/Methods/FEAnalysis.h"
#include "../Solvers/fem_solver.h"
#include "../CoreNCFEM/GaussianField.h"
#include "../CoreNCFEM/FiniteElements/Triangle.h"
#include <random>
#include <math.h>
```

Macros

- `#define` [_USE_MATH_DEFINES](#)

Functions

- `const double` [kekus](#) (`const double c`, `const double a=0`, `const double b=90.`)

7.127.1 Macro Definition Documentation

7.127.1.1 [_USE_MATH_DEFINES](#)

```
#define _USE_MATH_DEFINES
```

7.127.2 Function Documentation

7.127.2.1 [kekus\(\)](#)

```
const double kekus (  
    const double c,  
    const double a = 0,  
    const double b = 90. )
```

7.128 Tests/test_case_elliptic_fem.h File Reference

Classes

- class [corenc::test_case_elliptic_fem](#)

Namespaces

- namespace [corenc](#)

7.129 test_case_elliptic_fem.h

[Go to the documentation of this file.](#)

```

1 #ifndef CORENC_TEST_CASE_ELLIPTIC_FEM_H_
2 #define CORENC_TEST_CASE_ELLIPTIC_FEM_H_
3
4 // SOME TEST PROBLEMS FOR ELLIPTIC CASE WITH FEM && DG\
5 // 0th, 1st, 2nd order definitely maybe more high-order
6 // LAGRANGE && HIERARCHICAL BASIS FUNCTIONS
7 // LATER MAYBE EVEN TESTS WITH MULTISCALE
8
9 namespace corenc
10 {
11     class test_case_elliptic_fem
12     {
13     public:
14         test_case_elliptic_fem();
15         ~test_case_elliptic_fem();
16         //const int          test_case_elliptic_fem_3d_tetra() const;
17         const int          elliptic_fem_2d_tria() const;
18         const int          elliptic_fem_solver() const;
19         const int          elliptic_fem_square_lin_basis() const;
20         const int          elliptic_fem_hp_fixed(const int h_ref_max, const int p_ref_max)
21             const;
22         const int          elliptic_fem_hp_fixed_triangle(const int h_ref_max, const int
23             p_ref_max) const;
24         const int          elliptic_fem_hp_lagrange_triangle(const int h_ref_max, const int
25             p_ref_max) const;
26         const int          elliptic_fem_hxhy_fixed_triangle(const int hx_max, const int hy_max)
27             const;
28         const int          conv_diff_fem_fixed_triangle(const int h_ref_max, const int
29             p_ref_max) const;
30         const int          global_matrix(const int h_ref_max, const int p_ref_max) const;
31         //const int          test_case_elliptic_fem_square_2nd_basis() const;
32         //const int          test_case_elliptic_fem_square_nth_basis() const;
33         const int          elliptic_2layer_fem_2d_tria_h() const;
34         const int          elliptic_fem_2d_rect_source() const;
35         const int          elliptic_gaussian_triangle() const;
36         const int          mass_matrix_3rd_order() const;
37         const int          stress_matrix_3rd_order() const;
38         const int          mass_matrix_4th_order() const;
39         const int          stress_matrix_4th_order() const;
40         const int          homotopy_conv_diff_fem(const double step) const;
41         //const int          test_case_elliptic_fem_2d_rect() const;
42         //const int          test_case_elliptic_fem_3d_hex() const;
43         //const int          test_case_elliptic_dg_3d_tetra() const;
44         //const int          test_case_elliptic_dg_2d_tria() const;
45         //const int          test_case_elliptic_dg_2d_rect() const;
46         //const int          test_case_elliptic_dg_3d_hex() const;
47     };
48 }
49
50 #endif // !CORENC_TEST_CASE_ELLIPTIC_FEM_H_

```

7.130 Tests/test_case_regular_mesh.cpp File Reference

```

#include "test_case_regular_mesh.h"
#include "../CoreNCFEM/Grids/RegularMesh.h"

```

7.131 Tests/test_case_regular_mesh.h File Reference

Classes

- class `corenc::tests::test_case_regular_mesh`

Namespaces

- namespace `corenc`
- namespace `corenc::tests`

Macros

- [#define CORENC_TEST_CASE_REGULAR_MESH_H_](#)

7.131.1 Macro Definition Documentation

7.131.1.1 CORENC_TEST_CASE_REGULAR_MESH_H_

```
#define CORENC_TEST_CASE_REGULAR_MESH_H_
```

7.132 test_case_regular_mesh.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef CORENC_TEST_CASE_REGULAR_MESH_H_
3 #define CORENC_TEST_CASE_REGULAR_MESH_H_
4
5 namespace corenc
6 {
7     namespace tests
8     {
9         class test_case_regular_mesh
10        {
11        public:
12            test_case_regular_mesh();
13            ~test_case_regular_mesh();
14            const int          construct_mesh() const;
15        };
16    }
17 }
18
19 #endif // !CORENC_TEST_CASE_REGULAR_MESH_H_
```

7.133 Tests/test_case_solver.cpp File Reference

```
#include "test_case_solver.h"
#include "../CoreNCFEM/Grids/TriangularMesh.h"
#include "../CoreNCFEM/Grids/RegularMesh.h"
#include "../CoreNCFEM/Methods/FEMethod.h"
#include "../Problems/DiffusionScalar.h"
#include "../CoreNCA/MatrixSkyline.h"
#include "../CoreNCFEM/Methods/FEAnalysis.h"
#include "../Solvers/fem_solver.h"
#include "../CoreNCFEM/GaussianField.h"
#include <random>
#include <math.h>
```

Macros

- [#define _USE_MATH_DEFINES](#)

Functions

- const int [solver](#) (const [Algebra::Matrix](#) &matrix, double *x, double *res)

7.133.1 Macro Definition Documentation

7.133.1.1 `_USE_MATH_DEFINES`

```
#define _USE_MATH_DEFINES
```

7.133.2 Function Documentation

7.133.2.1 `solver()`

```
const int solver (
    const Algebra::Matrix & matrix,
    double * x,
    double * res )
```

7.134 Tests/test_case_solver.h File Reference

Classes

- class [corenc::test_case_solver](#)

Namespaces

- namespace [corenc](#)

7.135 test_case_solver.h

[Go to the documentation of this file.](#)

```
1 #ifndef CORENC_TEST_CASE_SOLVER_H_
2 #define CORENC_TEST_CASE_SOLVER_H_
3
4 // SOME TEST PROBLEMS FOR ELLIPTIC CASE WITH FEM && DG\
5 // 0th, 1st, 2nd order definitely maybe more high-order
6 // LAGRANGE && HIERARHICAL BASIS FUNCTIONS
7 // LATER MAYBE EVEN TESTS WITH MULTISCALE
8
9 namespace corenc
10 {
11     class test_case_solver
12     {
13     public:
14         test_case_solver();
15         ~test_case_solver();
16         const int          gauss_solver() const;
17     };
18 }
19
20 #endif // !CORENC_TEST_CASE_SOLVER_H_
```

7.136 Tests/test_cases.cpp File Reference

```
#include "test_cases.h"
#include "test_case_elliptic_fem.h"
#include "test_case_solver.h"
#include "test_case_regular_mesh.h"
#include "FiniteElements/test_case_rectanglebasis.h"
#include "FiniteElements/test_case_trianglebasis.h"
#include <iostream>
#include <thread>
#include <future>
#include <chrono>
#include <ostream>
#include "../colors.h"
#include "test_conv_diff.h"
```

7.137 Tests/test_cases.h File Reference

```
#include <functional>
#include <ostream>
```

Classes

- class [corenc::test_cases](#)

Namespaces

- namespace [corenc](#)

Macros

- [#define CORENC_TEST_CASES_H_](#)

7.137.1 Macro Definition Documentation

7.137.1.1 CORENC_TEST_CASES_H_

```
#define CORENC_TEST_CASES_H_
```

7.138 test_cases.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #ifndef CORENC_TEST_CASES_H_
3 #define CORENC_TEST_CASES_H_
4 #include <functional>
5 #include <ostream>
6 namespace corenc
7 {
8     class test_cases
9     {
10     public:
11         test_cases();
12         ~test_cases();
13         const int perform() const;
14         const int perform(const std::function<const int()>&) const;
15         const int perform(const std::function<const int(std::ostream&)>&, std::ostream&) const;
16     };
17 }
18
19
20 #endif // !CORENC_TEST_CASES_H_
```

7.139 Tests/test_conv_diff.cpp File Reference

```
#include "test_conv_diff.h"
#include "../CoreNCFEM/Grids/TriangularMesh.h"
#include "../CoreNCFEM/Grids/RegularMesh.h"
#include "../CoreNCFEM/Methods/FEMethod.h"
#include "../Problems/DiffusionScalar.h"
#include "../CoreNCA/MatrixSkyline.h"
#include "../CoreNCFEM/Methods/FEAnalysis.h"
#include "../Solvers/fem_solver.h"
#include "../Solvers/fem_solver_lib.h"
#include "../CoreNCFEM/GaussianField.h"
#include "../CoreNCFEM/FiniteElements/Triangle.h"
#include <random>
#include <math.h>
```

Macros

- `#define _USE_MATH_DEFINES`

7.139.1 Macro Definition Documentation

7.139.1.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

7.140 Tests/test_conv_diff.h File Reference

Classes

- class `corenc::test_conv_diff`

Namespaces

- namespace `corenc`

7.141 test_conv_diff.h

[Go to the documentation of this file.](#)

