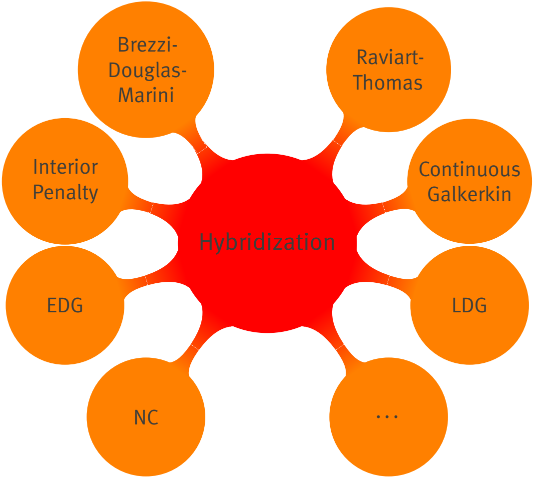
Westfälische
Wilhelms-Universität
Münster

Unified software framework for hybridized discretization methods

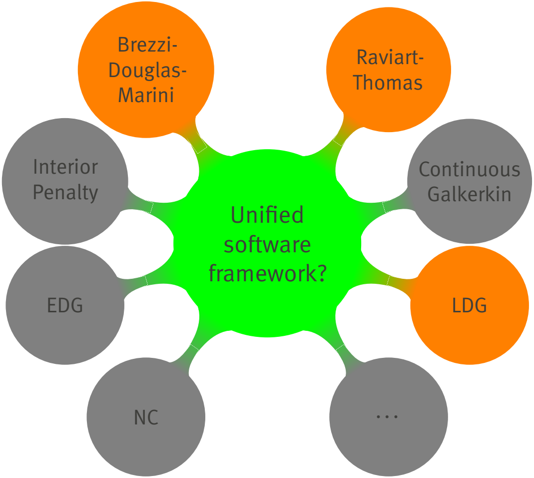
Stefan Girke

June 19th, 2012

Motivation [Cockburn2009]



Motivation [Cockburn2009]



Outline

Hybridization

Unifying framework: Implementation in DUNE

- DUNE-FEM-LOCALFUNCTIONS
- DUNE-HYBRIDFEM

Numerical tests

- Implementation
- Numerical results

Hybridization

Unifying framework: Implementation in DUNE

- DUNE-FEM-LOCALFUNCTIONS
- DUNE-HYBRIDFEM

Numerical tests

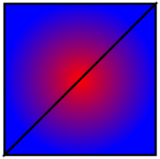
- Implementation
- Numerical results

Hybridization I

Let $u \in X$ the solution of

$$\mathcal{L}(u) = f \qquad \text{on } \Omega.$$

- $\Omega \subset \mathbb{R}^2$ domain,
- X Hilbert space,
- $\mathcal{L} : X \to X'$ partial differential operator,
- $f \in X'$ source term
- $\mathcal{T}_h$ triangulation of Ω,
- $\mathcal{E}_h$ edges, $\mathcal{E}_h^\circ$ inner edges

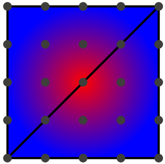


Hybridization II

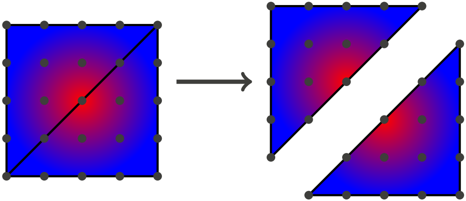
Let $u_h \in X_h$ be the solution of

$$\mathcal{L}_h(u_h) = f \qquad \text{on } \Omega.$$

- $X_h \subset X$ finite dimensional subspace,
- $\mathcal{L}_h : X_h \to X_h'$ discretization operator,

