

High Order Vectorial Finite Elements and their Implementation in C++



Joachim Schöberl

Computational Mathematics in Engineering
Institute for Analysis and Scientific Computing
Vienna University of Technology



contributions by

Center for Computational Engineering Science
RWTH Aachen University



FWF - Start project “hp-FEM”



Der Wissenschaftsfonds.

FEMWorks Finite Element Software and Consulting

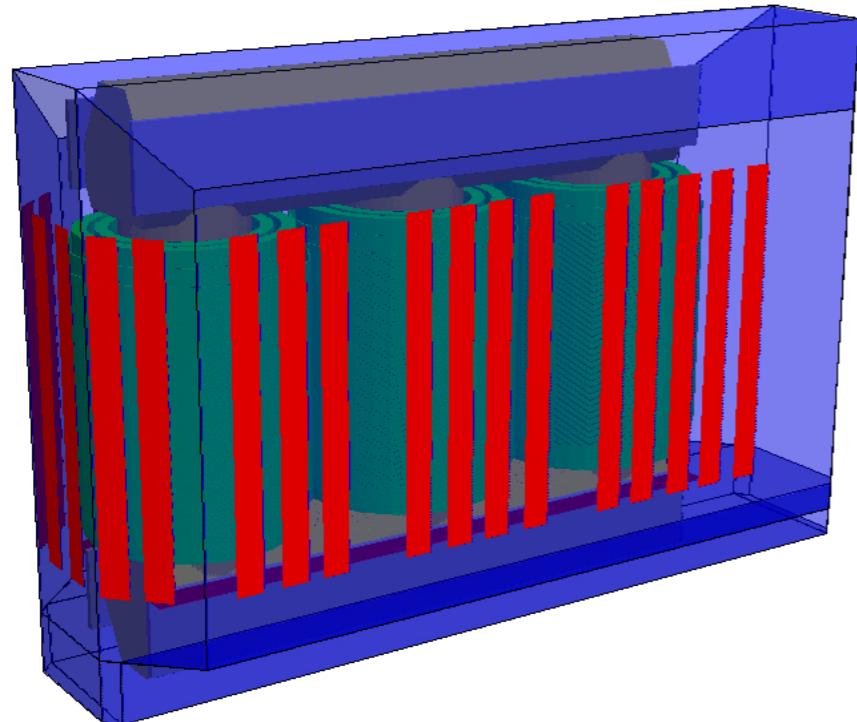


CEA-EDF-Inria School : New Trends in Compatible Discretization, July 2015, Paris

Magnetic field simulation in a transformer

Project with Siemens Austria Transformers, Linz:

Three phase transformer



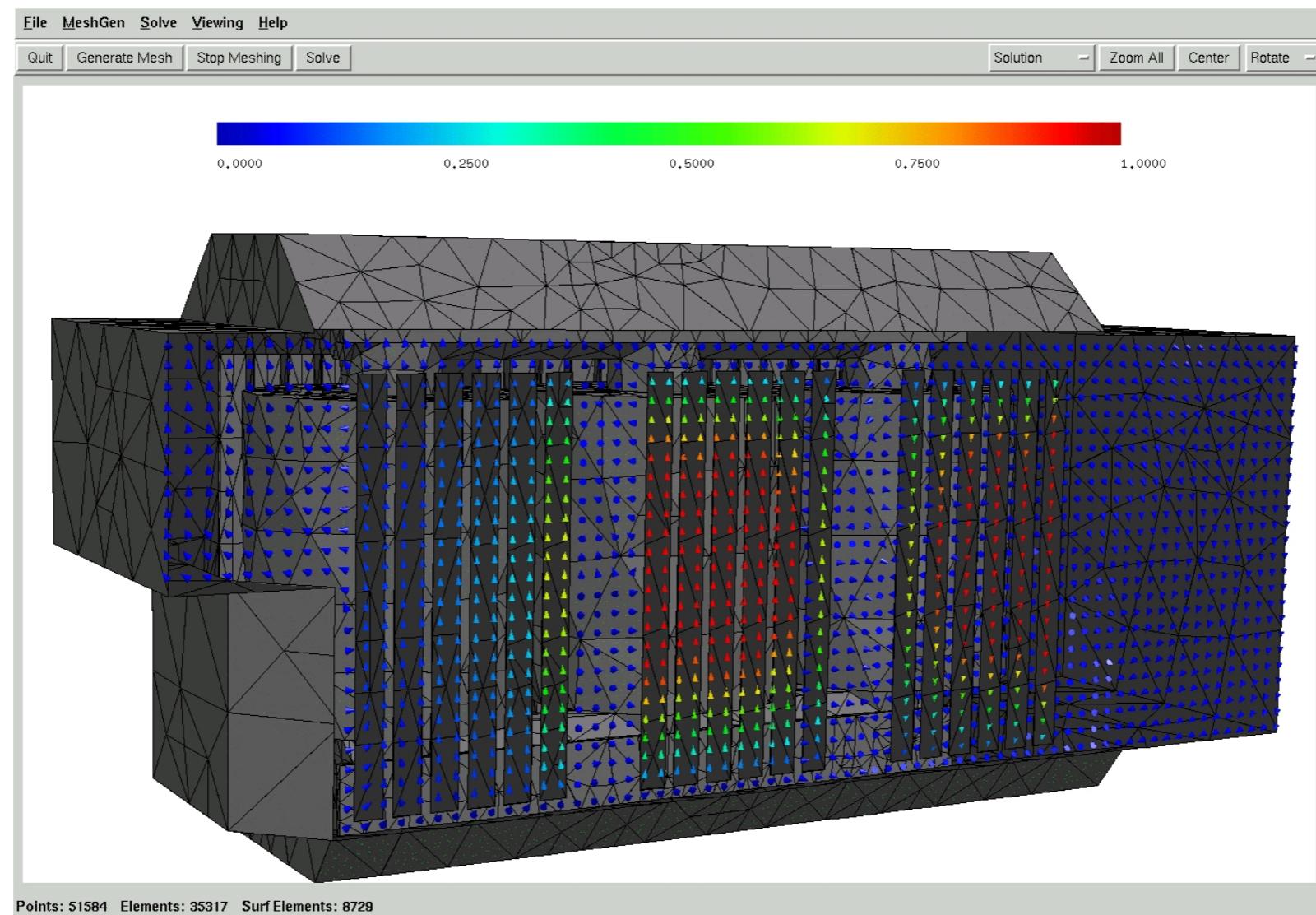
- prescribed current sources in coils
- main flux through layered core (homogenization)

- flux penetrating casing causes eddy currents
- thin shields collecting stray fluxes