```
1 #ifndef TEST_CONV_DIFF_H
2 #define TEST_CONV_DIFF_H
3
4 namespace corenc
5 {
6     class test_conv_diff
7     {
8     public:
9         test_conv_diff(){};
10        ~test_conv_diff(){};
11        void conv_diff_fem(const int h_ref_max, const int p_ref_max = 1) const;
12        void conv_diff_eigen(const int h_ref_max, const int p_ref_max = 1) const;
13    };
14 }
15
16 #endif // TEST_CONV_DIFF_H
```

Index

- `_NOPE_`
 - `MatrixSkyline.cpp`, [336](#)
- `_USE_MATH_DEFINES`
 - `test_case_elliptic_fem.cpp`, [518](#)
 - `test_case_solver.cpp`, [521](#)
 - `test_conv_diff.cpp`, [523](#)
- `_centrs`
 - `corenc::GaussianKernel`, [285](#)
- `~CBurgersScalar`
 - `corenc::CBurgersScalar`, [25](#)
- `~CCube`
 - `corenc::Mesh::CCube`, [29](#)
- `~CCubeBasis`
 - `corenc::Mesh::CCubeBasis`, [34](#)
- `~CDGMethod`
 - `corenc::method::CDGMethod< Type >`, [36](#)
- `~CDGMethodZero`
 - `corenc::method::CDGMethodZero< Type >`, [38](#)
- `~CDiffusionScalar`
 - `corenc::CDiffusionScalar`, [40](#)
- `~CEdge`
 - `corenc::Mesh::CEdge`, [46](#)
- `~CEdge2ndBasis`
 - `corenc::Mesh::CEdge2ndBasis`, [50](#)
- `~CEdgeConstantBasis`
 - `corenc::Mesh::CEdgeConstantBasis`, [53](#)
- `~CEdgeHermiteBasis`
 - `corenc::Mesh::CEdgeHermiteBasis`, [56](#)
- `~CEdgeLinearBasis`
 - `corenc::Mesh::CEdgeLinearBasis`, [59](#)
- `~CEdgeMultiBasis`
 - `corenc::Mesh::CEdgeMultiBasis`, [62](#)
- `~CElement`
 - `corenc::Mesh::CElement< bool >`, [79](#)
 - `corenc::Mesh::CElement< T >`, [64](#)
- `~CElement2D`
 - `corenc::Mesh::CElement2D< bool >`, [75](#)
 - `corenc::Mesh::CElement2D< T >`, [70](#)
- `~CFEMethod`
 - `corenc::method::CFEMethod< Type >`, [82](#)
- `~CFEMethodZero`
 - `corenc::method::CFEMethodZero< Type >`, [84](#)
- `~CFESolution`
 - `corenc::CFESolution`, [86](#)
- `~CFEweights`
 - `corenc::CFEweights`, [89](#)
- `~CFiniteElement`
 - `corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >`, [109](#)
- `corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >`, [117](#)
- `corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >`, [92](#)
- `~CFiniteElement2D`
 - `corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >`, [101](#)
- `~CFiniteSolver`
 - `corenc::CFiniteSolver< Method, Mesh, Solver >`, [124](#)
- `~CMesh`
 - `corenc::Mesh::CMesh< bool >`, [135](#)
 - `corenc::Mesh::CMesh< T >`, [125](#)
- `~CMesh1D`
 - `corenc::Mesh::CMesh1D`, [130](#)
- `~CNode`
 - `corenc::Mesh::CNode`, [139](#)
- `~CNodeBasis`
 - `corenc::Mesh::CNodeBasis`, [143](#)
- `~CParameter`
 - `corenc::Mesh::CParameter`, [145](#)
- `~CProblem`
 - `corenc::CProblem`, [147](#)
- `~CRectangle`
 - `corenc::Mesh::CRectangle`, [150](#)
- `~CRectangleBasis`
 - `corenc::Mesh::CRectangleBasis`, [155](#)
- `~CRectangleBasis2`
 - `corenc::Mesh::CRectangleBasis2`, [158](#)
- `~CRectangleBasis2x`
 - `corenc::Mesh::CRectangleBasis2x`, [162](#)
- `~CRectangleBasis2y`
 - `corenc::Mesh::CRectangleBasis2y`, [165](#)
- `~CRectangleConstantBasis`
 - `corenc::Mesh::CRectangleConstantBasis`, [168](#)
- `~CRectangleHBasis`
 - `corenc::Mesh::CRectangleHBasis`, [172](#)
- `~CRegularMesh`
 - `corenc::Mesh::CRegularMesh`, [176](#)
- `~CRegularMesh3D`
 - `corenc::Mesh::CRegularMesh3D`, [182](#)
- `~CSMethod`
 - `Methods::CSMethod`, [198](#)
- `~CShallowWater`
 - `corenc::CShallowWater`, [188](#)
- `~CShape`
 - `corenc::Mesh::CShape`, [192](#)
- `~CShapeFunction`
 - `corenc::Mesh::CShapeFunction< Type >`, [196](#)

- ~CSolution
 - corenc::CSolution, 199
- ~CTriangle
 - corenc::Mesh::CTriangle, 201
- ~CTriangleBasis
 - corenc::Mesh::CTriangleBasis, 206, 207
- ~CTriangleLagrangeBasis
 - corenc::Mesh::CTriangleLagrangeBasis, 211
- ~CTriangleLinear
 - corenc::Mesh::CTriangleLinear, 215
- ~CTriangleLinearBasis
 - corenc::Mesh::CTriangleLinearBasis, 219
- ~CTriangularMesh
 - corenc::Mesh::CTriangularMesh, 223
- ~CTriangularMeshLinear
 - corenc::Mesh::CTriangularMeshLinear, 229
- ~CVecSolution
 - corenc::CVecSolution, 234
- ~DGMethod
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 241
- ~DGMethodZero
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 248
- ~DGSolution
 - corenc::method::DGSolution< Grid >, 253
- ~ESolver
 - Algebra::ESolver, 257
- ~FEAnalysis
 - corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >, 261
- ~FEMethod
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 269
- ~FEMethodZero
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 276
- ~FVMethod1d
 - corenc::method::FVMethod1d, 281
- ~Matrix
 - Algebra::Matrix, 296
- ~MatrixDiag
 - Algebra::MatrixDiag, 299
- ~MatrixSkyline
 - Algebra::MatrixSkyline, 301
- ~RungeKutta
 - corenc::method::RungeKutta< Problem, Type >, 315
- ~STSolution
 - corenc::method::STSolution< Grid >, 317
- ~dg_shallow_water
 - corenc::solvers::dg_shallow_water< Mesh >, 235
- ~dg_solver
 - corenc::solvers::dg_solver< _Problem, _Mesh, _Result >, 236
- ~dg_solver_shallow_water
 - corenc::solvers::dg_solver_shallow_water, 238
- ~eigen_solver
 - corenc::solvers::eigen_solver< Matrix, Solver >, 255
- ~fem_solver
 - corenc::solvers::fem_solver< _Problem, _Mesh, _Result >, 262
- ~fem_solver_lib
 - corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >, 265
- ~multi_vector
 - corenc::multi_vector< T >, 304
- ~parameter
 - corenc::Mesh::parameter< T >, 308
- ~system_dg_method
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, 319
- ~test_case_elliptic_fem
 - corenc::test_case_elliptic_fem, 323
- ~test_case_rectanglebasis
 - corenc::tests::test_case_rectanglebasis, 326
- ~test_case_regular_mesh
 - corenc::tests::test_case_regular_mesh, 327
- ~test_case_solver
 - corenc::test_case_solver, 328
- ~test_case_trianglebasis
 - corenc::tests::test_case_trianglebasis, 329
- ~test_cases
 - corenc::test_cases, 330
- ~test_conv_diff
 - corenc::test_conv_diff, 331
- A
 - corenc::GaussianProcess, 287
- a
 - corenc::GaussianProcess, 287
- add_boundary_parameter
 - corenc::CDiffusionScalar, 40
 - corenc::CShallowWater, 188
- add_parameter
 - corenc::CDiffusionScalar, 40, 41
 - corenc::CShallowWater, 188
- AddElement
 - Algebra::Matrix, 296
 - Algebra::MatrixDiag, 299
 - Algebra::MatrixSkyline, 301
- addTerm
 - corenc::CBurgersScalar, 26
 - corenc::CDiffusionScalar, 41
 - corenc::CProblem, 147
 - corenc::CShallowWater, 189
- addTimeLayer
 - corenc::method::STSolution< Grid >, 317
- ADVECTION
 - corenc, 15
- Algebra, 13
 - BiCGStab, 13
 - Gauss, 13
 - GMRES, 13
 - GMRES_BiCGStab, 13
 - PARDISO, 13

- Solvers, 13
- Algebra::ESolver, 255
 - ~ESolver, 257
 - BiCGStab, 257
 - BiCGStabPrecond, 257
 - ESolver, 256
 - Gauss, 257, 258
 - GetSolution, 258
 - GMRES, 259
 - MatrixprodVector, 259
 - Pardiso, 260
 - Reload, 260
 - Solve, 260
- Algebra::Matrix, 295
 - ~Matrix, 296
 - AddElement, 296
 - Create, 296
 - GetElement, 297
 - GetSize, 297
 - Matrix, 295, 296
 - NullMatrix, 297
 - NullRow, 297
 - operator(), 297
 - operator=, 297
- Algebra::MatrixDiag, 298
 - ~MatrixDiag, 299
 - AddElement, 299
 - Create, 299
 - GetSize, 299
 - MatrixDiag, 298, 299
 - NullMatrix, 299
 - NullRow, 300
 - operator(), 300
 - operator=, 300
- Algebra::MatrixSkyline, 300
 - ~MatrixSkyline, 301
 - AddElement, 301
 - Create, 302
 - diff_skymatrix, 302
 - GetElement, 302
 - GetSize, 302
 - MatrixSkyline, 301
 - NullMatrix, 302
 - NullRow, 303
 - operator(), 303
 - operator=, 303
 - transpose_sky, 303
- Assemble
 - corenc::method::CDGMethod< Type >, 36
 - corenc::method::CDGMethodZero< Type >, 38
 - corenc::method::CFEMethod< Type >, 83
 - corenc::method::CFEMethodZero< Type >, 84
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, 319
- B
 - corenc::GaussianProcess, 287
- b
 - corenc::GaussianProcess, 287
- BBLACK
 - corenc::color, 16
- BBLUE
 - corenc::color, 16
- BCYAN
 - corenc::color, 16
- BGREEN
 - corenc::color, 16
- BiCGStab
 - Algebra, 13
 - Algebra::ESolver, 257
- BiCGStabPrecond
 - Algebra::ESolver, 257
- BLACK
 - corenc::color, 17
- BLUE
 - corenc::color, 17
- BMAGENTA
 - corenc::color, 17
- BoundaryType
 - corenc::method, 21
- BRED
 - corenc::color, 17
- BWHITE
 - corenc::color, 17
- BYELLOW
 - corenc::color, 17
- c
 - corenc::GaussianProcess, 287
- CBurgersScalar
 - corenc::CBurgersScalar, 25
- CCube
 - corenc::Mesh::CCube, 28, 29
- CCubeBasis
 - corenc::Mesh::CCubeBasis, 33
- CDGMethod
 - corenc::method::CDGMethod< Type >, 36
- CDGMethodZero
 - corenc::method::CDGMethodZero< Type >, 37
- CDiffusionScalar
 - corenc::CDiffusionScalar, 40
- CEdge
 - corenc::Mesh::CEdge, 46
- CEdge2ndBasis
 - corenc::Mesh::CEdge2ndBasis, 49, 50
- CEdgeConstantBasis
 - corenc::Mesh::CEdgeConstantBasis, 52, 53
- CEdgeHermiteBasis
 - corenc::Mesh::CEdgeHermiteBasis, 55, 56
- CEdgeLinearBasis
 - corenc::Mesh::CEdgeLinearBasis, 58, 59
- CEdgeMultiBasis
 - corenc::Mesh::CEdgeMultiBasis, 61, 62
- CElement
 - corenc::Mesh::CElement< bool >, 79
 - corenc::Mesh::CElement< T >, 64
- CElement2D
 - corenc::Mesh::CElement2D< bool >, 75

- corenc::Mesh::CElement2D< T >, 69
- center_point
 - Triangle.cpp, 379
 - wtf, 23
- Central
 - corenc::method, 22
- CFEMethod
 - corenc::method::CFEMethod< Type >, 82
- CFEMethodZero
 - corenc::method::CFEMethodZero< Type >, 84
- CFESolution
 - corenc::CFESolution, 86
- CFEweights
 - corenc::CFEweights, 89
- CFiniteElement
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 108, 109
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 116, 117
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 91, 92
- CFiniteElement2D
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 99–101
- CFiniteSolver
 - corenc::CFiniteSolver< Method, Mesh, Solver >, 123
- cfunc
 - corenc::Mesh::parameter< T >, 307
- cfunc_old
 - corenc::Mesh::parameter< T >, 307
- changeFlux
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, 319
- Clone
 - corenc::Mesh::CElement< bool >, 79
 - corenc::Mesh::CElement< T >, 65
 - corenc::Mesh::CElement2D< bool >, 75
 - corenc::Mesh::CElement2D< T >, 70
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 109
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 117
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 92
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 101
 - corenc::Mesh::CRegularMesh, 176
 - corenc::Mesh::CRegularMesh3D, 183
 - corenc::Mesh::CTriangularMesh, 223
- CMesh
 - corenc::Mesh::CMesh< bool >, 135
 - corenc::Mesh::CMesh< T >, 125
- CMesh1D
 - corenc::Mesh::CMesh1D, 129, 130
- CNode
 - corenc::Mesh::CNode, 139
- CNodeBasis
 - corenc::Mesh::CNodeBasis, 142
- colors.h, 333
- construct_mesh
 - corenc::tests::test_case_regular_mesh, 327
- conv_diff_eigen
 - corenc::test_conv_diff, 331
- conv_diff_fem
 - corenc::test_conv_diff, 331
- conv_diff_fem_fixed_triangle
 - corenc::test_case_elliptic_fem, 323
- corenc, 14
 - ADVECTION, 15
 - DIFFUSION, 15
 - EDUDV, 15
 - EDUV, 15
 - EFV, 15
 - EUDV, 15
 - EUV, 15
 - IDUDV, 15
 - IDUV, 15
 - IUDV, 15
 - IUV, 15
 - MASS, 15
 - Parameters, 15
 - RUV, 15
 - scalar_func, 15
 - SUPG, 15
 - Terms, 15
 - vector_func, 15
- corenc::CBurgersScalar, 25
 - ~CBurgersScalar, 25
 - addTerm, 26
 - CBurgersScalar, 25
 - getFlux, 26
 - getNumberOfTerms, 26
 - getTerm, 26
 - load_parameters, 26
 - removeTerm, 26
 - setTerm, 27
- corenc::CDiffusionScalar, 39
 - ~CDiffusionScalar, 40
 - add_boundary_parameter, 40
 - add_parameter, 40, 41
 - addTerm, 41
 - CDiffusionScalar, 40
 - findTerm, 41
 - get_boundary_parameter, 41, 42
 - get_boundary_type, 42
 - get_number_of_boundaries, 42
 - get_parameter, 42, 43
 - get_point_source, 43
 - get_total_sources, 43
 - getNumberOfTerms, 43
 - getTerm, 43
 - load_parameters, 44
 - removeTerm, 44
 - set_boundary_parameter, 44
 - set_parameter, 44

- set_point_source, 44
- setTerm, 45
- corenc::CFESolution, 85
 - ~CFESolution, 86
 - CFESolution, 86
 - operator double, 86
 - operator!=, 86
 - operator*, 88
 - operator*=: 87
 - operator+, 88
 - operator+=, 87
 - operator-, 88
 - operator-=, 87
 - operator/, 88
 - operator/=: 87
 - operator=, 87
 - operator==, 87
- corenc::CFEweights, 89
 - ~CFEweights, 89
 - CFEweights, 89
 - getWeight, 89
 - updateWeight, 89
- corenc::CFiniteSolver< Method, Mesh, Solver >, 123
 - ~CFiniteSolver, 124
 - CFiniteSolver, 123
 - Solve, 124
- corenc::color, 16
 - BBLACK, 16
 - BBLUE, 16
 - BCYAN, 16
 - BGREEN, 16
 - BLACK, 17
 - BLUE, 17
 - BMAGENTA, 17
 - BRED, 17
 - BWHITE, 17
 - BYELLOW, 17
 - CYAN, 17
 - ESCAPE, 17
 - GREEN, 18
 - MAGENTA, 18
 - PURPLE, 18
 - RED, 18
 - WHITE, 18
 - YELLOW, 18
- corenc::CProblem, 147
 - ~CProblem, 147
 - addTerm, 147
 - CProblem, 147
 - getNumberOfTerms, 147
 - getTerm, 148
 - load_parameters, 148
 - setTerm, 148
- corenc::CShallowWater, 187
 - ~CShallowWater, 188
 - add_boundary_parameter, 188
 - add_parameter, 188