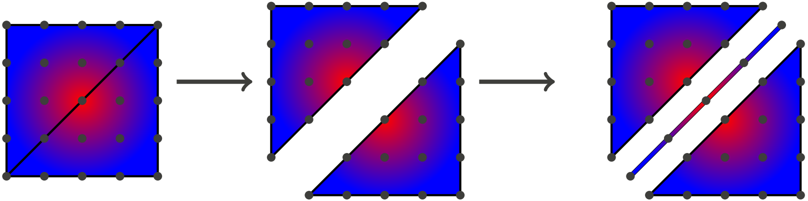


Hybridization III



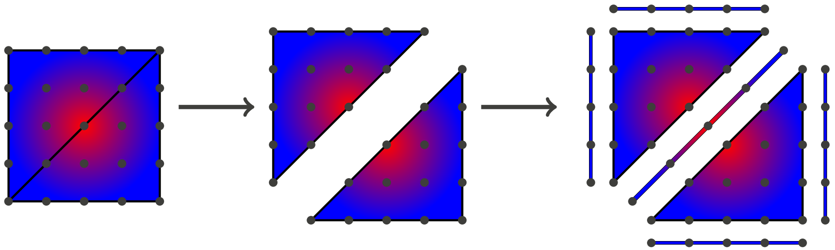
1. neglect continuity conditions ("transmission condition"),

Hybridization III



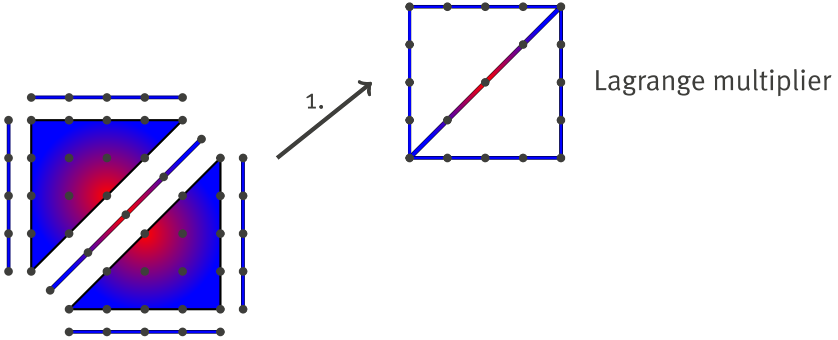
1. neglect continuity conditions ("transmission condition"),
2. reinforce them by new unkowns ("Lagrange multipliers")

Hybridization III



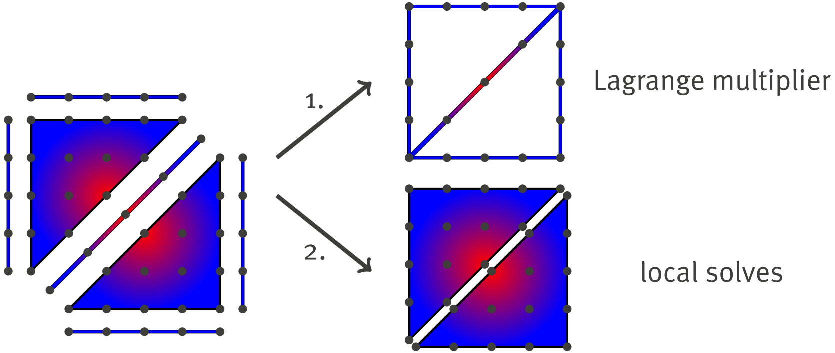
1. neglect continuity conditions ("transmission condition"),
2. reinforce them by new unkowns ("Lagrange multipliers")

Hybridization IV



1. solve global system for the Lagrange multipliers,

Hybridization IV



1. solve global system for the Lagrange multipliers,
2. solve local systems approximating the solution u_h element-wise

Hybridization V

To hybridize a discretization method "N" with operator $\mathcal{L}_h$ we need to:

1. define local solvers by using $\mathcal{L}_h|_K$ for each element $K \in \mathcal{T}_h$,

Hybridization V

To hybridize a discretization method "N" with operator $\mathcal{L}_h$ we need to:

1. define local solvers by using $\mathcal{L}_h|_K$ for each element $K \in \mathcal{T}_h$,
2. find a suitable transmission condition if possible.