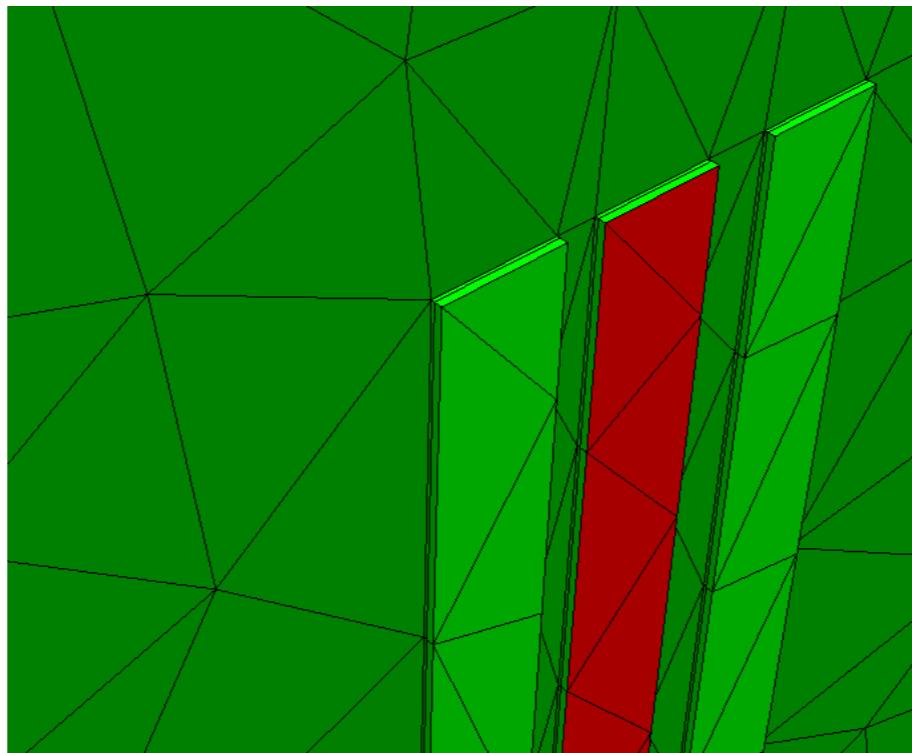
Model

- Time harmonic, low frequency
- Nonlinear terms due to saturation in casing

Magnetic flux density

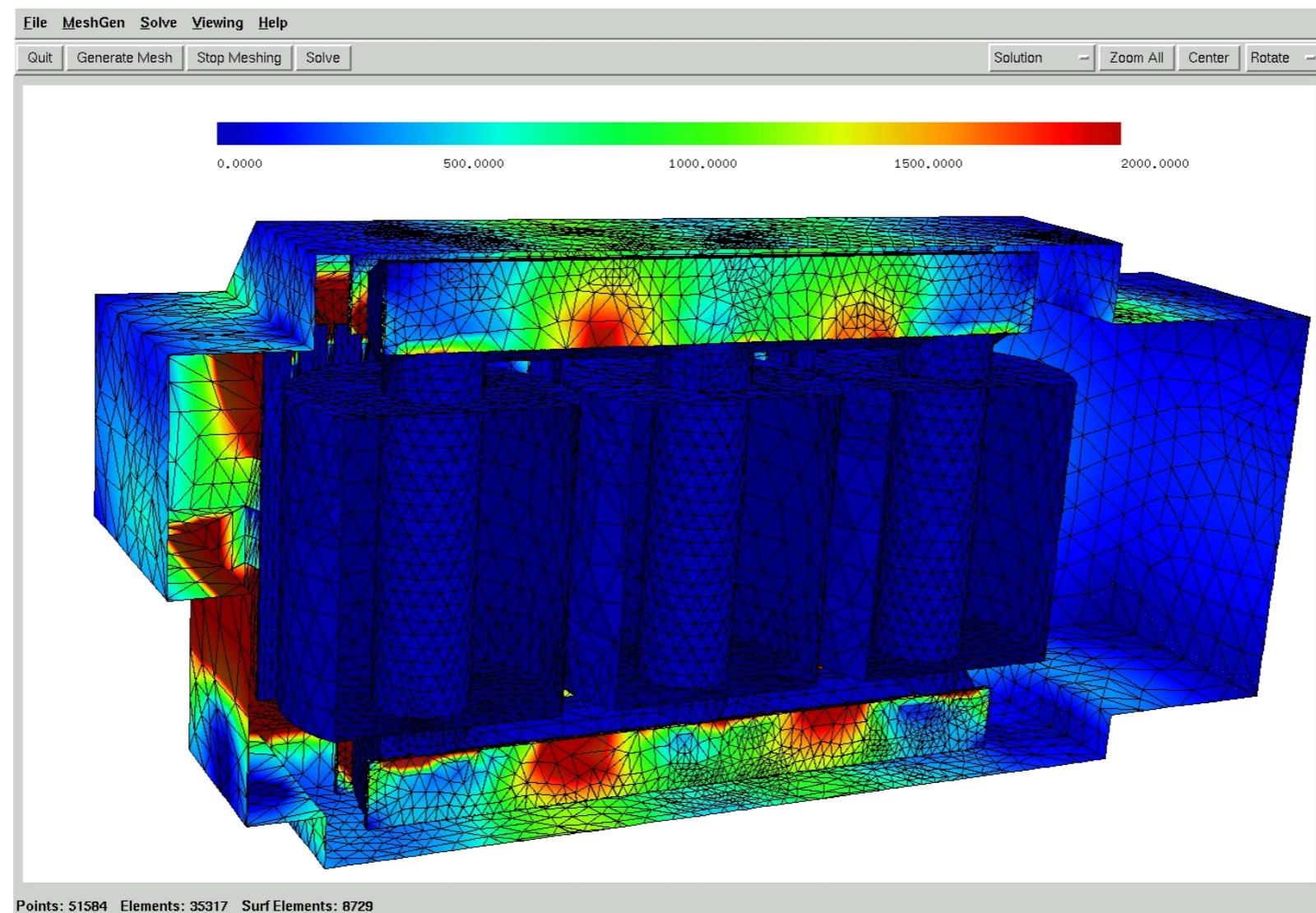


Anisotropic elements for thin shields

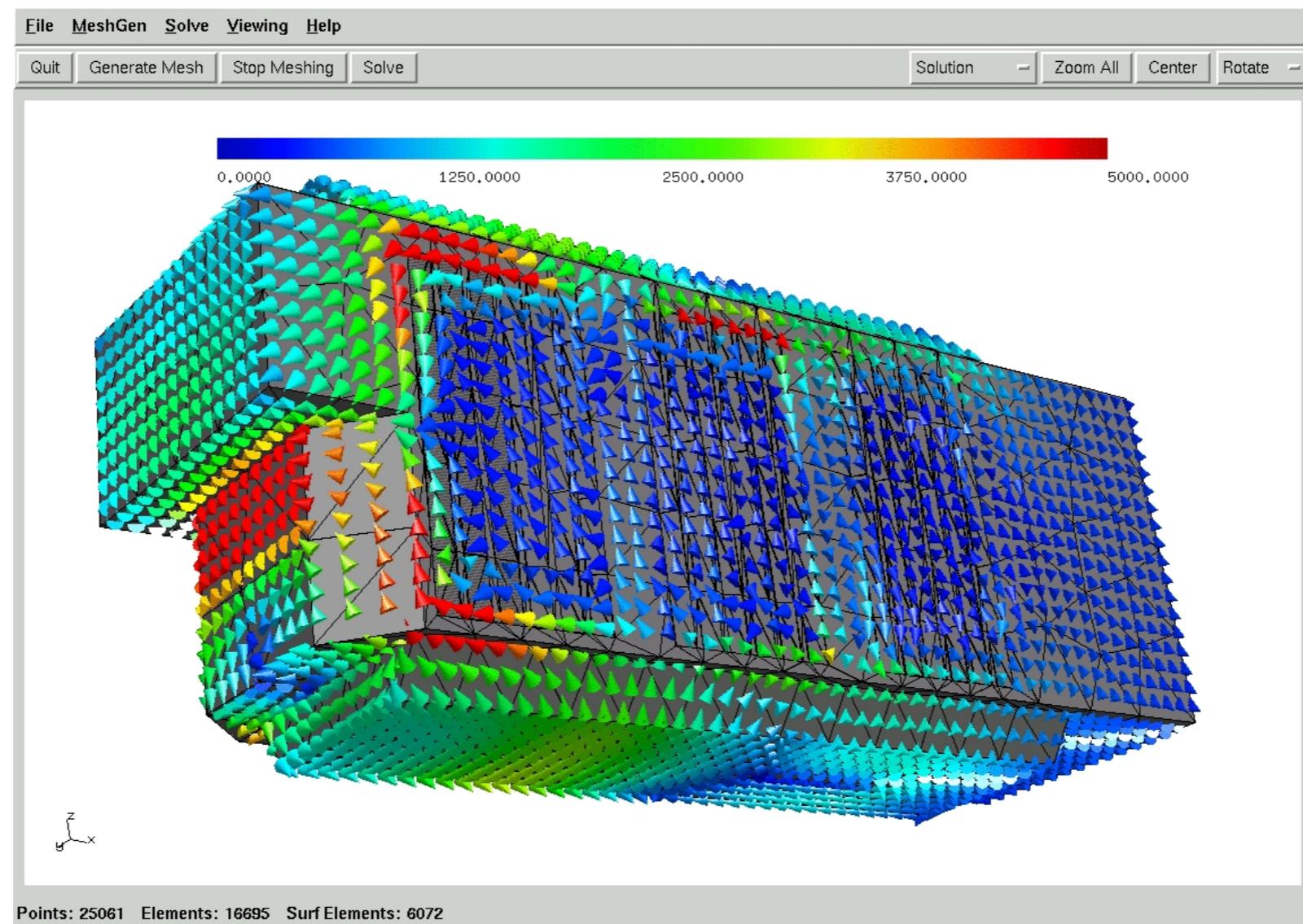


- Total object size: 6m, Thickness of shields: 2cm
- Prism elements in and behind shields, pyramid transition elements

Loss density in pressing plates



Eddy current density



Simulation data

- Simulation with our software Netgen / NGsolve and transformer-plugin
- 2^{nd} or 3^{rd} order type 2 - Nédélec elements
- 1-4 Levels of adaptive refinement, 1-4 M complex unknowns
- 2 Newton iterations per level
- 20-40 cg iterations with Schwarz preconditioner per Newton iteration
- Resources: 16 GB RAM, PC with 16 cores, 20-60 min

Equations of Magnetostatics

Given:

j .. current density s.t. $\operatorname{div} j = 0$

Compute:

B .. magnetic flux density

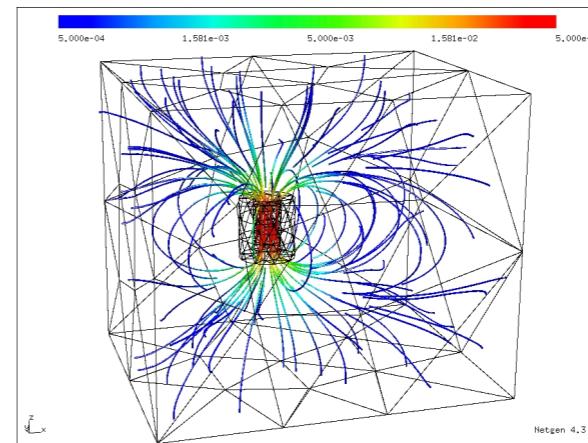
H .. magnetic field intensity

such that

$$B = \mu H \quad \operatorname{div} B = 0 \quad \operatorname{curl} H = j$$

with the boundary conditions

$$\text{either } B \cdot n = 0 \quad \text{or} \quad H \times n = 0$$



Vector potential formulation

Since $\operatorname{div} B = 0$ (plus compatibility conditions), there exists a vector potential A such that

$$B = \operatorname{curl} A$$

Combining the equations above gives us

$$\operatorname{curl} \mu^{-1} \operatorname{curl} A = j$$

with boundary conditions

$$\text{either } A \times n = 0 \quad \text{or} \quad (\mu^{-1} \operatorname{curl} A) \times n = 0$$

The weak formulation is to find $A \in V := H_D(\operatorname{curl})$ such that

$$\int \mu^{-1} \operatorname{curl} A \operatorname{curl} v \, dx = \int jv \, dx \quad \forall v \in V.$$

Problem: A is defined up to a $\nabla\varphi$!