 - addTerm, 189
 - CShallowWater, 188
 - get_boundary_parameter, 189
 - get_boundary_type, 189
 - get_number_of_boundaries, 189
 - get_parameter, 189, 190
 - get_solution, 190
 - getNumberOfTerms, 190
 - getTerm, 190
 - load_parameters, 190
 - removeTerm, 191
 - set_boundary_parameter, 191
 - set_parameter, 191
 - setTerm, 191
- corenc::CSolution, 199
 - ~CSolution, 199
 - CSolution, 199
- corenc::CVecSolution, 233
 - ~CVecSolution, 234
 - CVecSolution, 234
 - m_w, 234
- corenc::GaussianKernel, 284
 - _centrs, 285
 - GaussianKernel, 284
 - get_gp, 285
 - gpexp, 285
 - gpstep, 285
 - N, 285
- corenc::GaussianProcess, 286
 - A, 287
 - a, 287
 - B, 287
 - b, 287
 - c, 287
 - GaussianProcess, 286
 - He, 286
 - K, 287
 - l, 287
 - lambda, 288
 - phi, 286
 - sigma2, 288
- corenc::Mesh, 19
 - Cube, 20
 - Elements, 20
 - FIRST, 21
 - function_dp, 20
 - Interval, 20
 - LAST, 21
 - Mesh1D, 20
 - Meshes, 20
 - NODES, 21
 - Rectangle, 20
 - TetrahedralMesh, 20
 - Tetrahedron, 20
 - Triangle, 20
 - TriangularMesh, 20
- corenc::Mesh::CCube, 27
 - ~CCube, 29
 - CCube, 28, 29

- GetEdge, 29
- GetFacet, 29
- GetNode, 29, 30
- GetNumberOfEdges, 30
- GetNumberOfFacets, 30
- GetNumberOfNodes, 30
- IncreaseOrder, 30
- Integrate, 30, 31
- operator>>, 31
- operator=, 31
- operator==, 31
- SetEdge, 31
- SetFacet, 32
- SetNode, 32
- SetOrder, 32
- corenc::Mesh::CCubeBasis, 32
 - ~CCubeBasis, 34
 - CCubeBasis, 33
 - GetGradShapeFunction, 34
 - GetMeasure, 34
 - GetNormal, 34
 - GetNumberOfShapeFunctions, 34
 - GetShapeFunction, 34
 - GetValue, 35
 - GetWeight, 35
 - IncreaseOrder, 35
 - operator=, 35
 - ReverseNormal, 35
- corenc::Mesh::CEdge, 45
 - ~CEdge, 46
 - CEdge, 46
 - GetNode, 46, 47
 - GetNumberOfNodes, 47
 - IncreaseOrder, 47
 - Integrate, 47
 - operator>>, 48
 - operator=, 48
 - operator==, 48
 - SetNode, 48
- corenc::Mesh::CEdge2ndBasis, 49
 - ~CEdge2ndBasis, 50
 - CEdge2ndBasis, 49, 50
 - GetGradShapeFunction, 50
 - GetMeasure, 50
 - GetNormal, 50
 - GetNumberOfShapeFunctions, 50
 - GetShapeFunction, 51
 - GetWeight, 51
 - IncreaseOrder, 51
 - operator=, 51
 - ReverseNormal, 51
- corenc::Mesh::CEdgeConstantBasis, 52
 - ~CEdgeConstantBasis, 53
 - CEdgeConstantBasis, 52, 53
 - GetGradShapeFunction, 53
 - GetMeasure, 53
 - GetNormal, 53
 - GetNumberOfShapeFunctions, 53
- GetShapeFunction, 54
- GetWeight, 54
- IncreaseOrder, 54
- operator=, 54
- ReverseNormal, 54
- corenc::Mesh::CEdgeHermiteBasis, 55
 - ~CEdgeHermiteBasis, 56
 - CEdgeHermiteBasis, 55, 56
 - GetGradShapeFunction, 56
 - GetMeasure, 56
 - GetNormal, 56
 - GetNumberOfShapeFunctions, 56
 - GetShapeFunction, 57
 - GetWeight, 57
 - IncreaseOrder, 57
 - operator=, 57
 - ReverseNormal, 57
- corenc::Mesh::CEdgeLinearBasis, 58
 - ~CEdgeLinearBasis, 59
 - CEdgeLinearBasis, 58, 59
 - GetGradShapeFunction, 59
 - GetMeasure, 59
 - GetNormal, 59
 - GetNumberOfShapeFunctions, 59
 - GetShapeFunction, 60
 - GetWeight, 60
 - IncreaseOrder, 60
 - operator=, 60
 - ReverseNormal, 60
- corenc::Mesh::CEdgeMultiBasis, 61
 - ~CEdgeMultiBasis, 62
 - CEdgeMultiBasis, 61, 62
 - GetGradShapeFunction, 62
 - GetMeasure, 62
 - GetNormal, 62
 - GetNumberOfShapeFunctions, 62
 - GetShapeFunction, 63
 - GetWeight, 63
 - IncreaseOrder, 63
 - operator=, 63
 - ReverseNormal, 63
- corenc::Mesh::CElement< bool >, 78
 - ~CElement, 79
 - CElement, 79
 - Clone, 79
 - GetDoFs, 79
 - GetGradShapeFunction, 79
 - GetMeasure, 79
 - GetNeighbour, 79
 - GetNode, 80
 - GetNormal, 80
 - GetNumberOfNodes, 80
 - GetShapeFunction, 80
 - GetType, 80
 - GetWeight, 80
 - IncreaseOrder, 80
 - Integrate, 81
 - ReverseNormal, 81

- SetNeighbour, [81](#)
- SetNode, [81](#)
- SetType, [82](#)
- corenc::Mesh::CElement< T >, [64](#)
 - ~CElement, [64](#)
 - CElement, [64](#)
 - Clone, [65](#)
 - GetDoFs, [65](#)
 - GetGradShapeFunction, [65](#)
 - GetMeasure, [65](#)
 - GetNeighbour, [65](#)
 - GetNode, [66](#)
 - GetNormal, [66](#)
 - GetNumberOfNodes, [66](#)
 - GetShapeFunction, [66](#)
 - GetType, [66](#)
 - GetWeight, [67](#)
 - IncreaseOrder, [67](#)
 - Integrate, [67](#)
 - ReverseNormal, [68](#)
 - SetNeighbour, [68](#)
 - SetNode, [68](#)
 - SetType, [68](#)
- corenc::Mesh::CElement2D< bool >, [74](#)
 - ~CElement2D, [75](#)
 - CElement2D, [75](#)
 - Clone, [75](#)
 - GetDoFs, [75](#)
 - GetGradShapeFunction, [75](#)
 - GetMeasure, [75](#)
 - GetNeighbour, [75](#)
 - GetNode, [76](#)
 - GetNormal, [76](#)
 - GetNumberOfNodes, [76](#)
 - GetShapeFunction, [76](#)
 - GetType, [76](#)
 - GetWeight, [76](#)
 - IncreaseOrder, [76](#)
 - Integrate, [77](#)
 - ReverseNormal, [77](#)
 - SetNeighbour, [77](#)
 - SetNode, [77](#)
 - SetOrder, [78](#)
 - SetType, [78](#)
- corenc::Mesh::CElement2D< T >, [69](#)
 - ~CElement2D, [70](#)
 - CElement2D, [69](#)
 - Clone, [70](#)
 - GetDoFs, [70](#)
 - GetGradShapeFunction, [70](#)
 - GetMeasure, [70](#)
 - GetNeighbour, [71](#)
 - GetNode, [71](#)
 - GetNormal, [71](#)
 - GetNumberOfNodes, [71](#)
 - GetShapeFunction, [71](#)
 - GetType, [72](#)
 - GetWeight, [72](#)
 - IncreaseOrder, [72](#)
 - Integrate, [72, 73](#)
 - ReverseNormal, [73](#)
 - SetNeighbour, [73](#)
 - SetNode, [73](#)
 - SetOrder, [73](#)
 - SetType, [74](#)
- corenc::Mesh::CFiniteElement< Shape, ShapeFunction, bool, bool >, [107](#)
 - ~CFiniteElement, [109](#)
 - CFiniteElement, [108, 109](#)
 - Clone, [109](#)
 - GetDoFs, [110](#)
 - GetGradShapeFunction, [110](#)
 - GetMeasure, [110](#)
 - GetNeighbour, [110](#)
 - GetNode, [110](#)
 - GetNormal, [111](#)
 - GetNumberOfNodes, [111](#)
 - GetShape, [111](#)
 - GetShapeFunction, [111](#)
 - GetShapeFunctions, [111](#)
 - GetType, [112](#)
 - GetWeight, [112](#)
 - IncreaseOrder, [112](#)
 - Integrate, [112, 113](#)
 - operator>>, [114](#)
 - operator=, [113](#)
 - operator==, [114](#)
 - ReverseNormal, [113](#)
 - SetNeighbour, [113](#)
 - SetNode, [113](#)
 - SetShape, [114](#)
 - SetShapeFunction, [114](#)
 - SetType, [114](#)
- corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, bool >, [115](#)
 - ~CFiniteElement, [117](#)
 - CFiniteElement, [116, 117](#)
 - Clone, [117](#)
 - GetDoF, [118](#)
 - GetDoFs, [118](#)
 - GetGradShapeFunction, [118](#)
 - GetMeasure, [118](#)
 - GetNeighbour, [118](#)
 - GetNode, [119](#)
 - GetNormal, [119](#)
 - GetNumberOfNodes, [119](#)
 - GetShape, [119](#)
 - GetShapeFunction, [119](#)
 - GetShapeFunctions, [120](#)
 - GetType, [120](#)
 - GetWeight, [120](#)
 - IncreaseOrder, [120](#)
 - Integrate, [120, 121](#)
 - operator>>, [123](#)
 - operator=, [121](#)
 - operator==, [123](#)

- ReverseNormal, [121](#)
- SetDoF, [121](#)
- SetNeighbour, [122](#)
- SetNode, [122](#)
- SetShape, [122](#)
- SetShapeFunction, [122](#)
- SetType, [122](#)
- corenc::Mesh::CFiniteElement< Shape, ShapeFunction, DoF, T >, [90](#)
 - ~CFiniteElement, [92](#)
 - CFiniteElement, [91](#), [92](#)
 - Clone, [92](#)
 - GetDoF, [93](#)
 - GetDoFs, [93](#)
 - GetGradShapeFunction, [93](#)
 - GetMeasure, [93](#)
 - GetNeighbour, [93](#)
 - GetNode, [94](#)
 - GetNormal, [94](#)
 - GetNumberOfNodes, [94](#)
 - GetShape, [94](#)
 - GetShapeFunction, [94](#)
 - GetShapeFunctions, [95](#)
 - GetType, [95](#)
 - GetWeight, [95](#)
 - IncreaseOrder, [95](#)
 - Integrate, [95](#), [96](#)
 - operator>>, [98](#)
 - operator=, [96](#)
 - operator==, [98](#)
 - ReverseNormal, [96](#)
 - SetDoF, [96](#)
 - SetNeighbour, [97](#)
 - SetNode, [97](#)
 - SetShape, [97](#)
 - SetShapeFunction, [97](#)
 - SetType, [97](#)
- corenc::Mesh::CFiniteElement2D< Shape, ShapeFunction >, [98](#)
 - ~CFiniteElement2D, [101](#)
 - CFiniteElement2D, [99](#)–[101](#)
 - Clone, [101](#)
 - GetDoFs, [101](#)
 - GetGradShapeFunction, [102](#)
 - GetMeasure, [102](#)
 - GetNeighbour, [102](#)
 - GetNode, [102](#)
 - GetNormal, [102](#)
 - GetNumberOfNodes, [103](#)
 - GetShape, [103](#)
 - GetShapeFunction, [103](#)
 - GetShapeFunctions, [103](#)
 - GetType, [103](#)
 - GetWeight, [103](#)
 - IncreaseOrder, [104](#)
 - Integrate, [104](#)
 - operator>>, [106](#)
 - operator=, [104](#)
- operator==, [106](#)
- ReverseNormal, [105](#)
- SetNeighbour, [105](#)
- SetNode, [105](#)
- SetOrder, [105](#)
- SetShape, [105](#)
- SetShapeFunction, [106](#)
- SetType, [106](#)
- corenc::Mesh::CMesh< bool >, [134](#)
 - ~CMesh, [135](#)
 - CMesh, [135](#)
 - FindElement, [135](#)
 - GetBoundary, [135](#)
 - GetElement, [136](#)
 - getMinSize, [136](#)
 - GetNode, [136](#)
 - GetNumberOfBoundaries, [136](#)
 - GetNumberOfElements, [136](#)
 - GetNumberOfNodes, [136](#)
 - getParameter, [136](#), [137](#)
 - getSolution, [137](#)
 - setParameter, [137](#)
 - updateSolution, [137](#), [138](#)
- corenc::Mesh::CMesh< T >, [124](#)
 - ~CMesh, [125](#)
 - CMesh, [125](#)
 - FindElement, [125](#)
 - GetBoundary, [125](#)
 - GetElement, [125](#)
 - getMinSize, [125](#)
 - GetNode, [126](#)
 - GetNumberOfBoundaries, [126](#)
 - GetNumberOfElements, [126](#)
 - GetNumberOfNodes, [126](#)
 - getParameter, [126](#), [127](#)
 - getSolution, [127](#)
 - setParameter, [127](#)
 - updateSolution, [127](#), [128](#)
- corenc::Mesh::CMesh1D, [129](#)
 - ~CMesh1D, [130](#)
 - CMesh1D, [129](#), [130](#)
 - FindElement, [131](#)
 - GetBoundary, [131](#)
 - GetElement, [131](#)
 - GetElements, [131](#)
 - getMinSize, [131](#)
 - GetNode, [132](#)
 - GetNumberOfBoundaries, [132](#)
 - GetNumberOfElements, [132](#)
 - GetNumberOfNodes, [132](#)
 - getParameter, [132](#)
 - getSolution, [133](#)
 - operator=, [133](#)
 - setParameter, [133](#)
 - updateSolution, [133](#), [134](#)
- corenc::Mesh::CNode, [138](#)
 - ~CNode, [139](#)
 - CNode, [139](#)

- GetNode, [139](#), [140](#)
- GetNumberOfNodes, [140](#)
- IncreaseOrder, [140](#)
- Integrate, [140](#)
- operator>>, [141](#)
- operator=, [141](#)
- operator==, [141](#)
- SetNode, [141](#)
- corenc::Mesh::CNodeBasis, [142](#)
 - ~CNodeBasis, [143](#)
 - CNodeBasis, [142](#)
 - GetGradShapeFunction, [143](#)
 - GetMeasure, [143](#)
 - GetNormal, [143](#)
 - GetNumberOfShapeFunctions, [143](#)
 - GetShapeFunction, [143](#)
 - GetWeight, [144](#)
 - IncreaseOrder, [144](#)
 - operator=, [144](#)
 - ReverseNormal, [144](#)
- corenc::Mesh::CParameter, [144](#)
 - ~CParameter, [145](#)
 - CParameter, [145](#)
 - GetAdvection, [145](#), [146](#)
 - GetDiffusion, [146](#)
 - GetMass, [146](#)
- corenc::Mesh::CRectangle, [148](#)
 - ~CRectangle, [150](#)
 - CRectangle, [149](#), [150](#)
 - GetEdge, [150](#)
 - GetFacet, [151](#)
 - GetNode, [151](#)
 - GetNumberOfEdges, [151](#)
 - GetNumberOfFacets, [151](#)
 - GetNumberOfNodes, [151](#)
 - IncreaseOrder, [152](#)
 - Integrate, [152](#)
 - operator>>, [153](#)
 - operator=, [152](#)
 - operator==, [152](#)
 - SetEdge, [153](#)
 - SetFacet, [153](#)
 - SetNode, [153](#)
 - SetOrder, [153](#)
- corenc::Mesh::CRectangleBasis, [154](#)
 - ~CRectangleBasis, [155](#)
 - CRectangleBasis, [154](#), [155](#)
 - GetGradShapeFunction, [155](#)
 - GetMeasure, [155](#)
 - GetNormal, [156](#)
 - GetNumberOfShapeFunctions, [156](#)
 - GetShapeFunction, [156](#)
 - GetValue, [156](#)
 - GetWeight, [156](#)
 - IncreaseOrder, [156](#)
 - operator=, [157](#)
 - ReverseNormal, [157](#)
- corenc::Mesh::CRectangleBasis2, [157](#)
 - ~CRectangleBasis2, [158](#)
 - CRectangleBasis2, [158](#)
 - GetGradShapeFunction, [158](#)
 - GetMeasure, [159](#)
 - GetNormal, [159](#)
 - GetNumberOfShapeFunctions, [159](#)
 - GetShapeFunction, [159](#)
 - GetValue, [159](#)
 - GetWeight, [159](#)
 - IncreaseOrder, [160](#)
 - operator=, [160](#)
 - ReverseNormal, [160](#)
- corenc::Mesh::CRectangleBasis2x, [160](#)
 - ~CRectangleBasis2x, [162](#)
 - CRectangleBasis2x, [161](#)
 - GetGradShapeFunction, [162](#)
 - GetMeasure, [162](#)
 - GetNormal, [162](#)
 - GetNumberOfShapeFunctions, [162](#)
 - GetShapeFunction, [162](#)
 - GetValue, [163](#)
 - GetWeight, [163](#)
 - IncreaseOrder, [163](#)
 - operator=, [163](#)
 - ReverseNormal, [163](#)
- corenc::Mesh::CRectangleBasis2y, [164](#)
 - ~CRectangleBasis2y, [165](#)
 - CRectangleBasis2y, [164](#), [165](#)
 - GetGradShapeFunction, [165](#)
 - GetMeasure, [165](#)
 - GetNormal, [165](#)
 - GetNumberOfShapeFunctions, [165](#)
 - GetShapeFunction, [166](#)
 - GetValue, [166](#)
 - GetWeight, [166](#)
 - IncreaseOrder, [166](#)
 - operator=, [166](#)
 - ReverseNormal, [166](#)
- corenc::Mesh::CRectangleConstantBasis, [167](#)
 - ~CRectangleConstantBasis, [168](#)
 - CRectangleConstantBasis, [167](#), [168](#)
 - GetGradShapeFunction, [168](#)
 - GetMeasure, [168](#)
 - GetNormal, [169](#)
 - GetNumberOfShapeFunctions, [169](#)
 - GetShapeFunction, [169](#)
 - GetValue, [169](#)
 - IncreaseOrder, [169](#)
 - operator=, [169](#)