Hybridization V

To hybridize a discretization method "N" with operator $\mathcal{L}_h$ we need to:

1. define local solvers by using $\mathcal{L}_h|_K$ for each element $K \in \mathcal{T}_h$,
2. find a suitable transmission condition if possible.

We call the hybridized method "N-Hybrid" or "N-H".

Hybridization V

To hybridize a discretization method "N" with operator $\mathcal{L}_h$ we need to:

1. define local solvers by using $\mathcal{L}_h|_K$ for each element $K \in \mathcal{T}_h$,
2. find a suitable transmission condition if possible.

We call the hybridized method "N-Hybrid" or "N-H".
Finally we need to check:

Hybridization V

To hybridize a discretization method "N" with operator $\mathcal{L}_h$ we need to:

1. define local solvers by using $\mathcal{L}_h|_K$ for each element $K \in \mathcal{T}_h$,
2. find a suitable transmission condition if possible.

We call the hybridized method "N-Hybrid" or "N-H".
Finally we need to check:

- Is the hybridized method equivalent to the non hybridized method, or is it a new method?

Advantages of hybridization

1. smaller (easier) global system,

Advantages of hybridization

1. smaller (easier) global system,
2. local solves can be computed element-wise,

Advantages of hybridization

1. smaller (easier) global system,
2. local solves can be computed element-wise,
3. more flexible, i.e.

Advantages of hybridization

1. smaller (easier) global system,
2. local solves can be computed element-wise,
3. more flexible, i.e.
   3.1 different local solvers on each element,

Advantages of hybridization

1. smaller (easier) global system,
2. local solves can be computed element-wise,
3. more flexible, i.e.
   3.1 different local solvers on each element,
   3.2 *p*-adaptivity…

Advantages of hybridization

1. smaller (easier) global system,
2. local solves can be computed element-wise,
3. more flexible, i.e.
   3.1 different local solvers on each element,
   3.2 *p*-adaptivity...
4. new methods can be obtained

Required issues for hybridization

- discrete function spaces for local solves,

Required issues for hybridization

- discrete function spaces for local solves,
- discrete function spaces for Lagrange multipliers, in particular spaces defined on codim-1-entities,

Required issues for hybridization

- discrete function spaces for local solves,
- discrete function spaces for Lagrange multipliers, in particular spaces defined on codim-1-entities,
- local assembler/solvers,

Required issues for hybridization

- discrete function spaces for local solves,
- discrete function spaces for Lagrange multipliers, in particular spaces defined on codim-1-entities,
- local assembler/solvers,
- assembler/solver for global Lagrange multiplier

Hybridization

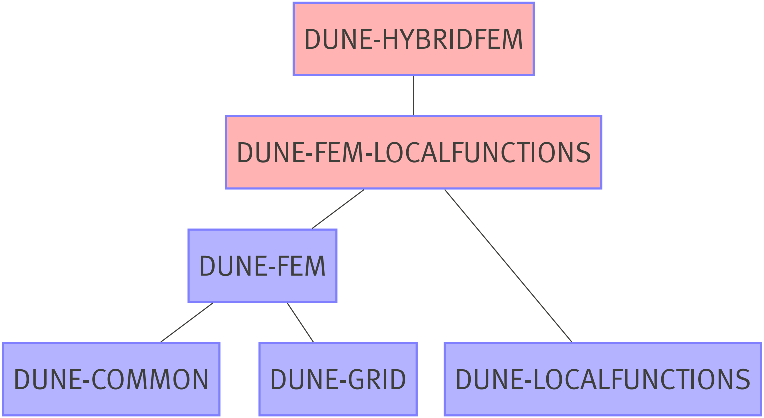
Unifying framework: Implementation in DUNE