Gauging possibilities

1. Do not gauge, work on factor space $H(\text{curl})/\nabla H^1$.

Fine, if error estimates etc. only depend on $\text{curl } A$

2. Gauging by explicit constraints, i.e., solve the mixed problem:

$$\begin{aligned} \int \mu^{-1} \text{curl } A \text{curl } v \, dx + \int v \nabla \varphi \, dx &= \int jv \, dx \quad \forall v \in H(\text{curl}) \\ \int A \nabla \psi \, dx &= 0 \quad \forall \psi \in H^1 \end{aligned}$$

Space for Lagrange parameter is H^1 . Fine, if one likes to solve saddle-point problems

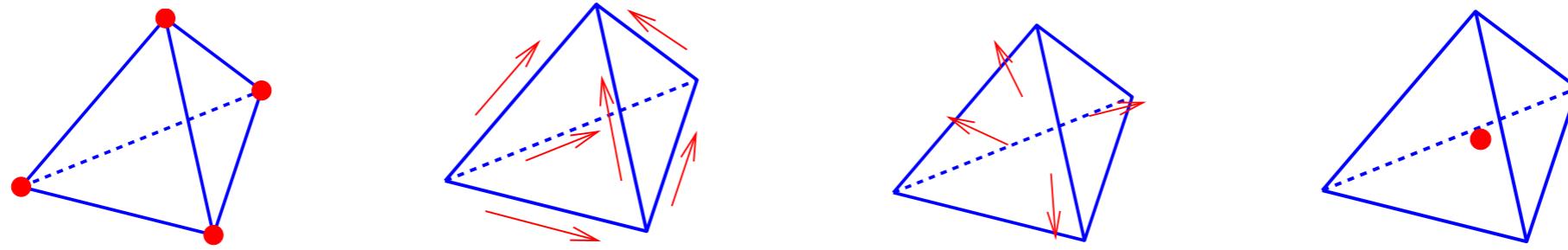
3. Gauging by regularization. Add small L_2 -term:

$$\int \mu^{-1} \text{curl } A \text{curl } v \, dx + \varepsilon \int Av \, dx = \int jv \, dx \quad \forall v \in V.$$

Fine, if error estimates etc. do not depend on ε . Exercise: $\|A^{\text{mixed}} - A^\varepsilon\|_{H(\text{curl})} = O(\varepsilon)$

The de Rham Complex

$$\begin{array}{ccccccc}
 H^1 & \xrightarrow{\nabla} & H(\text{curl}) & \xrightarrow{\text{curl}} & H(\text{div}) & \xrightarrow{\text{div}} & L^2 \\
 \cup & & \cup & & \cup & & \cup \\
 W_h & \xrightarrow{\nabla} & V_h & \xrightarrow{\text{curl}} & Q_h & \xrightarrow{\text{div}} & S_h
 \end{array}$$



satisfies the **exact sequence property**

$$\text{range}(\nabla) = \ker(\text{curl})$$

$$\text{range}(\text{curl}) = \ker(\text{div})$$

on the continuous and the discrete level.

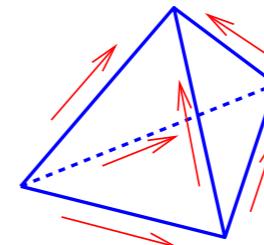
Important for stability, a priori and a posteriori error estimates, preconditioning, ...

Low-order $H(\text{curl})$ finite elements

First order Nédélec I elements:

$$V_h = \{v \in H(\text{curl}) : v|_T = a_T + b_T \times x\}$$

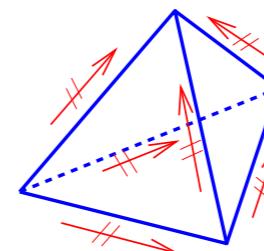
first order approximation for A -field and B -field



First order Nédélec II elements:

$$V_h = \{v \in H(\text{curl}) : v|_T \in [P^1]^3\}$$

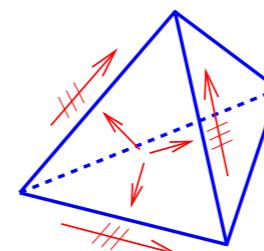
second order for A -field, first order for B -field



Second order Nédélec II elements:

$$V_h = \{v \in H(\text{curl}) : v|_T \in [P^2]^3\}$$

third order for A -field, second order for B -field



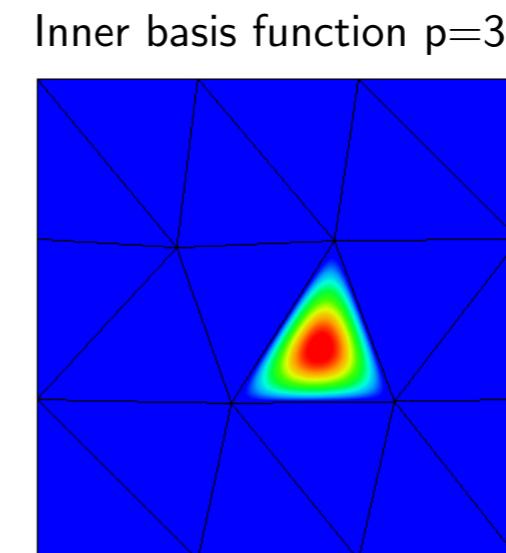
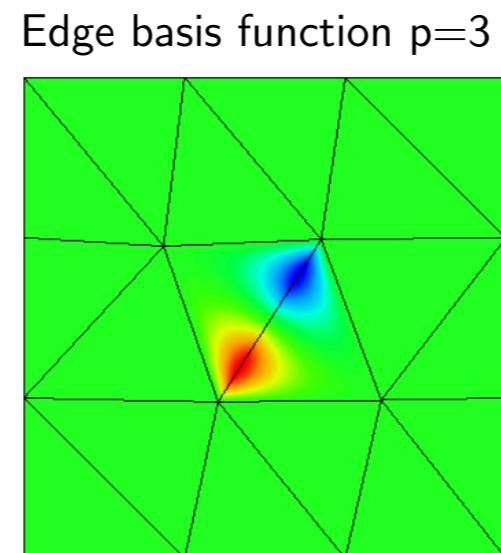
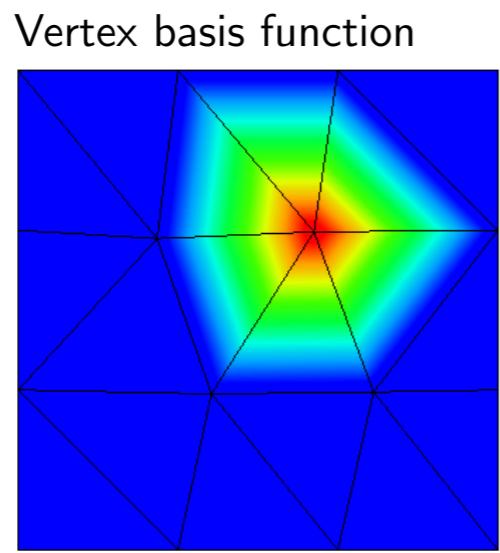
If comparing resources vs. accuracy, second order elements are more efficient.

On the construction of high order $H(\text{curl})$ finite elements

- [Dubiner, Karniadakis+Sherwin] H^1 -conforming shape functions in tensor product structure
→ allows fast summation techniques
- [Webb] $H(\text{curl})$ hierarchical shape functions with local exact sequence property
convenient to implement up to order 4
- [Demkowicz+Monk] Based on global exact sequence property construction of Nédélec elements of variable order (with constraints on order distribution) for hexahedra
- [Ainsworth+Coyle] Systematic construction of $H(\text{curl})$ -conforming elements of arbitrarily high order for tetrahedra
- [Schöberl+Zaglmayr] Based on **local exact sequence property** and by using **tensor-product structure** we achieve a **systematic strategy** for the construction of $H(\text{curl})$ -conforming hierarchical shape functions of **arbitrary** and **variable order for common element geometries** (segments, quadrilaterals, triangles, hexahedra, tetrahedra, prisms, pyramids).
[COMPEL, 2005], PhD-Thesis Zaglmayr 2006

Hierarchical $V - E - F - C$ basis for H^1 -conforming Finite Elements

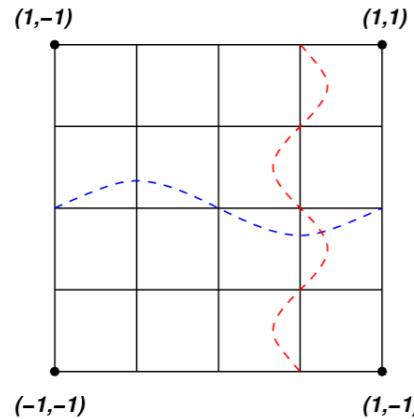
The high order elements have basis functions associated with the vertices, edges, (faces,) and cells of the mesh:



This allows an individual polynomial order for each edge, face, and cell.

High-order H^1 -conforming shape functions in tensor product structure

Exploit the tensor product structure of quadrilateral elements to build edge and face shapes



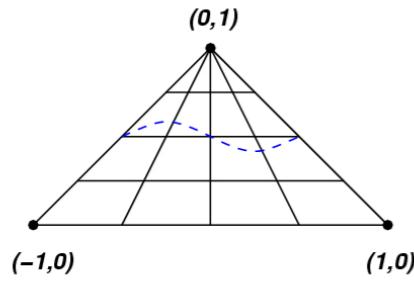
Family of orthogonal polynomials $(P_k^0[-1, 1])_{2 \leq k \leq p}$ vanishing in ± 1 .

$$\varphi_{i,j}^F(x, y) = P_i^0(x) P_j^0(y),$$

$$\varphi_i^{E_1}(x, y) = P_i^0(x) \frac{1-y}{2}.$$

Tensor-product structure for triangle [Dubiner, Karniadakis+Sherwin]:

Collapse the quadrilateral to the triangle by $x \rightarrow (1 - y)x$



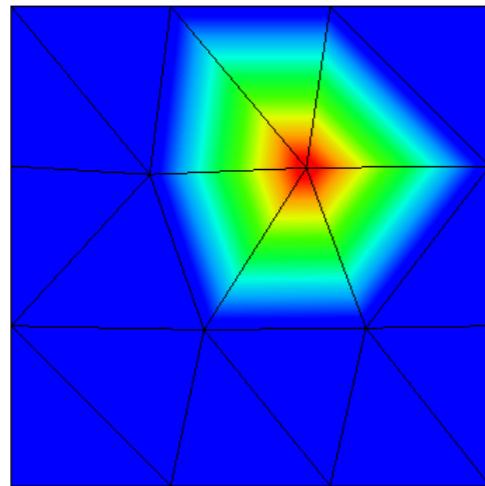
$$\varphi_i^{E_1}(x, y) = P_i^0\left(\frac{x}{1-y}\right) (1-y)^i$$

$$\varphi_{i,j}^F(x, y) = \underbrace{P_i^0\left(\frac{x}{1-y}\right)}_{u_i(x,y)} (1-y)^i \underbrace{P_j(2y-1)y}_{v_j(y)}$$

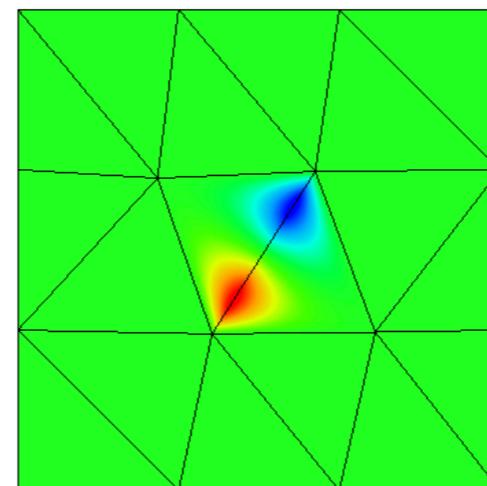
Remark: Implementation is free of divisions

The deRham Complex tells us that $\nabla H^1 \subset H(\text{curl})$, as well for discrete spaces $\nabla W^{p+1} \subset V^p$.

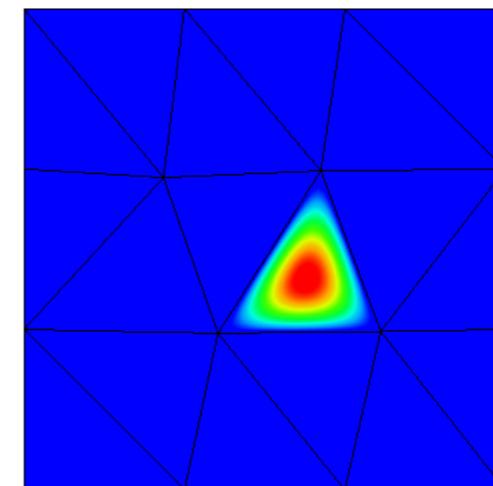
Vertex basis function



Edge basis function $p=3$

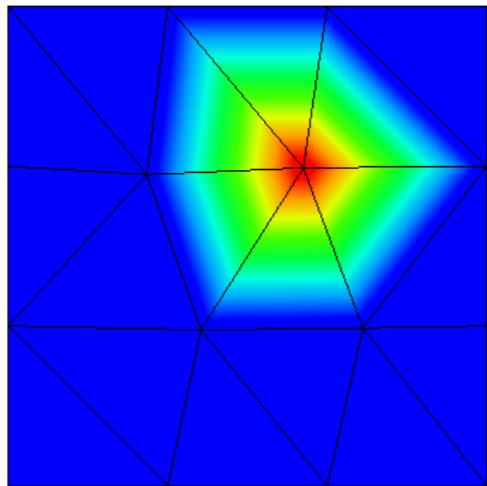


Inner basis function $p=3$

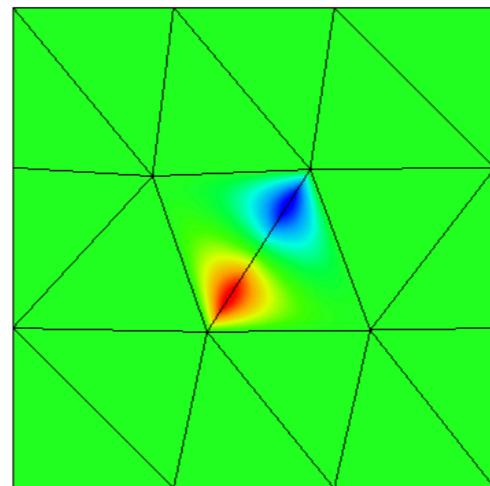


The deRham Complex tells us that $\nabla H^1 \subset H(\text{curl})$, as well for discrete spaces $\nabla W^{p+1} \subset V^p$.

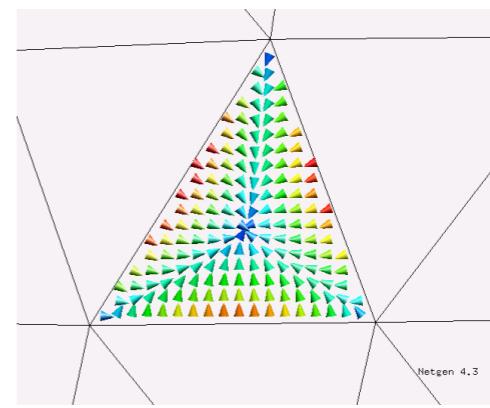
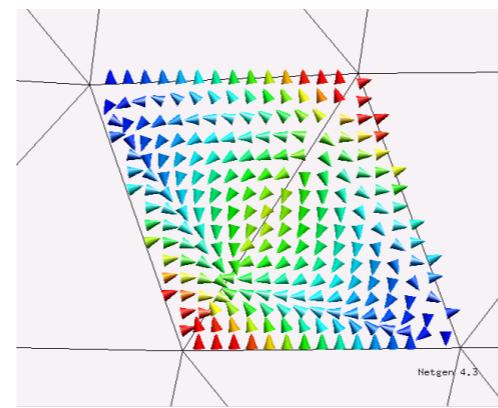
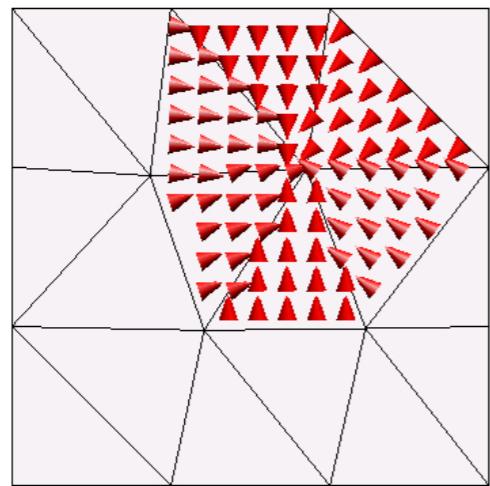
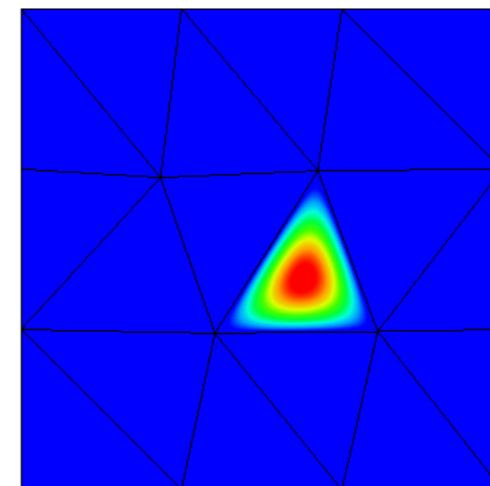
Vertex basis function



Edge basis function $p=3$



Inner basis function $p=3$



$$\nabla W_{V_i} \subset V_{\mathcal{N}_0}$$

$$\nabla W_{E_k}^{p+1} = V_{E_k}^p$$

$$\nabla W_{F_k}^{p+1} \subset V_{F_k}^p$$

$H(\text{curl})$ -conforming face shape functions with $\nabla W_F^{p+1} \subset V_F^p$

We use inner H^1 -shape functions spanning $W_F^{p+1} \subset H^1$ of the structure

$$\varphi_{i,j}^{F,\nabla} = u_i(x, y) v_j(y).$$

We suggest the following $H(\text{curl})$ face shape functions consisting of 3 types:

- **Type 1: Gradient-fields**

$$\varphi_{1,i,j}^{F,\text{curl}} = \nabla \varphi_{i,j}^{F,\nabla} = \nabla(u_i v_j) = u_i \nabla v_j + v_j \nabla u_i$$

- **Type 2: other combination**

$$\varphi_{2,i,j}^{F,\text{curl}} = u_i \nabla v_j - v_j \nabla u_i$$

- **Type 3:** to achieve a basis for V_F , $(p-1)$ lin. independent functions are missing:

$$\varphi_{3,j}^{F,\text{curl}} = N_0(x, y) v_j(y).$$

Localized exact sequence property

We have constructed **V**ertex-**E**dge-**F**ace-**C**ell shape functions satisfying

$$\begin{aligned} W_{h, p+1=1}^V &\xrightarrow{\nabla} V_h^{\mathcal{N}_0} \xrightarrow{\text{curl}} Q_h^{\mathcal{RT}_0} \xrightarrow{\text{div}} S_{h, 0} \\ W_{p_E+1}^E &\xrightarrow{\nabla} V_{p_E}^E \\ W_{p_F+1}^F &\xrightarrow{\nabla} V_{p_F}^F \xrightarrow{\text{curl}} Q_{p_F-1}^F \\ W_{p_C+1}^C &\xrightarrow{\nabla} V_{p_C}^C \xrightarrow{\text{curl}} Q_{p_C-1}^C \xrightarrow{\text{div}} S_{p_C-2}^C. \end{aligned}$$

Advantages are

- allows arbitrary and variable polynomial order on each edge, face and cell
- Additive Schwarz Preconditioning with cheap $\mathcal{N}_0 - E - F - C$ blocks gets efficient
- Reduced-basis gauging by skipping higher-order gradient bases functions
- discrete differential operators $B_\nabla, B_{\text{curl}}, B_{\text{div}}$ are trivial