 - ReverseNormal, [170](#)
- corenc::Mesh::CRectangleHBasis, [170](#)
 - ~CRectangleHBasis, [172](#)
 - CRectangleHBasis, [171](#)
 - GetGradShapeFunction, [172](#)
 - GetMeasure, [172](#)
 - GetNormal, [172](#)
 - GetNumberOfShapeFunctions, [172](#)
 - GetShapeFunction, [172](#)

- GetValue, 173
- GetWeight, 173
- IncreaseOrder, 173
- operator=, 173
- ReverseNormal, 173
- SetOrder, 173
- corenc::Mesh::CRegularMesh, 174
 - ~CRegularMesh, 176
 - Clone, 176
 - CRegularMesh, 175
 - FindElement, 176
 - GetBoundary, 176
 - GetElement, 176
 - GetElements, 177
 - getMinSize, 177
 - GetNode, 177
 - GetNumberOfBoundaries, 177
 - GetNumberOfElements, 177
 - GetNumberOfNodes, 177
 - GetNumberOfNodes, 177
 - getParameter, 178
 - getSolution, 178
 - interpolate, 178
 - operator=, 178
 - refine_h, 179
 - refine_hp, 179
 - refine_hx, 179
 - refine_hy, 179
 - refine_p, 179
 - setParameter, 179
 - updateSolution, 180
- corenc::Mesh::CRegularMesh3D, 180
 - ~CRegularMesh3D, 182
 - Clone, 183
 - CRegularMesh3D, 181, 182
 - FindElement, 183
 - GetBoundary, 183
 - GetElement, 183
 - GetElements, 183
 - getMinSize, 183
 - GetNode, 184
 - GetNumberOfBoundaries, 184
 - GetNumberOfElements, 184
 - GetNumberOfNodes, 184
 - GetNumberOfNodes, 184
 - getParameter, 184
 - getSolution, 185
 - interpolate, 185
 - operator=, 185
 - refine_h, 185
 - refine_hp, 185
 - refine_hx, 185
 - refine_hy, 186
 - refine_p, 186
 - setParameter, 186
 - updateSolution, 186, 187
- corenc::Mesh::CShape, 192
 - ~CShape, 192
 - CShape, 192
 - GetEdge, 193
 - GetFacet, 193
 - GetNode, 193
 - GetNumberOfEdges, 193
 - GetNumberOfFacets, 194
 - GetNumberOfNodes, 194
 - Integrate, 194
 - SetEdge, 194
 - SetFacet, 195
 - SetNode, 195
- corenc::Mesh::CShapeFunction< Type >, 195
 - ~CShapeFunction, 196
 - CShapeFunction, 196
 - GetGradShapeFunction, 196
 - GetMeasure, 196
 - GetNormal, 197
 - GetNumberOfShapeFunctions, 197
 - GetShapeFunction, 197
 - ReverseNormal, 197
- corenc::Mesh::CTriangle, 199
 - ~CTriangle, 201
 - CTriangle, 200, 201
 - GetEdge, 201
 - GetFacet, 202
 - GetNode, 202
 - GetNumberOfEdges, 202
 - GetNumberOfFacets, 202
 - GetNumberOfNodes, 202
 - IncreaseOrder, 203
 - Integrate, 203
 - operator>>, 204
 - operator=, 203
 - operator==, 203
 - SetEdge, 204
 - SetFacet, 204
 - SetNode, 204
- corenc::Mesh::CTriangleBasis, 205
 - ~CTriangleBasis, 206, 207
 - CTriangleBasis, 205–207
 - GetGradShapeFunction, 207
 - GetMeasure, 207
 - GetNormal, 207, 208
 - GetNumberOfShapeFunctions, 208
 - GetShapeFunction, 208
 - GetValue, 208, 209
 - GetWeight, 209
 - IncreaseOrder, 209
 - operator=, 209
 - ReverseNormal, 209
- corenc::Mesh::CTriangleLagrangeBasis, 210
 - ~CTriangleLagrangeBasis, 211
 - CTriangleLagrangeBasis, 210, 211
 - GetAlpha, 211
 - GetGradShapeFunction, 211
 - GetMeasure, 212
 - GetNormal, 212
 - GetNumberOfShapeFunctions, 212

- GetShapeFunction, [212](#)
- GetValue, [212](#)
- GetWeight, [212](#)
- IncreaseOrder, [213](#)
- operator=, [213](#)
- ReverseNormal, [213](#)
- corenc::Mesh::CTriangleLinear, [213](#)
 - ~CTriangleLinear, [215](#)
 - CTriangleLinear, [214](#), [215](#)
 - GetEdge, [215](#)
 - GetFacet, [215](#)
 - GetNode, [216](#)
 - GetNumberOfEdges, [216](#)
 - GetNumberOfFacets, [216](#)
 - GetNumberOfNodes, [216](#)
 - IncreaseOrder, [216](#)
 - Integrate, [217](#)
 - operator>>, [217](#)
 - operator=, [217](#)
 - operator==, [217](#)
 - SetEdge, [218](#)
 - SetFacet, [218](#)
 - SetNode, [218](#)
- corenc::Mesh::CTriangleLinearBasis, [218](#)
 - ~CTriangleLinearBasis, [219](#)
 - CTriangleLinearBasis, [219](#)
 - GetGradShapeFunction, [220](#)
 - GetMeasure, [220](#)
 - GetNormal, [220](#)
 - GetNumberOfShapeFunctions, [220](#)
 - GetShapeFunction, [220](#)
 - GetValue, [220](#)
 - IncreaseOrder, [221](#)
 - operator=, [221](#)
 - ReverseNormal, [221](#)
- corenc::Mesh::CTriangularMesh, [221](#)
 - ~CTriangularMesh, [223](#)
 - Clone, [223](#)
 - CTriangularMesh, [222](#), [223](#)
 - FindElement, [223](#)
 - GetBoundary, [223](#), [224](#)
 - GetElement, [224](#)
 - GetElements, [224](#)
 - getMinSize, [224](#)
 - GetNode, [224](#)
 - GetNumberOfBoundaries, [224](#)
 - GetNumberOfElements, [225](#)
 - GetNumberOfNodes, [225](#)
 - GetNumberOfNodes, [225](#)
 - getParameter, [225](#)
 - getSolution, [225](#), [226](#)
 - interpolate, [226](#)
 - operator=, [226](#)
 - refine_h, [226](#)
 - refine_hp, [226](#)
 - refine_p, [226](#)
 - set2ndOrder, [226](#)
 - set3rdOrder, [227](#)
 - set4thOrder, [227](#)
 - setParameter, [227](#)
 - updateSolution, [227](#), [228](#)
- corenc::Mesh::CTriangularMeshLinear, [228](#)
 - ~CTriangularMeshLinear, [229](#)
 - CTriangularMeshLinear, [229](#)
 - FindElement, [230](#)
 - GetBoundary, [230](#)
 - GetElement, [230](#)
 - GetElements, [230](#)
 - getMinSize, [230](#)
 - GetNode, [231](#)
 - GetNumberOfBoundaries, [231](#)
 - GetNumberOfElements, [231](#)
 - GetNumberOfNodes, [231](#)
 - getParameter, [231](#)
 - getSolution, [232](#)
 - refine_h, [232](#)
 - setParameter, [232](#)
 - updateSolution, [232](#), [233](#)
- corenc::Mesh::Gauss1dim, [282](#)
 - m_a, [282](#)
 - m_order, [282](#)
 - m_sqrt35, [283](#)
 - m_w, [283](#)
- corenc::Mesh::Gauss1dimN< N >, [283](#)
 - m_a, [283](#)
 - m_order, [284](#)
 - m_w, [284](#)
- corenc::Mesh::GaussRectangular, [288](#)
 - m_a, [288](#)
 - m_b, [288](#)
 - m_c, [289](#)
 - m_ra, [289](#)
 - m_rb, [289](#)
 - m_rw, [289](#)
 - m_wa, [289](#)
 - m_wb, [289](#)
 - m_wc, [289](#)
- corenc::Mesh::GaussRectangularCubic, [290](#)
 - m_a, [290](#)
 - m_b, [290](#)
 - m_c, [290](#)
 - m_ra, [291](#)
 - m_rb, [291](#)
 - m_rc, [291](#)
 - m_rw, [291](#)
 - m_s, [291](#)
 - m_w1, [291](#)
 - m_w2, [291](#)
 - m_w3, [291](#)
 - m_w4, [292](#)
- corenc::Mesh::GaussTetrahedron, [292](#)
 - m_la, [292](#)
 - m_lb, [292](#)
 - m_lc, [293](#)
 - m_ld, [293](#)
 - m_msq, [293](#)

- m_psq, 293
 - m_w, 293
- corenc::Mesh::GaussTriangle, 293
 - m_order, 294
 - m_sqrt15, 294
 - m_tra, 294
 - m_trb, 294
 - m_trw, 294
- corenc::Mesh::parameter< T >, 306
 - ~parameter, 308
 - cfunc, 307
 - cfunc_old, 307
 - get, 308
 - parameter, 307, 308
 - set, 309
- corenc::Mesh::Point, 309
 - Jacobian, 310
 - operator!=, 311
 - operator<, 311
 - operator*, 310–312
 - operator*=: 310
 - operator+, 312
 - operator+=, 311
 - operator-, 312
 - operator=, 311
 - operator==, 311
 - Point, 310
 - x, 312
 - y, 312
 - z, 313
- corenc::Mesh::point_source< T >, 313
 - get_point, 314
 - get_value, 314
 - operator=, 314
 - point_source, 313
- corenc::method, 21
 - BoundaryType, 21
 - Central, 22
 - CUSTOM, 22
 - DGFlux, 22
 - EBaumannOden, 22
 - EBaumannOdenIP, 22
 - ECentral, 22
 - EIP, 22
 - ELaxFriedrichs, 22
 - ENIPG, 22
 - EUpwind, 22
 - FREE, 22
 - FVFlux, 22
 - IBaumannOden, 22
 - IBaumannOdenIP, 22
 - ICentral, 22
 - IIP, 22
 - ILaxFriedrichs, 22
 - INIPG, 22
 - IUpwind, 22
 - LaxFriedrichs, 22
 - MAIN, 22
 - NOFLUX, 22
 - SECOND, 22
 - THIRD, 22
 - Upwind, 22
- corenc::method::CDGMethod< Type >, 36
 - ~CDGMethod, 36
 - Assemble, 36
 - CDGMethod, 36
 - GetMaxSolution, 36
 - GetMinSolution, 36
 - GetSolution, 37
- corenc::method::CDGMethodZero< Type >, 37
 - ~CDGMethodZero, 38
 - Assemble, 38
 - CDGMethodZero, 37
 - GetMaxSolution, 38
 - GetMinSolution, 38
 - GetSolution, 38
- corenc::method::CFEMethod< Type >, 82
 - ~CFEMethod, 82
 - Assemble, 83
 - CFEMethod, 82
 - GetMaxSolution, 83
 - GetMinSolution, 83
 - GetSolution, 83
- corenc::method::CFEMethodZero< Type >, 83
 - ~CFEMethodZero, 84
 - Assemble, 84
 - CFEMethodZero, 84
 - GetMaxSolution, 84
 - GetMinSolution, 84
 - GetSolution, 85
- corenc::method::DGMethod< Problem, Grid, Matrix >, 239
 - ~DGMethod, 241
 - DGMethod, 240, 241
 - Discretization, 242
 - GetEffective, 242
 - GetGlobalMatrix, 242
 - GetGradSolution, 242
 - GetMesh, 242
 - GetRightVector, 243
 - GetSolution, 243
 - GetValue, 243, 244
 - LoadSolution, 244
 - OutDatFormat, 244
 - OutMeshFormat, 244
 - OutMeshTimeFormat, 244
 - ProjectSolution, 245
 - Rediscretization, 245
 - SetSolution, 245
 - SetTimeStep, 246
- corenc::method::DGMethodZero< Problem, Grid, Matrix >, 246
 - ~DGMethodZero, 248
 - DGMethodZero, 247, 248
 - Discretization, 248
 - GetEffective, 248

- GetGlobalMatrix, [248](#)
- GetGradSolution, [249](#)
- GetMesh, [249](#)
- GetRightVector, [249](#)
- GetSolution, [249](#), [250](#)
- GetValue, [250](#)
- LoadSolution, [250](#)
- OutDatFormat, [251](#)
- OutMeshFormat, [251](#)
- OutMeshTimeFormat, [251](#)
- ProjectSolution, [251](#)
- Rediscretization, [252](#)
- SetSolution, [252](#)
- SetTimeStep, [252](#)
- corenc::method::DGSolution< Grid >, [253](#)
 - ~DGSolution, [253](#)
 - DGSolution, [253](#)
 - getWeight, [254](#)
 - getWeights, [254](#)
 - operator=, [254](#)
 - updateWeight, [254](#)
- corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >, [261](#)
 - ~FEAnalysis, [261](#)
 - FEAnalysis, [261](#)
 - L2Norm, [261](#)
- corenc::method::FEMethod< Problem, Grid, Matrix >, [267](#)
 - ~FEMethod, [269](#)
 - Discretization, [269](#)
 - FEMethod, [268](#), [269](#)
 - GetEffective, [270](#)
 - GetGlobalMatrix, [270](#)
 - GetGradSolution, [270](#)
 - GetMesh, [270](#)
 - GetRightVector, [270](#)
 - GetSolution, [271](#)
 - GetValue, [271](#)
 - LoadSolution, [272](#)
 - operator=, [272](#)
 - OutDatFormat, [272](#)
 - OutMeshFormat, [272](#)
 - OutMeshTimeFormat, [272](#)
 - ProjectSolution, [273](#)
 - Rediscretization, [273](#)
 - SetSolution, [273](#)
 - SetTimeStep, [274](#)
- corenc::method::FEMethodZero< Problem, Grid, Matrix >, [274](#)
 - ~FEMethodZero, [276](#)
 - Discretization, [276](#)
 - FEMethodZero, [275](#), [276](#)
 - GetEffective, [276](#)
 - GetGlobalMatrix, [276](#)
 - GetGradSolution, [277](#)
 - GetMesh, [277](#)
 - GetRightVector, [277](#)
 - GetSolution, [277](#), [278](#)
 - GetValue, [278](#)
 - LoadSolution, [278](#)
 - OutDatFormat, [279](#)
 - OutMeshFormat, [279](#)
 - OutMeshTimeFormat, [279](#)
 - ProjectSolution, [279](#)
 - Rediscretization, [280](#)
 - SetSolution, [280](#)
 - SetTimeStep, [280](#)
- corenc::method::FVMethod1d, [281](#)
 - ~FVMethod1d, [281](#)
 - FVMethod1d, [281](#)
 - GetSolution, [281](#)
 - Solve, [281](#)
- corenc::method::RungeKutta< Problem, Type >, [314](#)
 - ~RungeKutta, [315](#)
 - discretize, [315](#)
 - explicitEuler, [315](#)
 - RungeKutta, [315](#)
 - updateTimestep, [315](#)
- corenc::method::STSolution< Grid >, [316](#)
 - ~STSolution, [317](#)
 - addTimeLayer, [317](#)
 - getWeight, [317](#)
 - getWeights, [317](#)
 - operator=, [318](#)
 - STSolution, [316](#), [317](#)
 - updateWeight, [318](#)
- corenc::method::system_dg_method< Grid, bool, bool >, [322](#)
 - GetSolution, [322](#)
- corenc::method::system_dg_method< Problem, Grid, Matrix >, [318](#)
 - ~system_dg_method, [319](#)
 - Assemble, [319](#)
 - changeFlux, [319](#)
 - DGtostandart, [320](#)
 - GetGlobalMatrix, [320](#)
 - GetMaxSolution, [320](#)
 - GetMinSolution, [320](#)
 - GetSolution, [320](#), [321](#)
 - system_dg_method, [319](#)
 - toDGSolution, [321](#)
 - updateWeights, [321](#)
- corenc::multi_vector< T >, [304](#)
 - ~multi_vector, [304](#)
 - fill_inc, [305](#)
 - get, [305](#)
 - multi_vector, [304](#)
 - resize, [305](#)
 - set, [305](#)
 - size, [306](#)
 - totalsize, [306](#)
- corenc::solvers, [23](#)
- corenc::solvers::dg_shallow_water< Mesh >, [234](#)
 - ~dg_shallow_water, [235](#)
 - dg_shallow_water, [235](#)
 - solve, [235](#)

- corenc::solvers::dg_solver< _Problem, _Mesh, _Result
 >, 236
- ~dg_solver, 236
- dg_solver, 236
- elliptic_solver, 237
- get_gradvalue, 237
- get_value, 237, 238
- corenc::solvers::dg_solver_shallow_water, 238
- ~dg_solver_shallow_water, 238
- dg_solver_shallow_water, 238
- solve, 239
- corenc::solvers::eigen_solver< Matrix, Solver >, 254
- ~eigen_solver, 255
- eigen_solver, 255
- rayleigh, 255
- corenc::solvers::fem_solver< _Problem, _Mesh, _Re-
 sult >, 262
- ~fem_solver, 262
- elliptic_solver, 263
- elliptic_solver_gauss, 263
- fem_solver, 262
- get_gradvalue, 263
- get_value, 263, 264
- corenc::solvers::fem_solver_lib< _Problem, _Mesh,
 _Result >, 264
- ~fem_solver_lib, 265
- elliptic_solver, 265
- elliptic_solver_gauss, 265
- fem_solver_lib, 265
- get_gradvalue, 266
- get_value, 266, 267
- corenc::solvers::vector_solution, 332
- S, 332
- vector_solution, 332
- corenc::test_case_elliptic_fem, 322
- ~test_case_elliptic_fem, 323
- conv_diff_fem_fixed_triangle, 323
- elliptic_2layer_fem_2d_tria_h, 323
- elliptic_fem_2d_rect_source, 323
- elliptic_fem_2d_tria, 323
- elliptic_fem_hp_fixed, 323
- elliptic_fem_hp_fixed_triangle, 324
- elliptic_fem_hp_lagrange_triangle, 324