- DUNE-FEM-LOCALFUNCTIONS
- DUNE-HYBRIDFEM

Numerical tests

- Implementation
- Numerical results

Implementation in DUNE

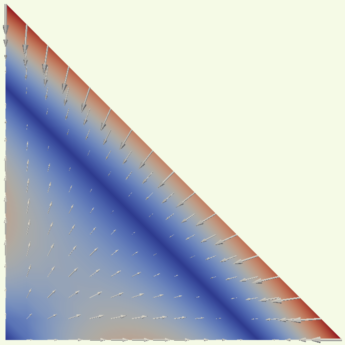


Dune
Distributed and Unified Numerics Environment

DUNE-FEM-LOCALFUNCTIONS - Motivation

1. want to use Raviart-Thomas finite elements in DUNE-FEM,

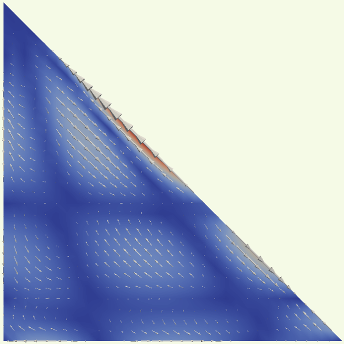
Example (first basis function for Raviart-Thomas, order 1)



DUNE-FEM-LOCALFUNCTIONS - Motivation

1. want to use Raviart-Thomas finite elements in DUNE-FEM,

Example (42nd basis function for Raviart-Thomas, order 5)



DUNE-FEM-LOCALFUNCTIONS - Motivation

1. want to use Raviart-Thomas finite elements in DUNE-FEM,
2. some implementation work in DUNE-FEM was already done, but not flexible enough,
3. want to have a discrete function space similar to DUNE-PDELAB.

DUNE-FEM-LOCALFUNCTIONS I

Discrete function spaces

1. are defined by finite elements from DUNE-LOCALFUNCTIONS,

DUNE-FEM-LOCALFUNCTIONS I

Discrete function spaces

1. are defined by finite elements from DUNE-LOCALFUNCTIONS,
2. fulfill the `DUNE::DiscreteFunctionSpaceInterface` from DUNE-FEM,

DUNE-FEM-LOCALFUNCTIONS I

Discrete function spaces

1. are defined by finite elements from DUNE-LOCALFUNCTIONS,
2. fulfill the `DUNE::DiscreteFunctionSpaceInterface` from DUNE-FEM,
3. can be either defined on codim-0-entities or intersections (codim-1-entities),

DUNE-FEM-LOCALFUNCTIONS I

Discrete function spaces

1. are defined by finite elements from DUNE-LOCALFUNCTIONS,
2. fulfill the `DUNE::DiscreteFunctionSpaceInterface` from DUNE-FEM,
3. can be either defined on codim-0-entities or intersections (codim-1-entities),
4. are mainly created through a "basis function set map" (next slide).

DUNE-FEM-LOCALFUNCTIONS I

Discrete function spaces

1. are defined by finite elements from DUNE-LOCALFUNCTIONS,
2. fulfill the `DUNE::DiscreteFunctionSpaceInterface` from DUNE-FEM,
3. can be either defined on codim-0-entities or intersections (codim-1-entities),
4. are mainly created through a "basis function set map" (next slide).

The finite element may be independent of geometry and polynomial order on each entity.

DUNE-FEM-LOCALFUNCTIONS I

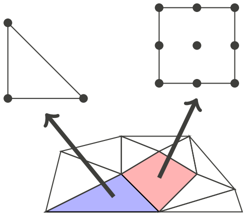
Discrete function spaces

1. are defined by finite elements from DUNE-LOCALFUNCTIONS,
2. fulfill the `DUNE::DiscreteFunctionSpaceInterface` from DUNE-FEM,
3. can be either defined on codim-0-entities or intersections (codim-1-entities),
4. are mainly created through a "basis function set map" (next slide).

The finite element may be independent of geometry and polynomial order on each entity.

- No *h*-adaptiv and parallel version implemented, yet.