Schwarz-Preconditioning for High order $\mathbf{H}(\text{curl})$ elements

The global stiffness matrix is split into the according unknowns:

$$A = \begin{pmatrix} A_{\mathcal{N}_0 \mathcal{N}_0} & A_{\mathcal{N}_0 E} & A_{\mathcal{N}_0 F} & A_{\mathcal{N}_0 C} \\ A_{E \mathcal{N}_0} & A_{EE} & A_{EF} & A_{EC} \\ A_{F \mathcal{N}_0} & A_{FE} & A_{FF} & A_{FC} \\ A_{C \mathcal{N}_0} & A_{CE} & A_{CF} & A_{CC} \end{pmatrix}.$$

A cheap preconditioner is the \mathcal{N}_0 -E-F-I block Jacobi-preconditioner

$$C = \begin{pmatrix} A_{\mathcal{N}_0 \mathcal{N}_0} & 0 & 0 & 0 \\ 0 & \tilde{A}_{EE} & 0 & 0 \\ 0 & 0 & \tilde{A}_{FF} & 0 \\ 0 & 0 & 0 & A_{CC} \end{pmatrix}.$$

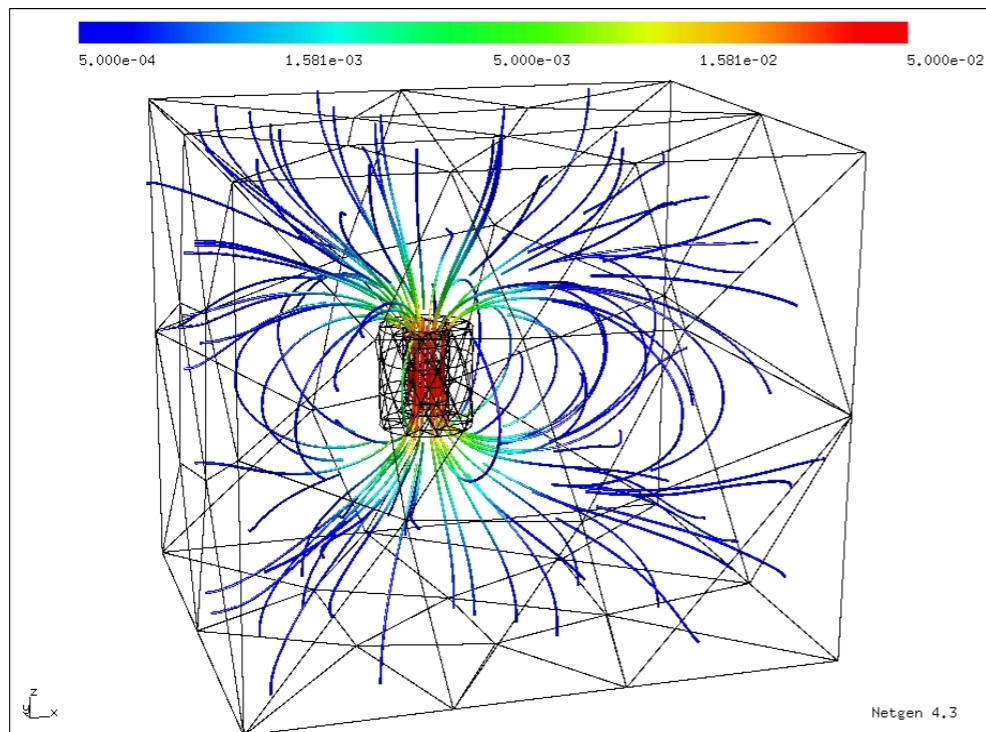
The Nedelec-0 block plays a special role: It is solved exactly, or, an h -version preconditioner is applied.

No cell-blocks after static condensation.

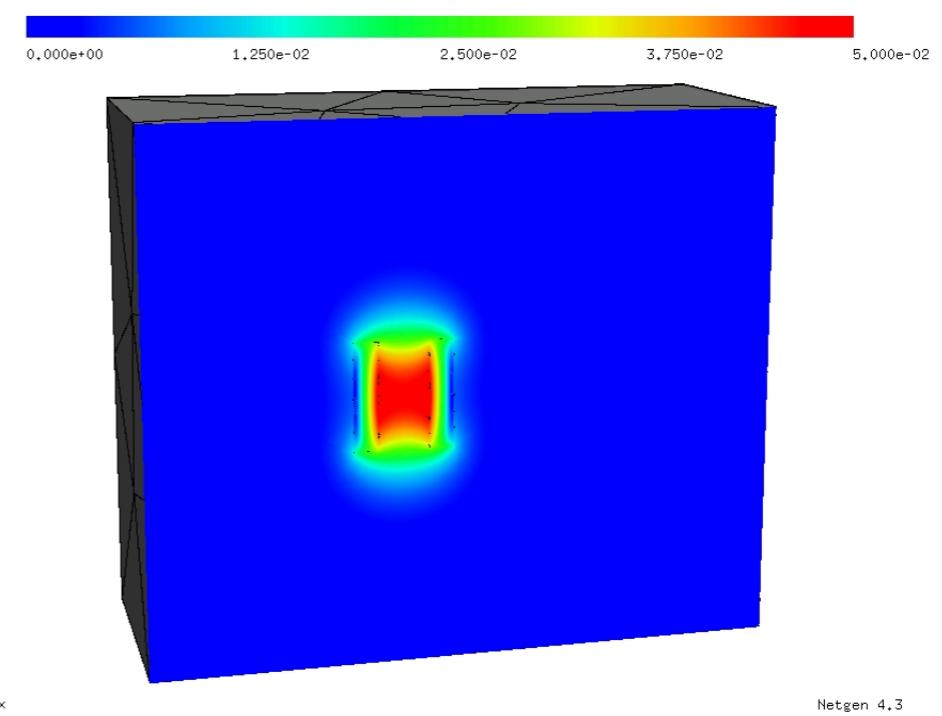
Magnetostatic boundary value problem - Numerical Results

Simulation of the magnetic field induced by a coil with prescribed currents (regularized formulation)

$$\text{Find } A \in H(\text{curl}) : \quad (\mu^{-1} \operatorname{curl} A, \operatorname{curl} v) + \epsilon(u, v) = (j, v) \quad \forall v \in H(\text{curl})$$



Field-lines induced by a coil, $p=6$.



Absolute value $|B| = |\operatorname{curl} A|$.

Tetrahedral mesh with 2035 curved elements.

Reduced Basis Gauging

- regularization term for lowest-order subspace
- skipping higher-order gradients basis functions

Reduced-base vs. full-space regularization in simulation of coil-problem:

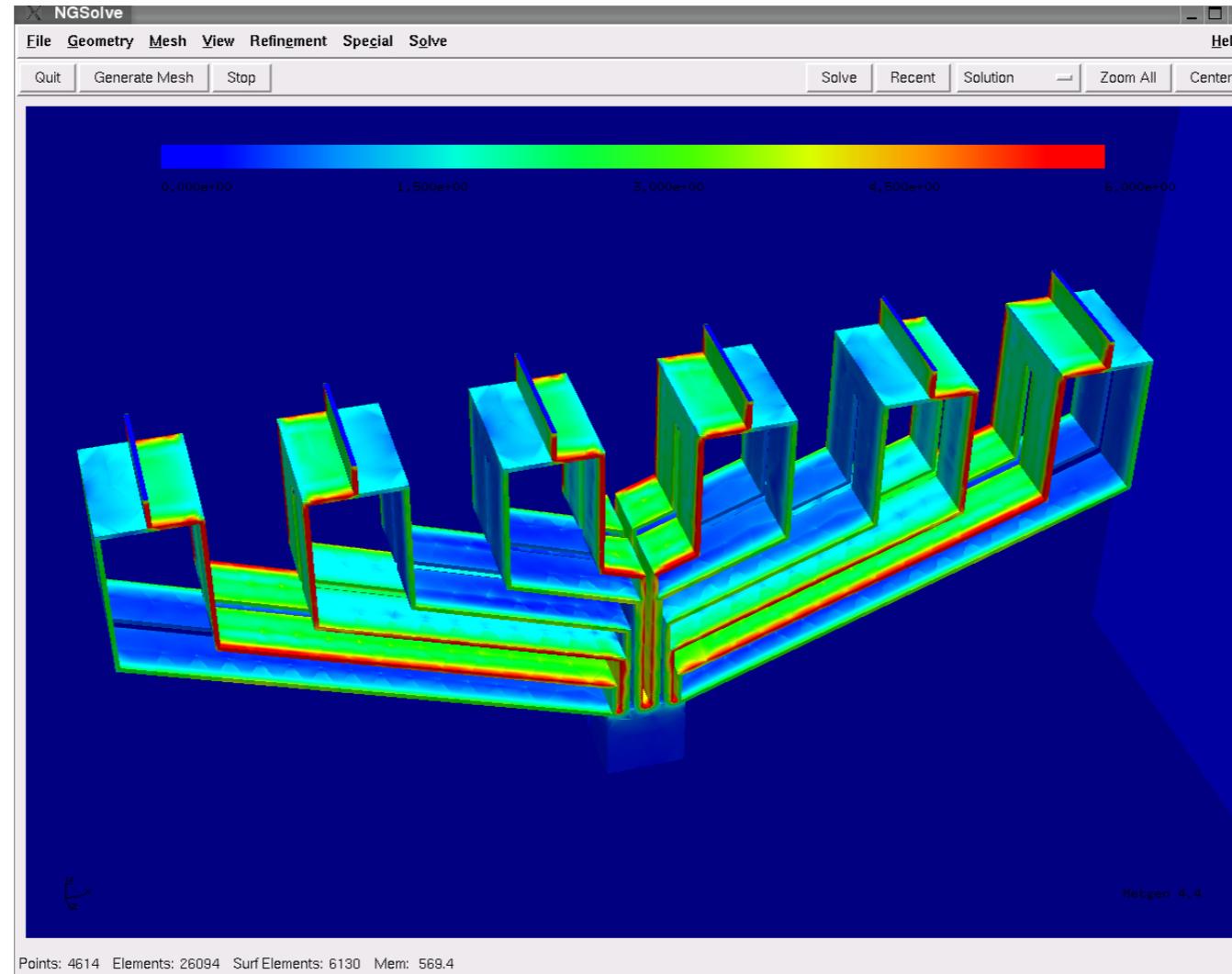
In reduced system about a third less shape functions $\rightarrow \approx 55\%$ faster integration

p	dofs	reduced/full	$\kappa(C^{-1}A)$	iterations	solver time
2	19719	full	7.9	20	1.9 s
	10686	reduced	7.9	21	0.7 s
3	50884	full	24.2	32	9.8 s
	29130	reduced	18.2	31	2.9 s
4	104520	full	71.4	48	40.5 s
	61862	reduced	32.3	40	10.7 s
5	186731	full	179.9	69	137.9 s
	112952	reduced	55.5	49	31.9 s
6	303625	full	421.0	97	427.8 s
	186470	reduced	84.0	59	87.4 s
7	286486	reduced	120.0	68	209.6 s