- elliptic_fem_hxhy_fixed_triangle, 324
- elliptic_fem_solver, 324
- elliptic_fem_square_lin_basis, 324
- elliptic_gaussian_triangle, 324
- global_matrix, 324
- homotopy_conv_diff_fem, 325
- mass_matrix_3rd_order, 325
- mass_matrix_4th_order, 325
- strees_matrix_3rd_order, 325
- stress_matrix_4th_order, 325
- test_case_elliptic_fem, 323
- corenc::test_case_solver, 327
- ~test_case_solver, 328
- gauss_solver, 328
- test_case_solver, 328
- corenc::test_cases, 329
- ~test_cases, 330
- perform, 330
- test_cases, 329
- corenc::test_conv_diff, 330
- ~test_conv_diff, 331
- conv_diff_eigen, 331
- conv_diff_fem, 331
- test_conv_diff, 331
- corenc::tests, 23
- corenc::tests::test_case_rectanglebasis, 325
- ~test_case_rectanglebasis, 326
- mass_matrix, 326
- stress_matrix, 326
- test_case_rectanglebasis, 326
- corenc::tests::test_case_regular_mesh, 326
- ~test_case_regular_mesh, 327
- construct_mesh, 327
- test_case_regular_mesh, 327
- corenc::tests::test_case_trianglebasis, 328
- ~test_case_trianglebasis, 329
- mass_matrix, 329
- stress_matrix, 329
- test_case_trianglebasis, 328
- CORENC_MESH_CUBE_H_
 Cube.h, 344
- CORENC_MESH_PARAMETER_H_
 Parameter.h, 489
- CORENC_MESH_Point_h
 Point.h, 491
- CORENC_METHODS_FEANALYSIS_H_
 FEAnalysis.h, 442
- CORENC_METHODS_SYSTEM_DG_METHOD_H_
 system_dg_method.h, 479
- CORENC_MULTI_VECTOR_H_
 multi_vector.h, 486
- CORENC_PROBLEMS_PROBLEMS_H_
 Problems.h, 497
- CORENC_TEST_CASE_RECTANGLEBASIS_H_
 test_case_rectanglebasis.h, 516
- CORENC_TEST_CASE_REGULAR_MESH_H_
 test_case_regular_mesh.h, 520
- CORENC_TEST_CASE_TRIANGLEBASIS_H_
 test_case_trianglebasis.h, 517
- CORENC_TEST_CASES_H_
 test_cases.h, 522
- CoreNCA/Matrix.cpp, 334
- CoreNCA/Matrix.h, 334, 335
- CoreNCA/MatrixDiag.cpp, 335
- CoreNCA/MatrixDiag.h, 335, 336
- CoreNCA/MatrixSkyline.cpp, 336
- CoreNCA/MatrixSkyline.h, 337, 338
- CoreNCFEM/FESolution.h, 341, 342
- CoreNCFEM/FiniteElements/CRectangleBasis2x.cpp,
 343
- CoreNCFEM/FiniteElements/Cube.cpp, 343
- CoreNCFEM/FiniteElements/Cube.h, 343, 344
- CoreNCFEM/FiniteElements/CubeHBasis.cpp, 346

- CoreNCFEM/FiniteElements/Edge.cpp, 346
- CoreNCFEM/FiniteElements/Edge.h, 346, 347
- CoreNCFEM/FiniteElements/FiniteElement.h, 350, 351
- CoreNCFEM/FiniteElements/FiniteElement2D.h, 363, 364
- CoreNCFEM/FiniteElements/Node.cpp, 368
- CoreNCFEM/FiniteElements/Node.h, 368, 369
- CoreNCFEM/FiniteElements/Rectangle.cpp, 370
- CoreNCFEM/FiniteElements/Rectangle.h, 370, 371
- CoreNCFEM/FiniteElements/RectangleBasis2.cpp, 376
- CoreNCFEM/FiniteElements/RectangleBasis2y.cpp, 376
- CoreNCFEM/FiniteElements/RectangleHBasis.cpp, 376
- CoreNCFEM/FiniteElements/Shape.h, 377
- CoreNCFEM/FiniteElements/ShapeFunction.h, 378
- CoreNCFEM/FiniteElements/Triangle.cpp, 379
- CoreNCFEM/FiniteElements/Triangle.h, 380
- CoreNCFEM/FiniteElements/TriangleLagrange.cpp, 383
- CoreNCFEM/FiniteElements/TriangleLinear.cpp, 383
- CoreNCFEM/FiniteElements/TriangleLinear.h, 383, 384
- CoreNCFEM/FiniteSolver.h, 386
- CoreNCFEM/GaussianField.h, 387
- CoreNCFEM/Grids/Mesh1D.cpp, 388
- CoreNCFEM/Grids/Mesh1D.h, 389
- CoreNCFEM/Grids/RegularMesh.cpp, 390
- CoreNCFEM/Grids/RegularMesh.h, 391
- CoreNCFEM/Grids/RegularMesh3D.cpp, 393
- CoreNCFEM/Grids/RegularMesh3D.h, 393, 394
- CoreNCFEM/Grids/TriangularMesh.cpp, 395
- CoreNCFEM/Grids/TriangularMesh.h, 396
- CoreNCFEM/Grids/TriangularMeshLinear.cpp, 398
- CoreNCFEM/Grids/TriangularMeshLinear.h, 398
- CoreNCFEM/Mesh.h, 399, 400
- CoreNCFEM/Methods/CSMethod.h, 401
- CoreNCFEM/Methods/dg_flux.h, 401, 402
- CoreNCFEM/Methods/DGMethod.h, 402, 403
- CoreNCFEM/Methods/DGMethodZero.h, 421, 422
- CoreNCFEM/Methods/DGSolution.h, 440, 441
- CoreNCFEM/Methods/FEAnalysis.h, 442, 443
- CoreNCFEM/Methods/FEMethod.h, 443, 444
- CoreNCFEM/Methods/FEMethodZero.h, 458
- CoreNCFEM/Methods/FVMMethod.cpp, 475
- CoreNCFEM/Methods/FVMMethod.h, 475, 476
- CoreNCFEM/Methods/RungeKutta.h, 476
- CoreNCFEM/Methods/system_dg_method.h, 478, 479
- CoreNCFEM/multi_vector.h, 486, 487
- CoreNCFEM/Parameter.cpp, 488
- CoreNCFEM/Parameter.h, 488, 489
- CoreNCFEM/Point.cpp, 490
- CoreNCFEM/Point.h, 490, 491
- CParameter
 - corenc::Mesh::CParameter, 145
- CProblem
 - corenc::CProblem, 147
- Create
 - Algebra::Matrix, 296
 - Algebra::MatrixDiag, 299
 - Algebra::MatrixSkyline, 302
- CRectangle
 - corenc::Mesh::CRectangle, 149, 150
- CRectangleBasis
 - corenc::Mesh::CRectangleBasis, 154, 155
- CRectangleBasis2
 - corenc::Mesh::CRectangleBasis2, 158
- CRectangleBasis2x
 - corenc::Mesh::CRectangleBasis2x, 161
- CRectangleBasis2y
 - corenc::Mesh::CRectangleBasis2y, 164, 165
- CRectangleConstantBasis
 - corenc::Mesh::CRectangleConstantBasis, 167, 168
- CRectangleHBasis
 - corenc::Mesh::CRectangleHBasis, 171
- CRegularMesh
 - corenc::Mesh::CRegularMesh, 175
- CRegularMesh3D
 - corenc::Mesh::CRegularMesh3D, 181, 182
- CShallowWater
 - corenc::CShallowWater, 188
- CShape
 - corenc::Mesh::CShape, 192
- CShapeFunction
 - corenc::Mesh::CShapeFunction< Type >, 196
- CSMethod
 - Methods::CSMethod, 198
- CSolution
 - corenc::CSolution, 199
- CTriangle
 - corenc::Mesh::CTriangle, 200, 201
- CTriangleBasis
 - corenc::Mesh::CTriangleBasis, 205–207
- CTriangleLagrangeBasis
 - corenc::Mesh::CTriangleLagrangeBasis, 210, 211
- CTriangleLinear
 - corenc::Mesh::CTriangleLinear, 214, 215
- CTriangleLinearBasis
 - corenc::Mesh::CTriangleLinearBasis, 219
- CTriangularMesh
 - corenc::Mesh::CTriangularMesh, 222, 223
- CTriangularMeshLinear
 - corenc::Mesh::CTriangularMeshLinear, 229
- Cube
 - corenc::Mesh, 20
- Cube.h
 - CORENC_MESH_CUBE_H_, 344
- CUSTOM
 - corenc::method, 22
- CVecSolution
 - corenc::CVecSolution, 234
- CYAN
 - corenc::color, 17
- dg_shallow_water
 - corenc::solvers::dg_shallow_water< Mesh >, 235
- dg_solver
 - corenc::solvers::dg_solver< _Problem, _Mesh, _Result >, 236

- dg_solver_shallow_water
 - corenc::solvers::dg_solver_shallow_water, [238](#)
- DGFlux
 - corenc::method, [22](#)
- DGMethod
 - corenc::method::DGMethod< Problem, Grid, Matrix >, [240](#), [241](#)
- DGMethodZero
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, [247](#), [248](#)
- DGSolution
 - corenc::method::DGSolution< Grid >, [253](#)
- DGtostandart
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, [320](#)
- diff_skymatrix
 - Algebra::MatrixSkyline, [302](#)
- DIFFUSION
 - corenc, [15](#)
- Discretization
 - corenc::method::DGMethod< Problem, Grid, Matrix >, [242](#)
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, [248](#)
 - corenc::method::FEMethod< Problem, Grid, Matrix >, [269](#)
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, [276](#)
- discretize
 - corenc::method::RungeKutta< Problem, Type >, [315](#)
- EBaumannOden
 - corenc::method, [22](#)
- EBaumannOdenIP
 - corenc::method, [22](#)
- ECentral
 - corenc::method, [22](#)
- EDUDV
 - corenc, [15](#)
- EDUV
 - corenc, [15](#)
- EFV
 - corenc, [15](#)
- eigen_solver
 - corenc::solvers::eigen_solver< Matrix, Solver >, [255](#)
- EIP
 - corenc::method, [22](#)
- ELaxFriedrichs
 - corenc::method, [22](#)
- Elements
 - corenc::Mesh, [20](#)
- elliptic_2layer_fem_2d_tria_h
 - corenc::test_case_elliptic_fem, [323](#)
- elliptic_fem_2d_rect_source
 - corenc::test_case_elliptic_fem, [323](#)
- elliptic_fem_2d_tria
 - corenc::test_case_elliptic_fem, [323](#)
- elliptic_fem_hp_fixed
 - corenc::test_case_elliptic_fem, [323](#)
- elliptic_fem_hp_fixed_triangle
 - corenc::test_case_elliptic_fem, [324](#)
- elliptic_fem_hp_lagrange_triangle
 - corenc::test_case_elliptic_fem, [324](#)
- elliptic_fem_hxhy_fixed_triangle
 - corenc::test_case_elliptic_fem, [324](#)
- elliptic_fem_solver
 - corenc::test_case_elliptic_fem, [324](#)
- elliptic_fem_square_lin_basis
 - corenc::test_case_elliptic_fem, [324](#)
- elliptic_gaussian_triangle
 - corenc::test_case_elliptic_fem, [324](#)
- elliptic_solver
 - corenc::solvers::dg_solver< _Problem, _Mesh, _Result >, [237](#)
 - corenc::solvers::fem_solver< _Problem, _Mesh, _Result >, [263](#)
 - corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >, [265](#)
- elliptic_solver_gauss
 - corenc::solvers::fem_solver< _Problem, _Mesh, _Result >, [263](#)
 - corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >, [265](#)
- ENIPG
 - corenc::method, [22](#)
- ESCAPE
 - corenc::color, [17](#)
- ESolver
 - Algebra::ESolver, [256](#)
- EUDV
 - corenc, [15](#)
- EUpwind
 - corenc::method, [22](#)
- EUV
 - corenc, [15](#)
- explicitEuler
 - corenc::method::RungeKutta< Problem, Type >, [315](#)
- FEAnalysis
 - corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >, [261](#)
- FEAnalysis.h
 - CORENC_METHODS_FEANALYSIS_H_, [442](#)
- fem_solver
 - corenc::solvers::fem_solver< _Problem, _Mesh, _Result >, [262](#)
- fem_solver_lib
 - corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >, [265](#)
- FEMethod
 - corenc::method::FEMethod< Problem, Grid, Matrix >, [268](#), [269](#)
- FEMethodZero
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, [275](#), [276](#)

- fill_inc
 - corenc::multi_vector< T >, 305
- FindElement
 - corenc::Mesh::CMesh< bool >, 135
 - corenc::Mesh::CMesh< T >, 125
 - corenc::Mesh::CMesh1D, 131
 - corenc::Mesh::CRegularMesh, 176
 - corenc::Mesh::CRegularMesh3D, 183
 - corenc::Mesh::CTriangularMesh, 223
 - corenc::Mesh::CTriangularMeshLinear, 230
- findTerm
 - corenc::CDiffusionScalar, 41
- FIRST
 - corenc::Mesh, 21
- FREE
 - corenc::method, 22
- function_dp
 - corenc::Mesh, 20
- FVFlux
 - corenc::method, 22
- FVMethod1d
 - corenc::method::FVMethod1d, 281
- Gauss
 - Algebra, 13
 - Algebra::ESolver, 257, 258
- gauss_solver
 - corenc::test_case_solver, 328
- GaussianKernel
 - corenc::GaussianKernel, 284
- GaussianProcess
 - corenc::GaussianProcess, 286
- get
 - corenc::Mesh::parameter< T >, 308
 - corenc::multi_vector< T >, 305
- get_boundary_parameter
 - corenc::CDiffusionScalar, 41, 42
 - corenc::CShallowWater, 189
- get_boundary_type
 - corenc::CDiffusionScalar, 42
 - corenc::CShallowWater, 189
- get_gp
 - corenc::GaussianKernel, 285
- get_gradvalue
 - corenc::solvers::dg_solver< _Problem, _Mesh, _Result >, 237
 - corenc::solvers::fem_solver< _Problem, _Mesh, _Result >, 263
 - corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >, 266
- get_number_of_boundaries
 - corenc::CDiffusionScalar, 42
 - corenc::CShallowWater, 189
- get_parameter
 - corenc::CDiffusionScalar, 42, 43
 - corenc::CShallowWater, 189, 190
- get_point
 - corenc::Mesh::point_source< T >, 314
- get_point_source
 - corenc::CDiffusionScalar, 43
- get_solution
 - corenc::CShallowWater, 190
- get_total_sources
 - corenc::CDiffusionScalar, 43
- get_value
 - corenc::Mesh::point_source< T >, 314
 - corenc::solvers::dg_solver< _Problem, _Mesh, _Result >, 237, 238
 - corenc::solvers::fem_solver< _Problem, _Mesh, _Result >, 263, 264
 - corenc::solvers::fem_solver_lib< _Problem, _Mesh, _Result >, 266, 267
- GetAdvection
 - corenc::Mesh::CParameter, 145, 146
- GetAlpha
 - corenc::Mesh::CTriangleLagrangeBasis, 211
- GetBoundary
 - corenc::Mesh::CMesh< bool >, 135
 - corenc::Mesh::CMesh< T >, 125
 - corenc::Mesh::CMesh1D, 131
 - corenc::Mesh::CRegularMesh, 176
 - corenc::Mesh::CRegularMesh3D, 183
 - corenc::Mesh::CTriangularMesh, 223, 224
 - corenc::Mesh::CTriangularMeshLinear, 230
- GetDiffusion
 - corenc::Mesh::CParameter, 146
- GetDoF
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 118
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 93
- GetDoFs
 - corenc::Mesh::CElement< bool >, 79
 - corenc::Mesh::CElement< T >, 65
 - corenc::Mesh::CElement2D< bool >, 75
 - corenc::Mesh::CElement2D< T >, 70
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 110
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 118
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 93
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 101
- GetEdge
 - corenc::Mesh::CCube, 29
 - corenc::Mesh::CRectangle, 150
 - corenc::Mesh::CShape, 193
 - corenc::Mesh::CTriangle, 201
 - corenc::Mesh::CTriangleLinear, 215
- GetEffective
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 242
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 248
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 270

- corenc::method::FEMethodZero< Problem, Grid, Matrix >, 276
- GetElement
 - Algebra::Matrix, 297
 - Algebra::MatrixSkyline, 302
 - corenc::Mesh::CMesh< bool >, 136
 - corenc::Mesh::CMesh< T >, 125
 - corenc::Mesh::CMesh1D, 131
 - corenc::Mesh::CRegularMesh, 176
 - corenc::Mesh::CRegularMesh3D, 183
 - corenc::Mesh::CTriangularMesh, 224
 - corenc::Mesh::CTriangularMeshLinear, 230
- GetElements
 - corenc::Mesh::CMesh1D, 131
 - corenc::Mesh::CRegularMesh, 177
 - corenc::Mesh::CRegularMesh3D, 183
 - corenc::Mesh::CTriangularMesh, 224