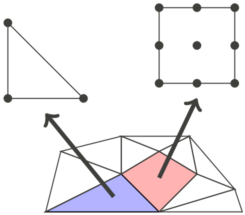
Basis function set maps



task: return correct basis function set depending on

1. polynomial degree,

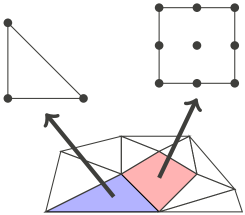
Basis function set maps



task: return correct basis function set depending on

1. polynomial degree,
2. orientation,

Basis function set maps



task: return correct basis function set depending on

1. polynomial degree,
2. orientation,
3. geometry…

Main challenges

1. fulfill interfaces in DUNE-FEM

Main challenges

1. fulfill interfaces in DUNE-FEM
2. intersections $\neq$ codim-1-entities

Main challenges

1. fulfill interfaces in DUNE-FEM
2. intersections $\neq$ codim-1-entities
3. no single basis functions in DUNE-LOCALFUNCTIONS

DUNE-FEM-LOCALFUNCTIONS II

Example (Lagrange finite element)

```
using namespace Dune;
using namespace Dune::FemLocalFunctions;

typedef LagrangeLocalFiniteElement< EquidistantPointSet,
                                    dimension,
                                    double,
                                    double >
  LagrangeElementType;
```

DUNE-FEM-LOCALFUNCTIONS III

Example (Linear Lagrange space)

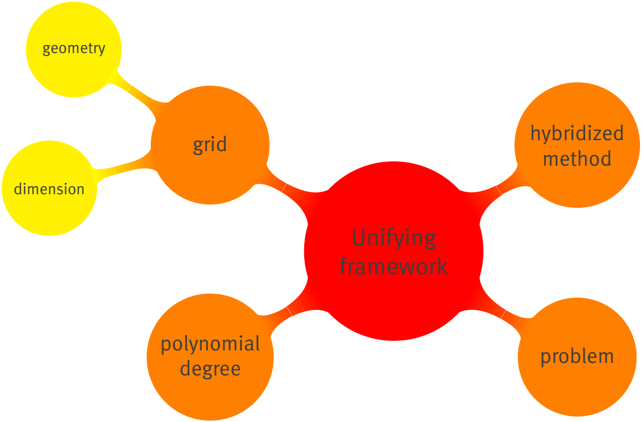
```
typedef LagrangeElementType
  FiniteElementType;

typedef BaseFunctionSetMap< GridPartType, FiniteElementType,
            NoTransformation, SimpleStorage, 1 >
  BaseFunctionSetMapType;

typedef DiscreteFunctionSpace< BaseFunctionSetMapType >
  DiscreteFunctionSpaceType;

BaseFunctionSetMapType      fem(gridPart);
DiscreteFunctionSpaceType   space(gridPart, fem);
```

DUNE-FEM-LOCALFUNCTIONS IV

Example (Linear Lagrange space on intersections)

```
typedef IntersectionLocalFiniteElement< LagrangeElementType >
  FiniteElementType;

typedef IntersectionBaseFunctionSetMap< GridPartType,
      FiniteElementType,NoTransformation,SimpleStorage,1>
  BaseFunctionSetMapType;

typedef DiscreteFunctionSpace< BaseFunctionSetMapType >
  DiscreteFunctionSpaceType;

typedef typename BaseFunctionSetMapType::GridPartType
  IntersectionGridPartType;

IntersectionGridPartType    iGridPart(grid);
BaseFunctionSetMapType      fem(iGridPart);
DiscreteFunctionSpaceType   space(iGridPart, fem);
```

DUNE-HYBRIDFEM - Concept



Hybridization

Unifying framework: Implementation in DUNE
DUNE-FEM-LOCALFUNCTIONS
DUNE-HYBRIDFEM

Numerical tests
Implementation
Numerical results

Implemented problem I

2nd order elliptic boundary value problem

Find $(\mathbf{q}, u) \in \mathbf{V} \times W$ fulfilling

$$\begin{aligned} \mathbf{q} + a\nabla u &= 0 && \text{on } \Omega, \\ \mathrm{div}\,\mathbf{q} + du &= f && \text{on } \Omega, \\ u &= g && \text{on } \partial\Omega. \end{aligned}$$

Implemented problem II

2nd order elliptic boundary value problem

Find $(\mathbf{q}_h, u_h) \in \mathbf{V}_h \times W_h$ fulfilling

$$\begin{aligned} \mathbf{q}_h + a\nabla u_h &= 0 && \text{on } \Omega, \\ \mathrm{div}\,\mathbf{q}_h + d u_h &= f && \text{on } \Omega, \\ u_h &= g && \text{on } \partial\Omega. \end{aligned}$$