Note: timings from 2006, sequential code

Connection of a transformer switch (bus-bar)

Gradients can be skipped in non-conducting domains in Eddy-current problems.



Full base for $p = 3$ in conductor, reduced base for $p = 3$ in air
 $n \approx 450k$ complex unknowns, 5 sec (in 2015, 12 thds), 2 GB RAM

Fast Integration utilizing tensor-product structures

Idea for scalar quadrilateral element:

$(p + 1)^2$ basis function in tensor product structure

$$\varphi_{i,j}(x, y) = \varphi_i(x)\varphi_j(y)$$

Sum-factorization technique for numerical integration of the source vector:

$$\begin{aligned} f_{i,j} &= \int_Q f(x, y)\varphi_{i,j}(x, y)d(x, y) \\ &= \int_{-1}^1 \int_{-1}^1 J(x, y)f(x, y)\varphi_i(x)\varphi_j(y) dy dx \\ &\approx \sum_{k=1}^p \sum_{l=1}^p w_k w_l J(x_k, y_l) f(x_k, y_l) \varphi_i(x_k) \varphi_j(y_l) \\ &= \sum_{k=1}^p w_k \varphi_i(x_k) \underbrace{\sum_{l=1}^p w_l J(x_k, y_l) f(x_k, y_l) \varphi_j(y_l)}_{g_{jk}} \end{aligned}$$

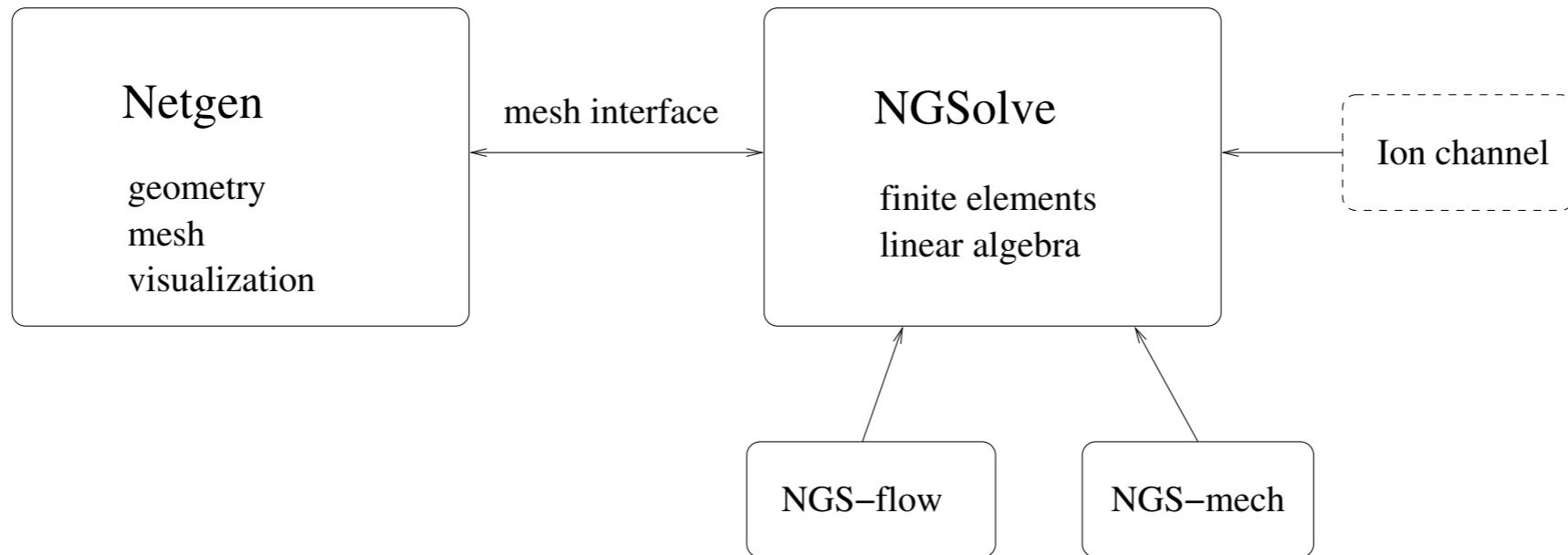
Reduces complexity from $O(p^{2d})$ to $O(p^{d+1})$ [Karniadakis-Sherwin, Melenk-Gerdes-Schwab].

Fast Integration for $H(\text{curl})$ elements

- Similar technique for element-matrices: Reduction from $O(p^9)$ to $O(p^7)$
- Possible for basis in tensor product structure.
- Timing (in [ms]) for integration of one curved Nedelec tetrahedral element
(1700 MHz Centrino, 2007)

p	N	$\int \text{curl } u \text{ curl } v$		$\int u \cdot v$	
		std	fast	std	fast
2	30	0.06	0.07	0.11	0.15
3	60	0.30	0.21	1.07	0.41
4	105	2.40	0.54	4.60	1.05
5	168	10.51	1.79	17.67	2.85
6	252	38.16	4.27	58.90	5.65
...					
10	858	1640.00	55.00	2180.00	98.00

Interplay of Netgen and NGSolve



- Netgen knows the geometry, maintains the mesh, visualization
- NGSolve contains the finite elements and linear algebra

Application classes can be loaded as shared libraries into NGSolve

Open source (LGPL license), available from sourceforge, runs on Linux, Windows, MacOS

Python - interface to NGSolve (NGSolve 6.0)

```
mesh = Mesh("square.vol")

v = FESpace ("h1ho", mesh, order=4, dirichlet=[1])
u = GridFunction (v)

f = LinearForm (v)
f += LFI ("source", coef=sin(x)*y)

a = BilinearForm (v, symmetric=True)
a += BFI ("laplace", coef = 1)

c = Preconditioner (a, "bddc")

f.Assemble()
a.Assemble()

solver = CGSolver (a.mat, c.mat, printrates=True, precision=1e-10)
u.vec.data = solver * f.vec
```

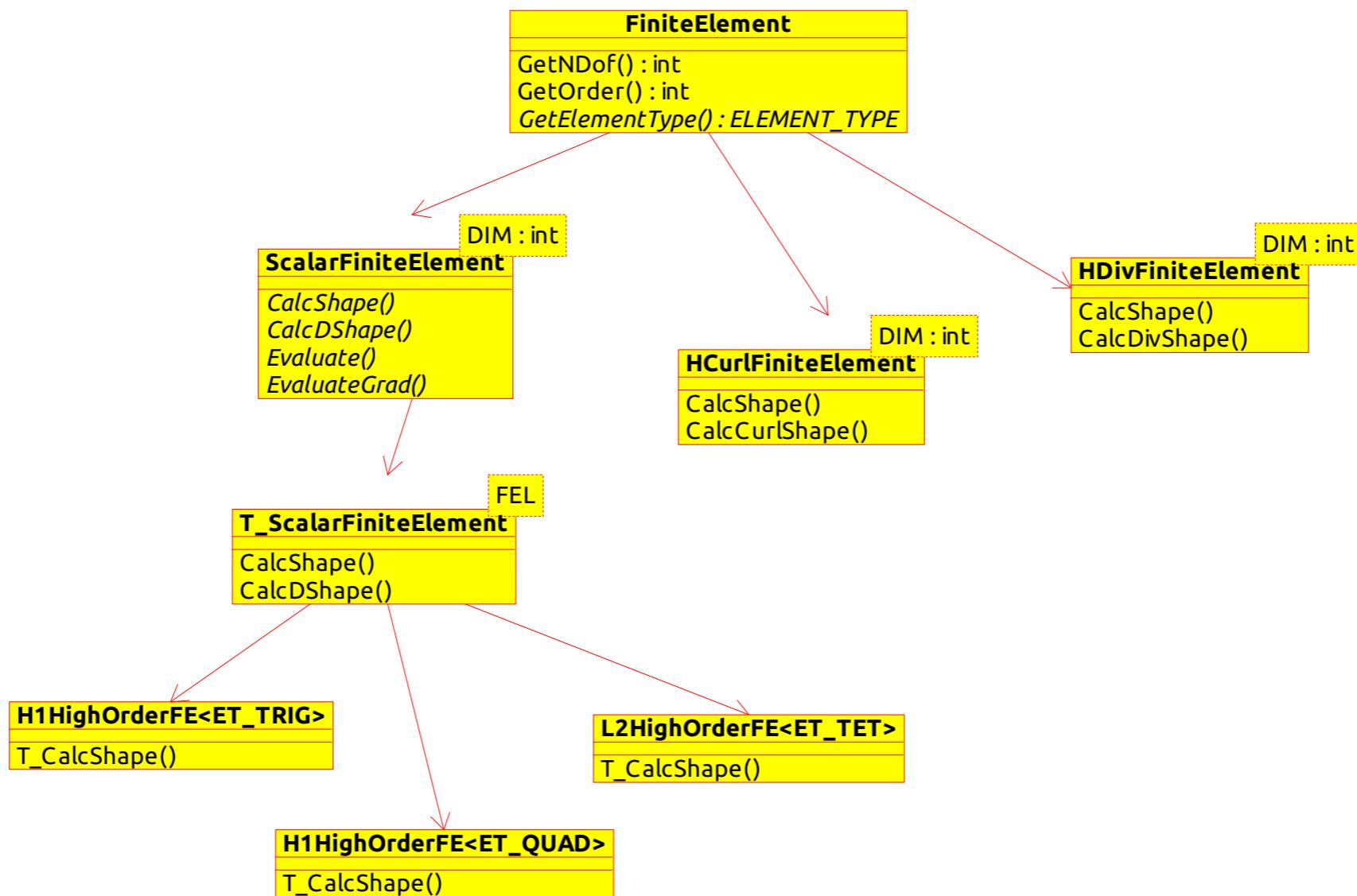
Tutorial code example: Element matrix calculation

```
void CurlCurlIntegrator ::  
CalcElementMatrix (FiniteElement & base_fel, ElementTransformation & eltrans,  
Matrix & elmat)  
{  
    HCurlFE<3> & el = static_cast<HCurlFE<3>&> (base_fel);  
  
    IntegrationRule ir(el.ElementType(), 2*el.Order());  
    Matrix cshape(el.NDof(),3);  
  
    elmat = 0;  
    for (int i = 0; i < ir.Size(); i++)  
    {  
        MappedIntegrationPoint<3,3> mip(ir[i], eltrans);  
        fel.CalcMappedCurlShape(mip, cshape);  
        double nu = coef -> Evaluate(mip);  
        elmat += mip.Weight() * nu * cshape * Trans(cshape);  
    }  
}
```