 - corenc::Mesh::CTriangularMeshLinear, 230
- GetFacet
 - corenc::Mesh::CCube, 29
 - corenc::Mesh::CRectangle, 151
 - corenc::Mesh::CShape, 193
 - corenc::Mesh::CTriangle, 202
 - corenc::Mesh::CTriangleLinear, 215
- getFlux
 - corenc::CBurgersScalar, 26
- GetGlobalMatrix
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 242
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 248
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 270
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 276
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, 320
- GetGradShapeFunction
 - corenc::Mesh::CCubeBasis, 34
 - corenc::Mesh::CEdge2ndBasis, 50
 - corenc::Mesh::CEdgeConstantBasis, 53
 - corenc::Mesh::CEdgeHermiteBasis, 56
 - corenc::Mesh::CEdgeLinearBasis, 59
 - corenc::Mesh::CEdgeMultiBasis, 62
 - corenc::Mesh::CElement< bool >, 79
 - corenc::Mesh::CElement< T >, 65
 - corenc::Mesh::CElement2D< bool >, 75
 - corenc::Mesh::CElement2D< T >, 70
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 110
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 118
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 93
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 102
 - corenc::Mesh::CNodeBasis, 143
 - corenc::Mesh::CRectangleBasis, 155
- corenc::Mesh::CRectangleBasis2, 158
- corenc::Mesh::CRectangleBasis2x, 162
- corenc::Mesh::CRectangleBasis2y, 165
- corenc::Mesh::CRectangleConstantBasis, 168
- corenc::Mesh::CRectangleHBasis, 172
- corenc::Mesh::CShapeFunction< Type >, 196
- corenc::Mesh::CTriangleBasis, 207
- corenc::Mesh::CTriangleLagrangeBasis, 211
- corenc::Mesh::CTriangleLinearBasis, 220
- GetGradSolution
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 242
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 249
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 270
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 277
- GetMass
 - corenc::Mesh::CParameter, 146
- GetMaxSolution
 - corenc::method::CDGMethod< Type >, 36
 - corenc::method::CDGMethodZero< Type >, 38
 - corenc::method::CFEMethod< Type >, 83
 - corenc::method::CFEMethodZero< Type >, 84
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, 320
- GetMeasure
 - corenc::Mesh::CCubeBasis, 34
 - corenc::Mesh::CEdge2ndBasis, 50
 - corenc::Mesh::CEdgeConstantBasis, 53
 - corenc::Mesh::CEdgeHermiteBasis, 56
 - corenc::Mesh::CEdgeLinearBasis, 59
 - corenc::Mesh::CEdgeMultiBasis, 62
 - corenc::Mesh::CElement< bool >, 79
 - corenc::Mesh::CElement< T >, 65
 - corenc::Mesh::CElement2D< bool >, 75
 - corenc::Mesh::CElement2D< T >, 70
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 110
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 118
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 93
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 102
 - corenc::Mesh::CNodeBasis, 143
 - corenc::Mesh::CRectangleBasis, 155
 - corenc::Mesh::CRectangleBasis2, 159
 - corenc::Mesh::CRectangleBasis2x, 162
 - corenc::Mesh::CRectangleBasis2y, 165
 - corenc::Mesh::CRectangleConstantBasis, 168
 - corenc::Mesh::CRectangleHBasis, 172
 - corenc::Mesh::CShapeFunction< Type >, 196
 - corenc::Mesh::CTriangleBasis, 207
 - corenc::Mesh::CTriangleLagrangeBasis, 212
 - corenc::Mesh::CTriangleLinearBasis, 220
- GetMesh

- corenc::method::DGMethod< Problem, Grid, Matrix >, [242](#)
- corenc::method::DGMethodZero< Problem, Grid, Matrix >, [249](#)
- corenc::method::FEMethod< Problem, Grid, Matrix >, [270](#)
- corenc::method::FEMethodZero< Problem, Grid, Matrix >, [277](#)
- getMinSize
 - corenc::Mesh::CMesh< bool >, [136](#)
 - corenc::Mesh::CMesh< T >, [125](#)
 - corenc::Mesh::CMesh1D, [131](#)
 - corenc::Mesh::CRegularMesh, [177](#)
 - corenc::Mesh::CRegularMesh3D, [183](#)
 - corenc::Mesh::CTriangularMesh, [224](#)
 - corenc::Mesh::CTriangularMeshLinear, [230](#)
- GetMinSolution
 - corenc::method::CDGMethod< Type >, [36](#)
 - corenc::method::CDGMethodZero< Type >, [38](#)
 - corenc::method::CFEMethod< Type >, [83](#)
 - corenc::method::CFEMethodZero< Type >, [84](#)
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, [320](#)
- GetNeighbour
 - corenc::Mesh::CElement< bool >, [79](#)
 - corenc::Mesh::CElement< T >, [65](#)
 - corenc::Mesh::CElement2D< bool >, [75](#)
 - corenc::Mesh::CElement2D< T >, [71](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [110](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [118](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [93](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [102](#)
- GetNode
 - corenc::Mesh::CCube, [29](#), [30](#)
 - corenc::Mesh::CEdge, [46](#), [47](#)
 - corenc::Mesh::CElement< bool >, [80](#)
 - corenc::Mesh::CElement< T >, [66](#)
 - corenc::Mesh::CElement2D< bool >, [76](#)
 - corenc::Mesh::CElement2D< T >, [71](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [110](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [119](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [94](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [102](#)
 - corenc::Mesh::CMesh< bool >, [136](#)
 - corenc::Mesh::CMesh< T >, [126](#)
 - corenc::Mesh::CMesh1D, [132](#)
 - corenc::Mesh::CNode, [139](#), [140](#)
 - corenc::Mesh::CRectangle, [151](#)
 - corenc::Mesh::CRegularMesh, [177](#)
 - corenc::Mesh::CRegularMesh3D, [184](#)
- corenc::Mesh::CShape, [193](#)
- corenc::Mesh::CTriangle, [202](#)
- corenc::Mesh::CTriangleLinear, [216](#)
- corenc::Mesh::CTriangularMesh, [224](#)
- corenc::Mesh::CTriangularMeshLinear, [231](#)
- GetNormal
 - corenc::Mesh::CCubeBasis, [34](#)
 - corenc::Mesh::CEdge2ndBasis, [50](#)
 - corenc::Mesh::CEdgeConstantBasis, [53](#)
 - corenc::Mesh::CEdgeHermiteBasis, [56](#)
 - corenc::Mesh::CEdgeLinearBasis, [59](#)
 - corenc::Mesh::CEdgeMultiBasis, [62](#)
 - corenc::Mesh::CElement< bool >, [80](#)
 - corenc::Mesh::CElement< T >, [66](#)
 - corenc::Mesh::CElement2D< bool >, [76](#)
 - corenc::Mesh::CElement2D< T >, [71](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [111](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [119](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [94](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [102](#)
 - corenc::Mesh::CNodeBasis, [143](#)
 - corenc::Mesh::CRectangleBasis, [156](#)
 - corenc::Mesh::CRectangleBasis2, [159](#)
 - corenc::Mesh::CRectangleBasis2x, [162](#)
 - corenc::Mesh::CRectangleBasis2y, [165](#)
 - corenc::Mesh::CRectangleConstantBasis, [169](#)
 - corenc::Mesh::CRectangleHBasis, [172](#)
 - corenc::Mesh::CShapeFunction< Type >, [197](#)
 - corenc::Mesh::CTriangleBasis, [207](#), [208](#)
 - corenc::Mesh::CTriangleLagrangeBasis, [212](#)
 - corenc::Mesh::CTriangleLinearBasis, [220](#)
- GetNumberOfBoundaries
 - corenc::Mesh::CMesh< bool >, [136](#)
 - corenc::Mesh::CMesh< T >, [126](#)
 - corenc::Mesh::CMesh1D, [132](#)
 - corenc::Mesh::CRegularMesh, [177](#)
 - corenc::Mesh::CRegularMesh3D, [184](#)
 - corenc::Mesh::CTriangularMesh, [224](#)
 - corenc::Mesh::CTriangularMeshLinear, [231](#)
- GetNumberOfEdges
 - corenc::Mesh::CCube, [30](#)
 - corenc::Mesh::CRectangle, [151](#)
 - corenc::Mesh::CShape, [193](#)
 - corenc::Mesh::CTriangle, [202](#)
 - corenc::Mesh::CTriangleLinear, [216](#)
- GetNumberOfElements
 - corenc::Mesh::CMesh< bool >, [136](#)
 - corenc::Mesh::CMesh< T >, [126](#)
 - corenc::Mesh::CMesh1D, [132](#)
 - corenc::Mesh::CRegularMesh, [177](#)
 - corenc::Mesh::CRegularMesh3D, [184](#)
 - corenc::Mesh::CTriangularMesh, [225](#)
 - corenc::Mesh::CTriangularMeshLinear, [231](#)
- GetNumberOfFacets

- corenc::Mesh::CCube, 30
- corenc::Mesh::CRectangle, 151
- corenc::Mesh::CShape, 194
- corenc::Mesh::CTriangle, 202
- corenc::Mesh::CTriangleLinear, 216
- GetNumberOfNodes
 - corenc::Mesh::CRegularMesh, 177
 - corenc::Mesh::CRegularMesh3D, 184
 - corenc::Mesh::CTriangularMesh, 225
- GetNumberOfNodes
 - corenc::Mesh::CCube, 30
 - corenc::Mesh::CEdge, 47
 - corenc::Mesh::CElement< bool >, 80
 - corenc::Mesh::CElement< T >, 66
 - corenc::Mesh::CElement2D< bool >, 76
 - corenc::Mesh::CElement2D< T >, 71
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 111
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 119
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 94
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 103
 - corenc::Mesh::CMesh< bool >, 136
 - corenc::Mesh::CMesh< T >, 126
 - corenc::Mesh::CMesh1D, 132
 - corenc::Mesh::CNode, 140
 - corenc::Mesh::CRectangle, 151
 - corenc::Mesh::CRegularMesh, 177
 - corenc::Mesh::CRegularMesh3D, 184
 - corenc::Mesh::CShape, 194
 - corenc::Mesh::CTriangle, 202
 - corenc::Mesh::CTriangleLinear, 216
 - corenc::Mesh::CTriangularMesh, 225
 - corenc::Mesh::CTriangularMeshLinear, 231
- GetNumberOfShapeFunctions
 - corenc::Mesh::CCubeBasis, 34
 - corenc::Mesh::CEdge2ndBasis, 50
 - corenc::Mesh::CEdgeConstantBasis, 53
 - corenc::Mesh::CEdgeHermiteBasis, 56
 - corenc::Mesh::CEdgeLinearBasis, 59
 - corenc::Mesh::CEdgeMultiBasis, 62
 - corenc::Mesh::CNodeBasis, 143
 - corenc::Mesh::CRectangleBasis, 156
 - corenc::Mesh::CRectangleBasis2, 159
 - corenc::Mesh::CRectangleBasis2x, 162
 - corenc::Mesh::CRectangleBasis2y, 165
 - corenc::Mesh::CRectangleConstantBasis, 169
 - corenc::Mesh::CRectangleHBasis, 172
 - corenc::Mesh::CShapeFunction< Type >, 197
 - corenc::Mesh::CTriangleBasis, 208
 - corenc::Mesh::CTriangleLagrangeBasis, 212
 - corenc::Mesh::CTriangleLinearBasis, 220
- getNumberOfTerms
 - corenc::CBurgersScalar, 26
 - corenc::CDiffusionScalar, 43
 - corenc::CProblem, 147
 - corenc::CShallowWater, 190
- getParameter
 - corenc::Mesh::CMesh< bool >, 136, 137
 - corenc::Mesh::CMesh< T >, 126, 127
 - corenc::Mesh::CMesh1D, 132
 - corenc::Mesh::CRegularMesh, 178
 - corenc::Mesh::CRegularMesh3D, 184
 - corenc::Mesh::CTriangularMesh, 225
 - corenc::Mesh::CTriangularMeshLinear, 231
- GetRightVector
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 243
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 249
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 270
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 277
- GetShape
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 111
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 119
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 94
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 103
- GetShapeFunction
 - corenc::Mesh::CCubeBasis, 34
 - corenc::Mesh::CEdge2ndBasis, 51
 - corenc::Mesh::CEdgeConstantBasis, 54
 - corenc::Mesh::CEdgeHermiteBasis, 57
 - corenc::Mesh::CEdgeLinearBasis, 60
 - corenc::Mesh::CEdgeMultiBasis, 63
 - corenc::Mesh::CElement< bool >, 80
 - corenc::Mesh::CElement< T >, 66
 - corenc::Mesh::CElement2D< bool >, 76
 - corenc::Mesh::CElement2D< T >, 71
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 111
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 119
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 94
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 103
 - corenc::Mesh::CNodeBasis, 143
 - corenc::Mesh::CRectangleBasis, 156
 - corenc::Mesh::CRectangleBasis2, 159
 - corenc::Mesh::CRectangleBasis2x, 162
 - corenc::Mesh::CRectangleBasis2y, 166
 - corenc::Mesh::CRectangleConstantBasis, 169
 - corenc::Mesh::CRectangleHBasis, 172
 - corenc::Mesh::CShapeFunction< Type >, 197
 - corenc::Mesh::CTriangleBasis, 208
 - corenc::Mesh::CTriangleLagrangeBasis, 212
 - corenc::Mesh::CTriangleLinearBasis, 220
- GetShapeFunctions

- corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [111](#)
- corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [120](#)
- corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [95](#)
- corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [103](#)
- GetSize
 - Algebra::Matrix, [297](#)
 - Algebra::MatrixDiag, [299](#)
 - Algebra::MatrixSkyline, [302](#)
- GetSolution
 - Algebra::ESolver, [258](#)
 - corenc::method::CDGMethod< Type >, [37](#)
 - corenc::method::CDGMethodZero< Type >, [38](#)
 - corenc::method::CFEMethod< Type >, [83](#)
 - corenc::method::CFEMethodZero< Type >, [85](#)
 - corenc::method::DGMethod< Problem, Grid, Matrix >, [243](#)
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, [249](#), [250](#)
 - corenc::method::FEMethod< Problem, Grid, Matrix >, [271](#)
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, [277](#), [278](#)
 - corenc::method::FVMMethod1d, [281](#)
 - corenc::method::system_dg_method< Grid, bool, bool >, [322](#)
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, [320](#), [321](#)
- getSolution
 - corenc::Mesh::CMesh< bool >, [137](#)
 - corenc::Mesh::CMesh< T >, [127](#)
 - corenc::Mesh::CMesh1D, [133](#)
 - corenc::Mesh::CRegularMesh, [178](#)