Implemented problem III

First local solver

Find for each element $K \in \mathcal{T}_h$ and each function m defined on ∂K a solution $(\mathbf{q}_{h,m}, u_{h,m}) \in \mathbf{V}(K) \times W(K)$ satisfying

$$\int_K c\mathbf{q}_{h,m} \cdot \mathbf{v} - \int_K u_{h,m} \mathrm{div}\mathbf{v} = -\int_{\partial K} m\mathbf{v} \cdot n \quad \forall \mathbf{v} \in \mathbf{V}(K),$$

$$-\int_K \nabla w \cdot \mathbf{q}_{h,m} + \int_{\partial K} w\widehat{\mathbf{q}_{h,m}} \cdot n + \int_K d u_{h,m} w = 0 \quad \forall w \in W(K),$$

where $\widehat{\mathbf{q}_{h,m}}$ depends on the method.

Implemented problem IV

Second local solver

Find for each element $K \in \mathcal{T}_h$ a solution $(\mathbf{q}_{h,f}, u_{h,f}) \in \mathbf{V}(K) \times W(K)$ satisfying

$$\int_K c\mathbf{q}_{h,f} \cdot \mathbf{v} - \int_K u_{h,f} \operatorname{div} \mathbf{v} = 0 \qquad \forall \mathbf{v} \in \mathbf{V}(K),$$

$$-\int_K \nabla w \cdot \mathbf{q}_{h,f} + \int_{\partial K} w\widehat{\mathbf{q}_{h,f}} \cdot n + \int_K d u_{h,f} w = \int_K fw \qquad \forall w \in W(K),$$

where $\widehat{\mathbf{q}_{h,f}}$ depends on the method.

living.knowledge WWU Münster

Implemented problem V

Solver for Lagrange multiplier

Find $\lambda_h \in M_h$ satisfying

$$a_h(\lambda_h, \mu) = b_h(\mu) \qquad \forall \mu \in M_h,$$

where

$$a_h(\eta, \mu) = \sum_{K \in \mathcal{T}_h} \int_K (c\mathbf{q}_{h,\eta} \cdot \mathbf{q}_{h,\mu} + d u_{h,\eta} u_{h,\mu}) + \sum_{e \in \mathcal{E}_h} \int_e [(\widehat{\mathbf{q}_{h,\eta}} - \mathbf{q}_{h,\eta})(u_{h,\mu} - \mu)] \qquad \forall \eta, \mu \in M_h,$$

$$b_h(\mu) = \sum_{e \in \mathcal{E}_h} \int_e g_h [\widehat{\mathbf{q}_{h,\mu}}] + \sum_{K \in \mathcal{T}_h} \int_K f u_{h,\mu} \qquad \forall \mu \in M_h.$$

Implemented methods

	$\mathbf{V}(K)$	$W(K)$	M_h
RT-H	$\mathbb{P}_k(K)^n + x\mathbb{P}_k(K)$	$\mathbb{P}_k(K)$	$\mathbb{P}_k(\mathcal{E}_h^\circ)$
BDM-H	$\mathbb{P}_k(K)^n$	$\mathbb{P}_{k-1}(K)$	$\mathbb{P}_k(\mathcal{E}_h^\circ)$
LDG-H I	$\mathbb{P}_k(K)^n$	$\mathbb{P}_{k-1}(K)$	$\mathbb{P}_k(\mathcal{E}_h^\circ)$
LDG-H II	$\mathbb{P}_k(K)^n$	$\mathbb{P}_k(K)$	$\mathbb{P}_k(\mathcal{E}_h^\circ)$
LDG-H III	$\mathbb{P}_{k-1}(K)^n$	$\mathbb{P}_k(K)$	$\mathbb{P}_k(\mathcal{E}_h^\circ)$