Function evaluation in integration points

$$u(x_k) = \sum_{i=1}^{NDOF_T} c_i \varphi_i(x_k)$$

- Precompute shape functions in integration points
 - matrix-vector products (BLAS2)
 - combine set of elements and use matrix-matrix products (vendor provided BLAS3 functions) to obtain close-to-peak floating-point performance
- Shape function evaluation on the fly:
 - reduces memory transfer and local memory requirements (L1-Cache, GPU, ...)
 - recursive evaluation requires 4 (fused mult-add) operations per shape function per point
 - efficient implementation is challenging



L1: common base, L2: user level, L3: functionality level, L4: shape-function level

Level 4: generic shape function implementation

```
template<> template<typename T, typename TSHAPE>
void H1HighOrderFE<ET_TRIG> :: T_CalcShape (T x[], TSHAPE & shape) const
{
    T lam[3] = { x[0], x[1], 1-x[0]-x[1] };
    for (int i = 0; i < N_VERTEX; i++) shape[i] = lam[i];

    int ii = N_VERTEX;
    for (int i = 0; i < N_EDGE; i++) { // edge-based basis functions
        INT<2> e = GetEdge(i);
        LegendrePolynomial::
            EvalScaledMult (order_edge[i]-2, lam[e[1]]-lam[e[0]], lam[e[0]]+lam[e[1]],
                            lam[e[0]]*lam[e[1]], shape+ii);
        ii += order_edge[i]-1;
    }
    DubinerBasis::EvalMult (order_face-3, lam[0], lam[1], lam[0]*lam[1]*lam[2], shape+ii);
}
```

Shape functions are computed, and immediately 'stored' in shape object.

No local arrays are required, all computations are done in registers.

Dubiner basis - L_2 -orthogonal on triangles

$$\varphi_{i,j}(x, y) = P_i \left(\frac{x}{1-y} \right) (1-y)^i P_j^{(2i+1,0)}(2y-1)$$

```
template <class S, class T>
void DubinerBasis::Eval (int n, S x, S y, T && values)
{
    LegendrePolynomial leg;
    int ii = 0;
    leg.EvalScaled (n, x, 1-y,
                    SBLambda ([&] (int i, S val)
                               {
                                   JacobiPolynomialAlpha jac(1+2*i);
                                   jac.EvalMult (n-i, 2*y-1, val, values+ii)
                                   ii += n+1-i;
                               }));
}
```

Level 3: Functionality level

```
void T_ScalarFE<FEL>::CalcShape (IntegrationPoint ip, Vector & shape)
{
    T_CalcShape (ip, shape);      // Up-cast via Barton-Neckman trick
}
```

Level 3: Functionality via Lambda-functions - programming in dual space

standard version using automatic differentiation:

```
void T_ScalarFE<FEL>::CalcDShape (IntegrationPoint ip, Matrix & dshapes)
{
    Vec<DIM,AutoDiff<DIM>> adip = ip;
    Array<AutoDiff<DIM>> ad_shapes(ndof);

    T_CalcShape (adip, ad_shapes);
    for (int i = 0; i < ndof; i++)
        ad_shapes[i].StoreGradient(dshapes.Row(i));
}
```

elimination of local array:

```
void T_ScalarFE<FEL>::CalcDShape (IntegrationPoint ip, Matrix & dshapes)
{
    Vec<DIM,AutoDiff<DIM>> adip = ip;
    T_CalcShape (adip, [&](int nr, AutoDiff<DIM> val)
                 { val.StoreGradient (dshapes.Row(nr)); })
}
```

Level 3: Functionality via Lambda-functions - programming in dual space

original version:

```
double T_ScalarFE<FEL>::Evaluate (IntegrationPoint ip, Vector & coefs)
{
    Array<double> shapes(ndof);
    T_CalcShape (ip, shapes);
    return InnerProduct (shapes, coefs);
}
```

elimination of local array:

```
double T_ScalarFE<FEL>::Evaluate (IntegrationPoint ip, Vector & coefs)
{
    double sum = 0;
    T_CalcShape (ip, [&](int nr, double val) { sum += val * coefs(nr); });
    return sum;
}
```

shape function gets processed immediately after being computed !

Optimization by vectorized operations

Modern CPUs can do 4 double-precision single-instruction-multiple-data (SIMD) operations simultaneously (AVX extension). 8 operations on Xeon-Phi and coming Skylake CPU

Useful for integration-rule operations

```
void T_ScalarFE<FEL>::Evaluate (IntegrationRule & ir, Vector & coefs, Vector & vals)
{
    for (int i = 0; i < ir.Size(); i += 4) {
        Vec<DIM,MD<4>> ip (ir[i], ir[i+1], ir[i+2], ir[i+3]);
        MD<4> sum = 0;
        T_CalcShape (ip, [&](int nr, MD<4> val) { sum += val * coefs(nr); });
        sum.Store(&vals(i), MD<4,int>::FirstInt() < ir.Size()-i); // conditional store
    }
}
```

motivated by Vc - library by Kretz+Lindenstrutz

similar approach on GPUs using CUDA.

[Thesis M. Hochsteger]

$H(\text{curl})$ -elements

High order $H(\text{curl})$ -basis functions are represented as gradients, or anti-symmetric forms (of scalar functions u, v):

$$\nabla u, \quad u \nabla v - v \nabla u$$

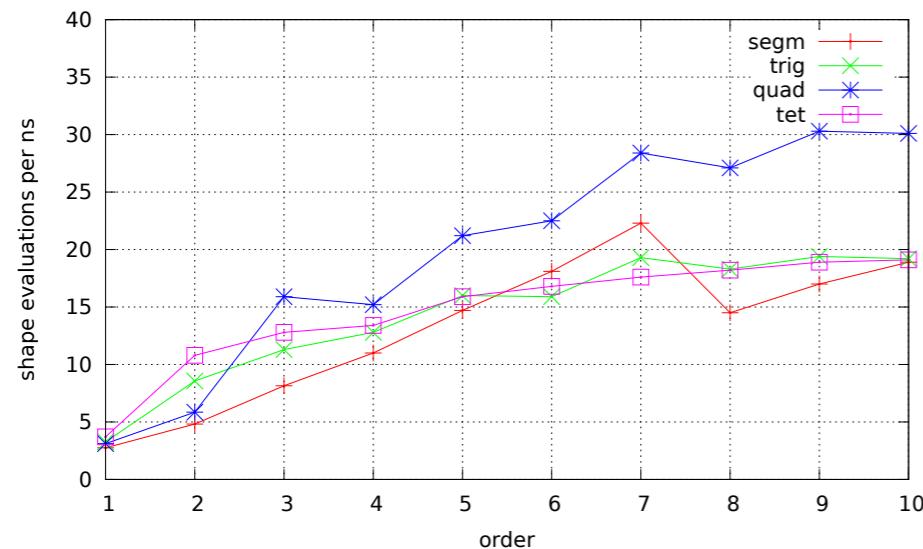
```
template<typename TSHAPE>
void HCurlHighOrderFE<ET_TRIG> :: T_CalcShape (AutoDiff<2> x[], TSHAPE & shape)
{
    AutoDiff<2> lam[3] = { x[0], x[1], 1-x[0]-x[1] };

    for (int i = 0; i < N_EDGE; i++)
        shape[i] = uDv_minus_vDu (lam[e[i][0]], lam[e[i][1]]);

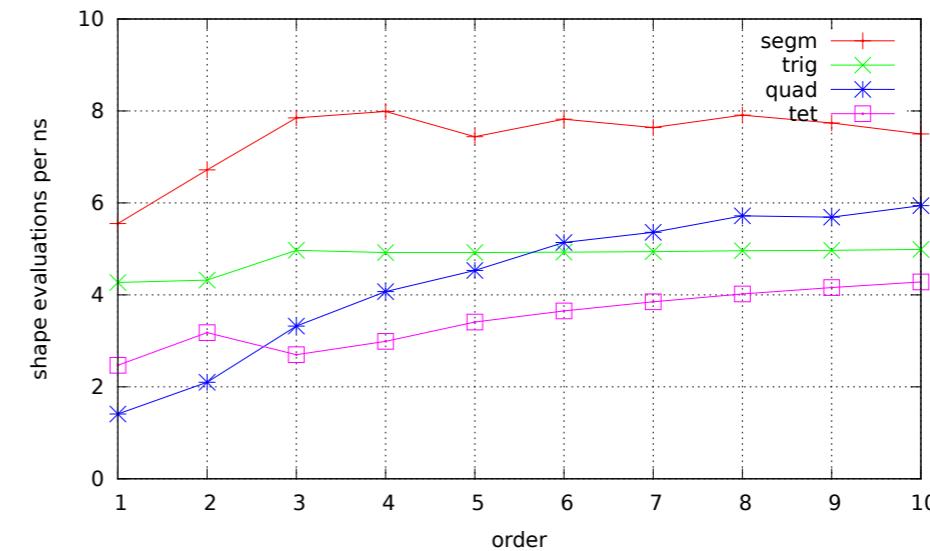
    for (int i = 0; i < N_EDGE; i++)
        LegendrePolynomial::
            EvalScaledMult (order_edge[i]-2, lam[e[1]]-lam[e[0]], ...
                            [&](int nr, T val) { shape[ii++] = Du(val); });
    ...
}
```