 - corenc::Mesh::CRegularMesh3D, [185](#)
 - corenc::Mesh::CTriangularMesh, [225](#), [226](#)
 - corenc::Mesh::CTriangularMeshLinear, [232](#)
- getTerm
 - corenc::CBurgersScalar, [26](#)
 - corenc::CDiffusionScalar, [43](#)
 - corenc::CProblem, [148](#)
 - corenc::CShallowWater, [190](#)
- GetType
 - corenc::Mesh::CElement< bool >, [80](#)
 - corenc::Mesh::CElement< T >, [66](#)
 - corenc::Mesh::CElement2D< bool >, [76](#)
 - corenc::Mesh::CElement2D< T >, [72](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [112](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [120](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [95](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [103](#)
 - corenc::Mesh::CNodeBasis, [144](#)
 - corenc::Mesh::CRectangleBasis, [156](#)
 - corenc::Mesh::CRectangleBasis2, [159](#)
 - corenc::Mesh::CRectangleBasis2x, [163](#)
 - corenc::Mesh::CRectangleBasis2y, [166](#)
 - corenc::Mesh::CRectangleHBasis, [173](#)
 - corenc::Mesh::CTriangleBasis, [208](#), [209](#)
 - corenc::Mesh::CTriangleLagrangeBasis, [212](#)
- getWeight
 - corenc::Mesh::CCubeBasis, [35](#)
 - corenc::Mesh::CEdge2ndBasis, [51](#)
 - corenc::Mesh::CEdgeConstantBasis, [54](#)
 - corenc::Mesh::CEdgeHermiteBasis, [57](#)
 - corenc::Mesh::CEdgeLinearBasis, [60](#)
 - corenc::Mesh::CEdgeMultiBasis, [63](#)
 - corenc::Mesh::CElement< bool >, [80](#)
 - corenc::Mesh::CElement< T >, [67](#)
 - corenc::Mesh::CElement2D< bool >, [76](#)
 - corenc::Mesh::CElement2D< T >, [72](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [112](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [120](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [95](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [103](#)
 - corenc::Mesh::CNodeBasis, [144](#)
 - corenc::Mesh::CRectangleBasis, [156](#)
 - corenc::Mesh::CRectangleBasis2, [159](#)
 - corenc::Mesh::CRectangleBasis2x, [163](#)
 - corenc::Mesh::CRectangleBasis2y, [166](#)
 - corenc::Mesh::CRectangleHBasis, [173](#)
 - corenc::Mesh::CTriangleBasis, [209](#)
 - corenc::Mesh::CTriangleLagrangeBasis, [212](#)
- getWeights
 - corenc::CFEweights, [89](#)
 - corenc::method::DGSolution< Grid >, [254](#)
 - corenc::method::STSolution< Grid >, [317](#)
- getWeights
 - corenc::method::DGSolution< Grid >, [254](#)
 - corenc::method::STSolution< Grid >, [317](#)
- global_matrix
 - corenc::test_case_elliptic_fem, [324](#)
- GMRES
 - Algebra, [13](#)
 - Algebra::ESolver, [259](#)
- GMRES_BiCGStab

- Algebra, [13](#)
- gpexp
 - corenc::GaussianKernel, [285](#)
- gpstep
 - corenc::GaussianKernel, [285](#)
- GREEN
 - corenc::color, [18](#)
- He
 - corenc::GaussianProcess, [286](#)
- homotopy_conv_diff_fem
 - corenc::test_case_elliptic_fem, [325](#)
- IBaumannOden
 - corenc::method, [22](#)
- IBaumannOdenIP
 - corenc::method, [22](#)
- ICentral
 - corenc::method, [22](#)
- IDUDV
 - corenc, [15](#)
- IDUV
 - corenc, [15](#)
- IIP
 - corenc::method, [22](#)
- ILaxFriedrichs
 - corenc::method, [22](#)
- IncreaseOrder
 - corenc::Mesh::CCube, [30](#)
 - corenc::Mesh::CCubeBasis, [35](#)
 - corenc::Mesh::CEdge, [47](#)
 - corenc::Mesh::CEdge2ndBasis, [51](#)
 - corenc::Mesh::CEdgeConstantBasis, [54](#)
 - corenc::Mesh::CEdgeHermiteBasis, [57](#)
 - corenc::Mesh::CEdgeLinearBasis, [60](#)
 - corenc::Mesh::CEdgeMultiBasis, [63](#)
 - corenc::Mesh::CElement< bool >, [80](#)
 - corenc::Mesh::CElement< T >, [67](#)
 - corenc::Mesh::CElement2D< bool >, [76](#)
 - corenc::Mesh::CElement2D< T >, [72](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [112](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [120](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [95](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [104](#)
 - corenc::Mesh::CNode, [140](#)
 - corenc::Mesh::CNodeBasis, [144](#)
 - corenc::Mesh::CRectangle, [152](#)
 - corenc::Mesh::CRectangleBasis, [156](#)
 - corenc::Mesh::CRectangleBasis2, [160](#)
 - corenc::Mesh::CRectangleBasis2x, [163](#)
 - corenc::Mesh::CRectangleBasis2y, [166](#)
 - corenc::Mesh::CRectangleConstantBasis, [169](#)
 - corenc::Mesh::CRectangleHBasis, [173](#)
 - corenc::Mesh::CTriangle, [203](#)
 - corenc::Mesh::CTriangleBasis, [209](#)
 - corenc::Mesh::CTriangleLagrangeBasis, [213](#)
 - corenc::Mesh::CTriangleLinear, [216](#)
 - corenc::Mesh::CTriangleLinearBasis, [221](#)
- INIPG
 - corenc::method, [22](#)
- Integrate
 - corenc::Mesh::CCube, [30](#), [31](#)
 - corenc::Mesh::CEdge, [47](#)
 - corenc::Mesh::CElement< bool >, [81](#)
 - corenc::Mesh::CElement< T >, [67](#)
 - corenc::Mesh::CElement2D< bool >, [77](#)
 - corenc::Mesh::CElement2D< T >, [72](#), [73](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [112](#), [113](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [120](#), [121](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [95](#), [96](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [104](#)
 - corenc::Mesh::CNode, [140](#)
 - corenc::Mesh::CRectangle, [152](#)
 - corenc::Mesh::CShape, [194](#)
 - corenc::Mesh::CTriangle, [203](#)
 - corenc::Mesh::CTriangleLinear, [217](#)
- interpolate
 - corenc::Mesh::CRegularMesh, [178](#)
 - corenc::Mesh::CRegularMesh3D, [185](#)
 - corenc::Mesh::CTriangularMesh, [226](#)
- Interval
 - corenc::Mesh, [20](#)
- IUDV
 - corenc, [15](#)
- IUpwind
 - corenc::method, [22](#)
- IUV
 - corenc, [15](#)
- Jacobian
 - corenc::Mesh::Point, [310](#)
- K
 - corenc::GaussianProcess, [287](#)
- kekus
 - test_case_elliptic_fem.cpp, [518](#)
- l
 - corenc::GaussianProcess, [287](#)
- L2Norm
 - corenc::method::FEAnalysis< Method1, Method2, Mesh1, Mesh2 >, [261](#)
- lambda
 - corenc::GaussianProcess, [288](#)
- LAST
 - corenc::Mesh, [21](#)
- LaxFriedrichs
 - corenc::method, [22](#)
- load_parameters
 - corenc::CBurgersScalar, [26](#)

- corenc::CDiffusionScalar, 44
- corenc::CProblem, 148
- corenc::CShallowWater, 190
- LoadSolution
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 244
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 250
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 272
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 278
- m_a
 - corenc::Mesh::Gauss1dim, 282
 - corenc::Mesh::Gauss1dimN< N >, 283
 - corenc::Mesh::GaussRectangular, 288
 - corenc::Mesh::GaussRectangularCubic, 290
- m_b
 - corenc::Mesh::GaussRectangular, 288
 - corenc::Mesh::GaussRectangularCubic, 290
- m_c
 - corenc::Mesh::GaussRectangular, 289
 - corenc::Mesh::GaussRectangularCubic, 290
- m_la
 - corenc::Mesh::GaussTetrahedron, 292
- m_lb
 - corenc::Mesh::GaussTetrahedron, 292
- m_lc
 - corenc::Mesh::GaussTetrahedron, 293
- m_ld
 - corenc::Mesh::GaussTetrahedron, 293
- m_msq
 - corenc::Mesh::GaussTetrahedron, 293
- m_order
 - corenc::Mesh::Gauss1dim, 282
 - corenc::Mesh::Gauss1dimN< N >, 284
 - corenc::Mesh::GaussTriangle, 294
- m_psq
 - corenc::Mesh::GaussTetrahedron, 293
- m_ra
 - corenc::Mesh::GaussRectangular, 289
 - corenc::Mesh::GaussRectangularCubic, 291
- m_rb
 - corenc::Mesh::GaussRectangular, 289
 - corenc::Mesh::GaussRectangularCubic, 291
- m_rc
 - corenc::Mesh::GaussRectangularCubic, 291
- m_rw
 - corenc::Mesh::GaussRectangular, 289
 - corenc::Mesh::GaussRectangularCubic, 291
- m_s
 - corenc::Mesh::GaussRectangularCubic, 291
- m_sqrt15
 - corenc::Mesh::GaussTriangle, 294
- m_sqrt35
 - corenc::Mesh::Gauss1dim, 283
- m_tra
 - corenc::Mesh::GaussTriangle, 294
- m_trb
 - corenc::Mesh::GaussTriangle, 294
- m_trw
 - corenc::Mesh::GaussTriangle, 294
- m_w
 - corenc::CVecSolution, 234
 - corenc::Mesh::Gauss1dim, 283
 - corenc::Mesh::Gauss1dimN< N >, 284
 - corenc::Mesh::GaussTetrahedron, 293
- m_w1
 - corenc::Mesh::GaussRectangularCubic, 291
- m_w2
 - corenc::Mesh::GaussRectangularCubic, 291
- m_w3
 - corenc::Mesh::GaussRectangularCubic, 291
- m_w4
 - corenc::Mesh::GaussRectangularCubic, 292
- m_wa
 - corenc::Mesh::GaussRectangular, 289
- m_wb
 - corenc::Mesh::GaussRectangular, 289
- m_wc
 - corenc::Mesh::GaussRectangular, 289
- MAGENTA
 - corenc::color, 18
- MAIN
 - corenc::method, 22
- main
 - main.cpp, 493
- main.cpp, 493
 - main, 493
- MASS
 - corenc, 15
- mass_matrix
 - corenc::tests::test_case_rectanglebasis, 326
 - corenc::tests::test_case_trianglebasis, 329
- mass_matrix_3rd_order
 - corenc::test_case_elliptic_fem, 325
- mass_matrix_4th_order
 - corenc::test_case_elliptic_fem, 325
- Matrix
 - Algebra::Matrix, 295, 296
- MatrixDiag
 - Algebra::MatrixDiag, 298, 299
- MatrixprodVector
 - Algebra::ESolver, 259
- MatrixSkyline
 - Algebra::MatrixSkyline, 301
- MatrixSkyline.cpp
 - _NOPE_, 336
 - N_MIN, 337
- Mesh1D
 - corenc::Mesh, 20
- Meshes
 - corenc::Mesh, 20
- Methods, 23
- Methods::CSMethod, 198
 - ~CSMethod, 198

- CSMethod, 198
- mid_point
 - Triangle.cpp, 379
 - wtf, 23
- multi_vector
 - corenc::multi_vector< T >, 304
- multi_vector.h
 - CORENC_MULTI_VECTOR_H_, 486
- N
 - corenc::GaussianKernel, 285
- N_MIN
 - MatrixSkyline.cpp, 337
- NODES
 - corenc::Mesh, 21
- NOFLUX
 - corenc::method, 22
- NullMatrix
 - Algebra::Matrix, 297
 - Algebra::MatrixDiag, 299
 - Algebra::MatrixSkyline, 302
- NullRow
 - Algebra::Matrix, 297
 - Algebra::MatrixDiag, 300
 - Algebra::MatrixSkyline, 303
- operator double
 - corenc::CFESolution, 86
- operator!=
 - corenc::CFESolution, 86
 - corenc::Mesh::Point, 311
- operator<
 - corenc::Mesh::Point, 311
- operator>>
 - corenc::Mesh::CCube, 31
 - corenc::Mesh::CEdge, 48
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 114
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 123
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 98
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 106
 - corenc::Mesh::CNode, 141
 - corenc::Mesh::CRectangle, 153
 - corenc::Mesh::CTriangle, 204
 - corenc::Mesh::CTriangleLinear, 217
- operator*
 - corenc::CFESolution, 88
 - corenc::Mesh::Point, 310–312
- operator*=
 - corenc::CFESolution, 87
 - corenc::Mesh::Point, 310
- operator()
 - Algebra::Matrix, 297
 - Algebra::MatrixDiag, 300
 - Algebra::MatrixSkyline, 303
- operator+
 - corenc::CFESolution, 88
 - corenc::Mesh::Point, 312
- operator+=
 - corenc::CFESolution, 87
 - corenc::Mesh::Point, 311
- operator-
 - corenc::CFESolution, 88
 - corenc::Mesh::Point, 312
- operator-=
 - corenc::CFESolution, 87
- operator/
 - corenc::CFESolution, 88
- operator/=
 - corenc::CFESolution, 87
- operator=
 - Algebra::Matrix, 297
 - Algebra::MatrixDiag, 300
 - Algebra::MatrixSkyline, 303
 - corenc::CFESolution, 87
 - corenc::Mesh::CCube, 31
 - corenc::Mesh::CCubeBasis, 35
 - corenc::Mesh::CEdge, 48
 - corenc::Mesh::CEdge2ndBasis, 51
 - corenc::Mesh::CEdgeConstantBasis, 54
 - corenc::Mesh::CEdgeHermiteBasis, 57
 - corenc::Mesh::CEdgeLinearBasis, 60
 - corenc::Mesh::CEdgeMultiBasis, 63
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 113
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 121
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 96
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 104
 - corenc::Mesh::CMesh1D, 133
 - corenc::Mesh::CNode, 141
 - corenc::Mesh::CNodeBasis, 144
 - corenc::Mesh::CRectangle, 152
 - corenc::Mesh::CRectangleBasis, 157
 - corenc::Mesh::CRectangleBasis2, 160
 - corenc::Mesh::CRectangleBasis2x, 163
 - corenc::Mesh::CRectangleBasis2y, 166
 - corenc::Mesh::CRectangleConstantBasis, 169
 - corenc::Mesh::CRectangleHBasis, 173
 - corenc::Mesh::CRegularMesh, 178
 - corenc::Mesh::CRegularMesh3D, 185
 - corenc::Mesh::CTriangle, 203
 - corenc::Mesh::CTriangleBasis, 209
 - corenc::Mesh::CTriangleLagrangeBasis, 213
 - corenc::Mesh::CTriangleLinear, 217
 - corenc::Mesh::CTriangleLinearBasis, 221
 - corenc::Mesh::CTriangularMesh, 226
 - corenc::Mesh::Point, 311
 - corenc::Mesh::point_source< T >, 314
 - corenc::method::DGSolution< Grid >, 254
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 272

- corenc::method::STSolution< Grid >, 318
- operator==
 - corenc::CFESolution, 87
 - corenc::Mesh::CCube, 31
 - corenc::Mesh::CEdge, 48
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 114
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 123
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 98
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 106
 - corenc::Mesh::CNode, 141
 - corenc::Mesh::CRectangle, 152
 - corenc::Mesh::CTriangle, 203
 - corenc::Mesh::CTriangleLinear, 217
 - corenc::Mesh::Point, 311
- OutDatFormat
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 244
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 251
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 272
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 279
- OutMeshFormat
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 244
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 251