	$\widehat{\mathbf{q}_{h,m}}$	$\widehat{\mathbf{q}_{h,f}}$
RT-H/BDM-H	$\mathbf{q}_{h,m}$	$\mathbf{q}_{h,f}$
LDG-H	$\mathbf{q}_{h,m} + \tau(u_{h,m} - m)n$	$\mathbf{q}_{h,f} + \tau(u_{h,f})n$

Example

```
Hybrid_Test< RT_HybridModel,
             AlbertaGridType,
             PoissonProblem,
             polOrder >( gridFile );
```

Raviart-Thomas-Hybrid, order 1

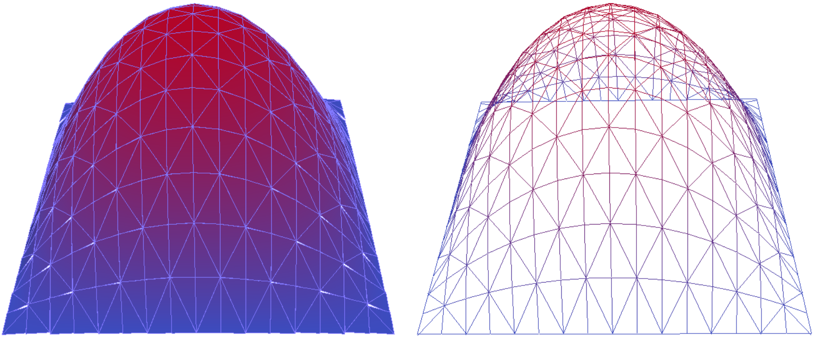


Figure: pressure/Lagrange multiplier

Brezzi-Douglas-Marini-Hybrid, order 1

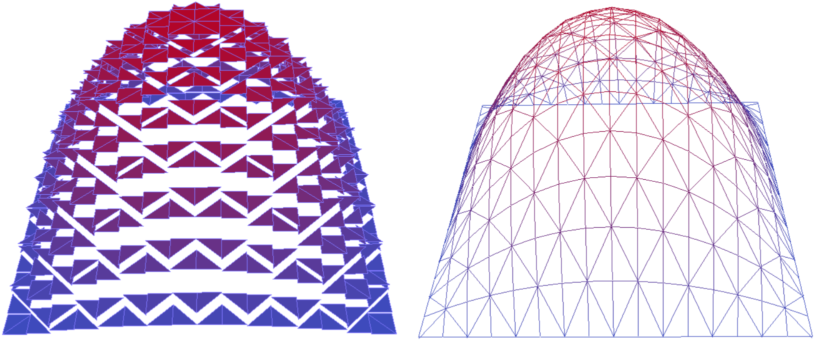


Figure: pressure/Lagrange multiplier

Comparison RT-H vs. BDM-H

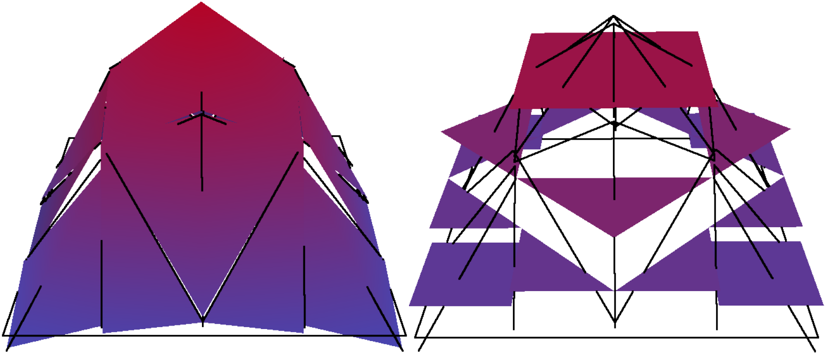


Figure: pressure/Lagrange multiplier

Raviart-Thomas-Hybrid, order 5, 8 elements

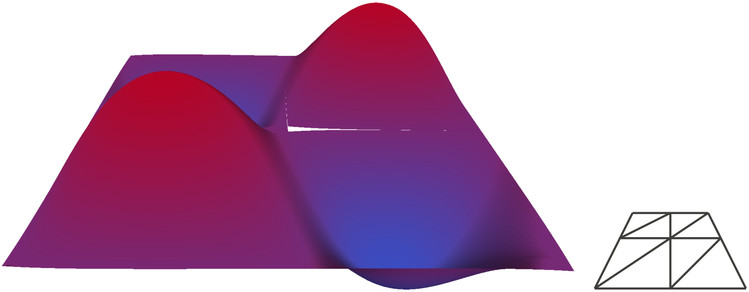
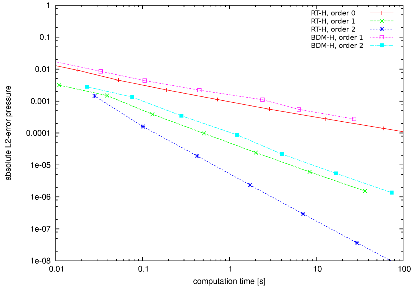
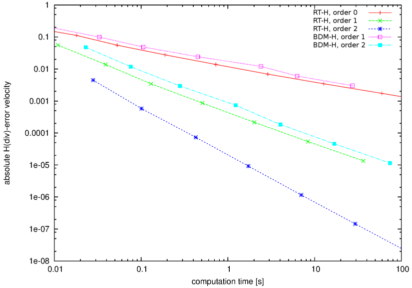


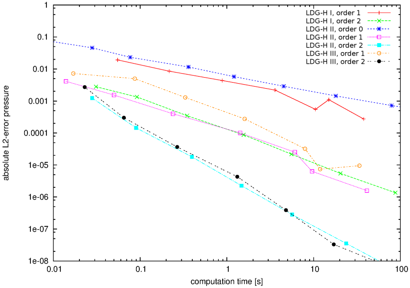
Figure: pressure/grid

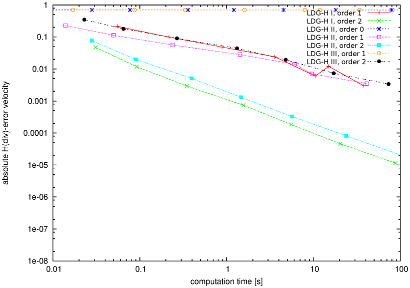
Error vs. computation time RT-H/BDM-H





Error vs. computation time LDG





Conclusion

- it is possible to implement a general unifying framework for hybridized methods,

Conclusion

- it is possible to implement a general unifying framework for hybridized methods,
- the hybridized Raviart-Thomas method turns out to have the best convergence rates,

Conclusion

- it is possible to implement a general unifying framework for hybridized methods,
- the hybridized Raviart-Thomas method turns out to have the best convergence rates,
- there are still many interesting things to implement,

Conclusion

- it is possible to implement a general unifying framework for hybridized methods,
- the hybridized Raviart-Thomas method turns out to have the best convergence rates,
- there are still many interesting things to implement,
- we have combined the flexibility of DUNE with the flexibility of hybridization

References

[Cockburn2009] B. Cockburn, J. Gopalakrishnan, and R. Lazarov,
Unified Hybridization of discontinuous Galerkin, mixed and continuous Galerkin methods for second order elliptic problems.
SIAM J. Numer. Anal 47(2):1319–1365, 2009

[Girke2012] S. Girke,
Vereinheitlichter Rahmen zur Implementierung hybridisierter Diskretisierungsverfahren.
Diploma Thesis, Fachbereich Mathematik und Informatik, Universität Münster, 2012

[Bastian2008] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander,
A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part II: Implementation and Tests in DUNE.
Computing, 82(2–3):121–138, 2008

[Dedner2010] A. Dedner, R. Klöfkorn, M. Nolte, and M. Ohlberger,
A Generic Interface for Parallel and Adaptive Scientific Computing: Abstraction Principles and the DUNE-FEM Module.
Computing, 90(3–4):165–196, 2010

[Bastian2010] P. Bastian, F. Heimann and S. Marnach,
Generic implementation of finite element methods in the Distributed and Unified Numerics Environment (DUNE).
Kybernetika, 46(2):294–315, 2010

Thank you for your attention!