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 272
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 279
- OutMeshTimeFormat
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 244
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 251
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 272
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 279
- parameter
 - corenc::Mesh::parameter< T >, 307, 308
- Parameter.h
 - CORENC_MESH_PARAMETER_H_, 489
- Parameters
 - corenc, 15
- PARDISO
 - Algebra, 13
- Pardiso
 - Algebra::ESolver, 260
- perform
 - corenc::test_cases, 330
- phi
 - corenc::GaussianProcess, 286
- Point
 - corenc::Mesh::Point, 310
- Point.h
 - CORENC_MESH_Point_h, 491
- point_source
 - corenc::Mesh::point_source< T >, 313
- Problems.h
 - CORENC_PROBLEMS_PROBLEMS_H_, 497
- Problems/BurgersScalar.cpp, 494
- Problems/BurgersScalar.h, 494
- Problems/DiffusionScalar.cpp, 495
- Problems/DiffusionScalar.h, 495
- Problems/Problems.h, 496, 497
- Problems/ShallowWater.cpp, 497
- Problems/ShallowWater.h, 497, 498
- ProjectSolution
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 245
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 251
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 273
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 279
- PURPLE
 - corenc::color, 18
- rayleigh
 - corenc::solvers::eigen_solver< Matrix, Solver >, 255
- Rectangle
 - corenc::Mesh, 20
- RED
 - corenc::color, 18
- Rediscretization
 - corenc::method::DGMethod< Problem, Grid, Matrix >, 245
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, 252
 - corenc::method::FEMethod< Problem, Grid, Matrix >, 273
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, 280
- refine_h
 - corenc::Mesh::CRegularMesh, 179
 - corenc::Mesh::CRegularMesh3D, 185
 - corenc::Mesh::CTriangularMesh, 226
 - corenc::Mesh::CTriangularMeshLinear, 232
- refine_hp
 - corenc::Mesh::CRegularMesh, 179
 - corenc::Mesh::CRegularMesh3D, 185
 - corenc::Mesh::CTriangularMesh, 226
- refine_hx
 - corenc::Mesh::CRegularMesh, 179
 - corenc::Mesh::CRegularMesh3D, 185
- refine_hy
 - corenc::Mesh::CRegularMesh, 179
 - corenc::Mesh::CRegularMesh3D, 186

- refine_p
 - corenc::Mesh::CRegularMesh, 179
 - corenc::Mesh::CRegularMesh3D, 186
 - corenc::Mesh::CTriangularMesh, 226
- RegularMesh.cpp
 - sort_indexes, 390
- RegularMesh3D.cpp
 - sort_indexes, 393
- Reload
 - Algebra::ESolver, 260
- removeTerm
 - corenc::CBurgersScalar, 26
 - corenc::CDiffusionScalar, 44
 - corenc::CShallowWater, 191
- resize
 - corenc::multi_vector< T >, 305
- ReverseNormal
 - corenc::Mesh::CCubeBasis, 35
 - corenc::Mesh::CEdge2ndBasis, 51
 - corenc::Mesh::CEdgeConstantBasis, 54
 - corenc::Mesh::CEdgeHermiteBasis, 57
 - corenc::Mesh::CEdgeLinearBasis, 60
 - corenc::Mesh::CEdgeMultiBasis, 63
 - corenc::Mesh::CElement< bool >, 81
 - corenc::Mesh::CElement< T >, 68
 - corenc::Mesh::CElement2D< bool >, 77
 - corenc::Mesh::CElement2D< T >, 73
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 113
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 121
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 96
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 105
 - corenc::Mesh::CNodeBasis, 144
 - corenc::Mesh::CRectangleBasis, 157
 - corenc::Mesh::CRectangleBasis2, 160
 - corenc::Mesh::CRectangleBasis2x, 163
 - corenc::Mesh::CRectangleBasis2y, 166
 - corenc::Mesh::CRectangleConstantBasis, 170
 - corenc::Mesh::CRectangleHBasis, 173
 - corenc::Mesh::CShapeFunction< Type >, 197
 - corenc::Mesh::CTriangleBasis, 209
 - corenc::Mesh::CTriangleLagrangeBasis, 213
 - corenc::Mesh::CTriangleLinearBasis, 221
- RungeKutta
 - corenc::method::RungeKutta< Problem, Type >, 315
- RUV
 - corenc, 15
- S
 - corenc::solvers::vector_solution, 332
- s_point
 - Triangle.cpp, 379
 - wtf, 24
- scalar_func
 - corenc, 15
- SECOND
 - corenc::method, 22
- set
 - corenc::Mesh::parameter< T >, 309
 - corenc::multi_vector< T >, 305
- set2ndOrder
 - corenc::Mesh::CTriangularMesh, 226
- set3rdOrder
 - corenc::Mesh::CTriangularMesh, 227
- set4thOrder
 - corenc::Mesh::CTriangularMesh, 227
- set_boundary_parameter
 - corenc::CDiffusionScalar, 44
 - corenc::CShallowWater, 191
- set_parameter
 - corenc::CDiffusionScalar, 44
 - corenc::CShallowWater, 191
- set_point_source
 - corenc::CDiffusionScalar, 44
- SetDoF
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 121
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 96
- SetEdge
 - corenc::Mesh::CCube, 31
 - corenc::Mesh::CRectangle, 153
 - corenc::Mesh::CShape, 194
 - corenc::Mesh::CTriangle, 204
 - corenc::Mesh::CTriangleLinear, 218
- SetFacet
 - corenc::Mesh::CCube, 32
 - corenc::Mesh::CRectangle, 153
 - corenc::Mesh::CShape, 195
 - corenc::Mesh::CTriangle, 204
 - corenc::Mesh::CTriangleLinear, 218
- SetNeighbour
 - corenc::Mesh::CElement< bool >, 81
 - corenc::Mesh::CElement< T >, 68
 - corenc::Mesh::CElement2D< bool >, 77
 - corenc::Mesh::CElement2D< T >, 73
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 113
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, 122
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, 97
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, 105
- SetNode
 - corenc::Mesh::CCube, 32
 - corenc::Mesh::CEdge, 48
 - corenc::Mesh::CElement< bool >, 81
 - corenc::Mesh::CElement< T >, 68
 - corenc::Mesh::CElement2D< bool >, 77
 - corenc::Mesh::CElement2D< T >, 73
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, 113

- corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [122](#)
- corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [97](#)
- corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [105](#)
- corenc::Mesh::CNode, [141](#)
- corenc::Mesh::CRectangle, [153](#)
- corenc::Mesh::CShape, [195](#)
- corenc::Mesh::CTriangle, [204](#)
- corenc::Mesh::CTriangleLinear, [218](#)
- SetOrder
 - corenc::Mesh::CCube, [32](#)
 - corenc::Mesh::CElement2D< bool >, [78](#)
 - corenc::Mesh::CElement2D< T >, [73](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [105](#)
 - corenc::Mesh::CRectangle, [153](#)
 - corenc::Mesh::CRectangleHBasis, [173](#)
- setParameter
 - corenc::Mesh::CMesh< bool >, [137](#)
 - corenc::Mesh::CMesh< T >, [127](#)
 - corenc::Mesh::CMesh1D, [133](#)
 - corenc::Mesh::CRegularMesh, [179](#)
 - corenc::Mesh::CRegularMesh3D, [186](#)
 - corenc::Mesh::CTriangularMesh, [227](#)
 - corenc::Mesh::CTriangularMeshLinear, [232](#)
- SetShape
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [114](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [122](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [97](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [105](#)
- SetShapeFunction
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [114](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [122](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [97](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [106](#)
- SetSolution
 - corenc::method::DGMethod< Problem, Grid, Matrix >, [245](#)
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, [252](#)
 - corenc::method::FEMethod< Problem, Grid, Matrix >, [273](#)
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, [280](#)
- setTerm
 - corenc::CBurgersScalar, [27](#)
 - corenc::CDiffusionScalar, [45](#)
 - corenc::CProblem, [148](#)
 - corenc::CShallowWater, [191](#)
- SetTimeStep
 - corenc::method::DGMethod< Problem, Grid, Matrix >, [246](#)
 - corenc::method::DGMethodZero< Problem, Grid, Matrix >, [252](#)
 - corenc::method::FEMethod< Problem, Grid, Matrix >, [274](#)
 - corenc::method::FEMethodZero< Problem, Grid, Matrix >, [280](#)
- SetType
 - corenc::Mesh::CElement< bool >, [82](#)
 - corenc::Mesh::CElement< T >, [68](#)
 - corenc::Mesh::CElement2D< bool >, [78](#)
 - corenc::Mesh::CElement2D< T >, [74](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, bool, bool >, [114](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, bool >, [122](#)
 - corenc::Mesh::CFiniteElement< Shape, Shape-Function, DoF, T >, [97](#)
 - corenc::Mesh::CFiniteElement2D< Shape, Shape-Function >, [106](#)
- sigma2
 - corenc::GaussianProcess, [288](#)
- size
 - corenc::multi_vector< T >, [306](#)
- Solve
 - Algebra::ESolver, [260](#)
 - corenc::CFiniteSolver< Method, Mesh, Solver >, [124](#)
 - corenc::method::FVMMethod1d, [281](#)
- solve
 - corenc::solvers::dg_shallow_water< Mesh >, [235](#)
 - corenc::solvers::dg_solver_shallow_water, [239](#)
- solver
 - test_case_solver.cpp, [521](#)
- Solvers
 - Algebra, [13](#)
 - Solvers/dg_solver.h, [498](#), [499](#)
 - Solvers/dg_solver_shallow_water.cpp, [500](#)
 - Solvers/dg_solver_shallow_water.h, [500](#), [501](#)
 - Solvers/eigen_solver.h, [508](#)
 - Solvers/fem_solver.h, [509](#)
 - Solvers/fem_solver_lib.h, [512](#)
- sort_indexes
 - RegularMesh.cpp, [390](#)
 - RegularMesh3D.cpp, [393](#)
 - TriangularMesh.cpp, [395](#)
- strees_matrix_3rd_order
 - corenc::test_case_elliptic_fem, [325](#)
- stress_matrix
 - corenc::tests::test_case_rectanglebasis, [326](#)
 - corenc::tests::test_case_trianglebasis, [329](#)
- stress_matrix_4th_order
 - corenc::test_case_elliptic_fem, [325](#)
- STSolution
 - corenc::method::STSolution< Grid >, [316](#), [317](#)

- SUPG
 - corenc, [15](#)
- system_dg_method
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, [319](#)
- system_dg_method.h
 - CORENC_METHODS_SYSTEM_DG_METHOD_H_, [THIRD](#) [479](#)
- Terms
 - corenc, [15](#)
- test_case_elliptic_fem
 - corenc::test_case_elliptic_fem, [323](#)
- test_case_elliptic_fem.cpp
 - _USE_MATH_DEFINES, [518](#)
 - kekus, [518](#)
- test_case_rectanglebasis
 - corenc::tests::test_case_rectanglebasis, [326](#)
- test_case_rectanglebasis.h
 - CORENC_TEST_CASE_RECTANGLEBASIS_H_, [516](#)
- test_case_regular_mesh
 - corenc::tests::test_case_regular_mesh, [327](#)
- test_case_regular_mesh.h
 - CORENC_TEST_CASE_REGULAR_MESH_H_, [520](#)
- test_case_solver
 - corenc::test_case_solver, [328](#)
- test_case_solver.cpp
 - _USE_MATH_DEFINES, [521](#)
 - solver, [521](#)
- test_case_trianglebasis
 - corenc::tests::test_case_trianglebasis, [328](#)
- test_case_trianglebasis.h
 - CORENC_TEST_CASE_TRIANGLEBASIS_H_, [517](#)
- test_cases
 - corenc::test_cases, [329](#)
- test_cases.h
 - CORENC_TEST_CASES_H_, [522](#)
- test_conv_diff
 - corenc::test_conv_diff, [331](#)
- test_conv_diff.cpp
 - _USE_MATH_DEFINES, [523](#)
- Tests/FiniteElements/test_case_rectanglebasis.cpp, [515](#)
- Tests/FiniteElements/test_case_rectanglebasis.h, [515](#), [516](#)
- Tests/FiniteElements/test_case_trianglebasis.cpp, [516](#)
- Tests/FiniteElements/test_case_trianglebasis.h, [516](#), [517](#)
- Tests/test_case_elliptic_fem.cpp, [517](#)
- Tests/test_case_elliptic_fem.h, [518](#), [519](#)
- Tests/test_case_regular_mesh.cpp, [519](#)
- Tests/test_case_regular_mesh.h, [519](#), [520](#)
- Tests/test_case_solver.cpp, [520](#)
- Tests/test_case_solver.h, [521](#)
- Tests/test_cases.cpp, [522](#)
- Tests/test_cases.h, [522](#), [523](#)
- Tests/test_conv_diff.cpp, [523](#)
- Tests/test_conv_diff.h, [524](#)
- TetrahedralMesh
 - corenc::Mesh, [20](#)
- Tetrahedron
 - corenc::Mesh, [20](#)
 - corenc::method, [22](#)
- toDGSolution
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, [321](#)
- totalsize
 - corenc::multi_vector< T >, [306](#)
- transpose_sky
 - Algebra::MatrixSkyline, [303](#)
- Triangle
 - corenc::Mesh, [20](#)
- Triangle.cpp
 - center_point, [379](#)
 - mid_point, [379](#)
 - s_point, [379](#)
- TriangularMesh
 - corenc::Mesh, [20](#)
- TriangularMesh.cpp
 - sort_indexes, [395](#)
- updateSolution
 - corenc::Mesh::CMesh< bool >, [137](#), [138](#)
 - corenc::Mesh::CMesh< T >, [127](#), [128](#)
 - corenc::Mesh::CMesh1D, [133](#), [134](#)
 - corenc::Mesh::CRegularMesh, [180](#)
 - corenc::Mesh::CRegularMesh3D, [186](#), [187](#)
 - corenc::Mesh::CTriangularMesh, [227](#), [228](#)
 - corenc::Mesh::CTriangularMeshLinear, [232](#), [233](#)
- updateTimestep
 - corenc::method::RungeKutta< Problem, Type >, [315](#)
- updateWeight
 - corenc::CFEweights, [89](#)
 - corenc::method::DGSolution< Grid >, [254](#)
 - corenc::method::STSolution< Grid >, [318](#)
- updateWeights
 - corenc::method::system_dg_method< Problem, Grid, Matrix >, [321](#)
- Upwind
 - corenc::method, [22](#)
- vector_func
 - corenc, [15](#)
- vector_solution
 - corenc::solvers::vector_solution, [332](#)
- WHITE
 - corenc::color, [18](#)
- wtf, [23](#)
 - center_point, [23](#)
 - mid_point, [23](#)
 - s_point, [24](#)

x

corenc::Mesh::Point, [312](#)

y

corenc::Mesh::Point, [312](#)

YELLOW

corenc::color, [18](#)

z

corenc::Mesh::Point, [313](#)