Caller converts basis-function objects to either values, or curls.

Finite Element function evaluation timings



Utilizing AVX



Without AVX

Evaluation takes 4 multiplications per shape function.

$$2 \text{ (CPUs)} \times 6 \text{ (cores)} \times 4 \text{ (AVX)} \times 2 \text{ GHz} = 96 \text{ GFlop/s}$$

$$\text{tetrahedral elements: } 20 \text{ G shapes/second} \times 4 = 80 \text{ GFlop/s}$$

Code utilizes 80% of theoretical peak performance !

JS 2015: C++11 Implementation of Finite Elements in NGSolve (submitted to TOMS - Transactions on Math Software)

Sum-factorization

Utilize tensor-product structure of shape functions and integration rule:

$$\varphi_i(x, y) = \varphi_i^X(x) \varphi_{i1d(i)}^Y(y)$$

Where $i1d : \{1 \dots n2d\} \rightarrow \{1 \dots n1d\}$ with $n2d = O(p^2), n1d = O(p)$.

Evaluation on tensor-product grid: $u(x_p, y_q) = \sum_{i=1}^{n2d} c_i \varphi_i^X(x_p) \varphi_{i1d(i)}^Y(y_q)$

$$h_{p,j} = \sum_{\substack{i=1 \\ i1d(i)=j}}^{n2d} c_i \varphi_i(x_p) \quad \forall p$$

$$u(x_p, y_q) = \sum_{j=1}^{n1d} h_{p,j} \varphi_j^Y(y_q) \quad \forall p, q$$

Similar 3D, trigs and tets: Duffy transform [S. A. Orszag 80, Karniadakis et al, ...]

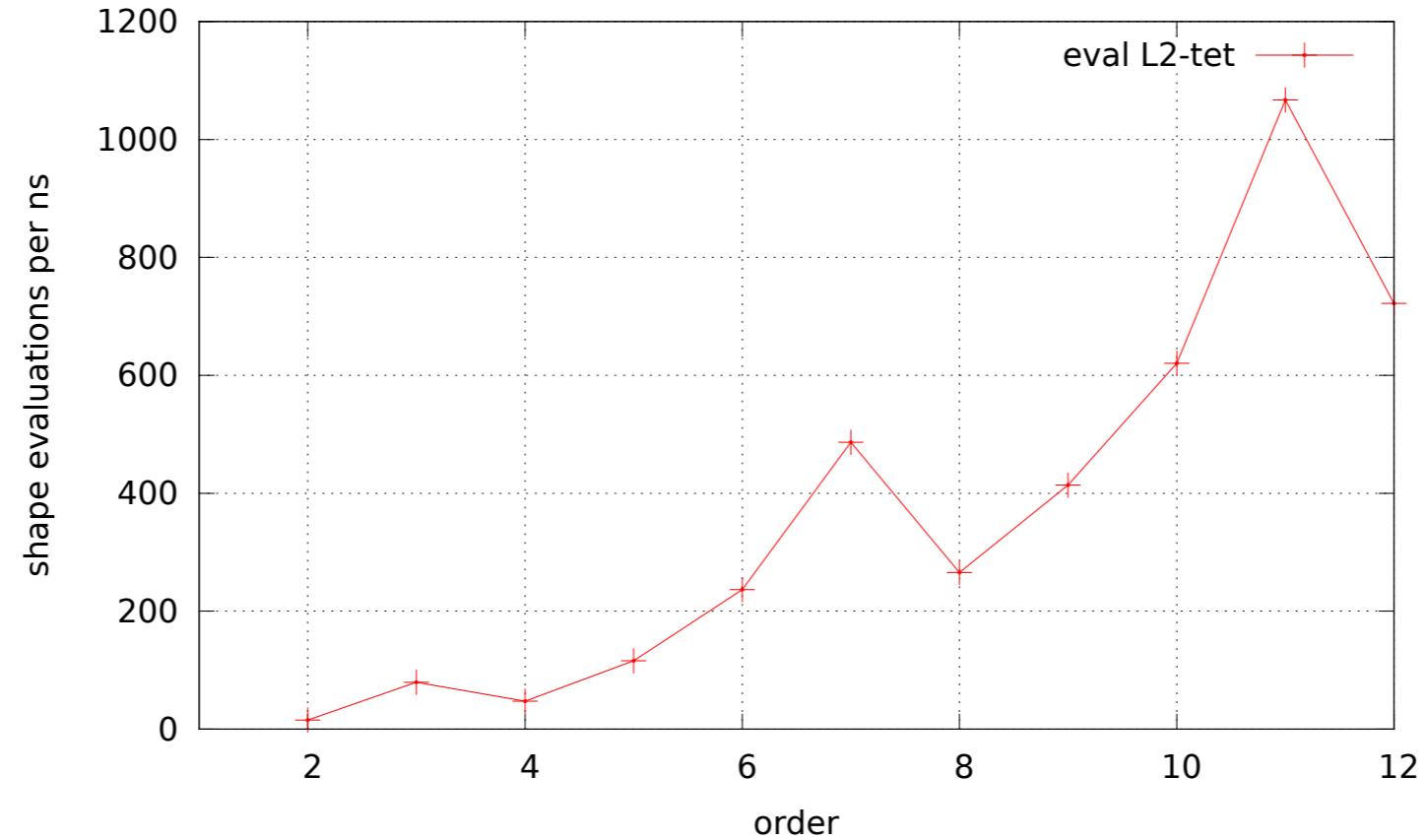
Generic implementation using sum-factorization

```
Vector<> coef1d (n1d);
for (int ix = 0, ii = 0; ix < irx.Size(); ix++)
{
    coef1d = 0; // 1D help-vector

    CalcXFactor (irx[ix](0),
                  [&](int i, int i1d, double val)
                  { coef1d(i1d) += val*coefs(i); });

    for (int iy = 0; iy < iry.Size(); iy++)
    {
        double sum = 0;
        CalcYFactor (iry[iy](0),
                      [&](int i, double val)
                      { sum += val*coef1d(i); });
        vals(ii++) = sum;
    }
}
```

Evaluation using sum-factorization



Beats best-possible matrix-matrix multiplication at order 5 !

Iterative Equation Solvers

Since the systems of equations get huge, the goal is to apply iterative methods.

The performance of Krylov space methods as CG, QMR, GMRES heavily depends on the applied *preconditioner*. Classical examples for preconditioners are the Jacobi preconditioner ($C = \text{diag } A$), Block-Jacobi, SSOR, ADI, Incomplete Cholesky, ...

Concept of Multigrid

To solve the problem on the finest mesh, take additional, coarser meshes.

In the best case, you have a hierarchy of meshes.

Next, pick a cheap iteration on each level (e.g. Jacobi, Gauss-Seidel, ...). This iteration is called smoother.

Multigrid is a strategy to combine these cheap (but inefficient) preconditioners, to one, cheap and efficient preconditioner.

Each component is responsible for certain components.

Algebraic Multigrid (AMG): Only the fine grid matrix is given. Compute an artificial hierarchy.

S. Reitzinger + J.S.: AMG for Maxwell problems (02)

Additive Schwarz preconditioning for parameter-dependent systems

Let

$$A = K + \kappa M \quad \text{with} \quad V_0 = \ker \{K\}$$

The ASM-Lemma gives an explicit representation for the preconditioner C :

$$u^T C u = \inf_{\substack{u_i \in V_i \\ u = \sum u_i}} \sum u_i^T A u_i$$

Theorem: The AS-preconditioner is robust in $\kappa \in (0, 1]$, i.e.,

$$u^T C u \approx u^T A u$$

if and only if

$$V_0 = \sum_{i=1}^m V_i \cap V_0.$$

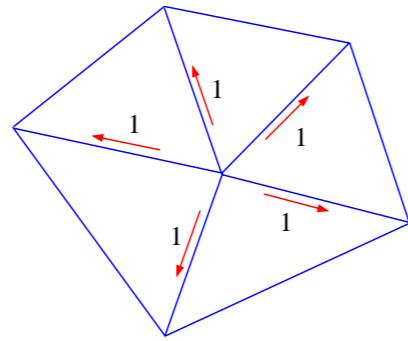
Hiptmair / Arnold-Falk-Winther smoothers for $H(\text{curl})$ problems

There is $V_0 = \nabla W_h$. Use a block smoother with blocks V_i such that

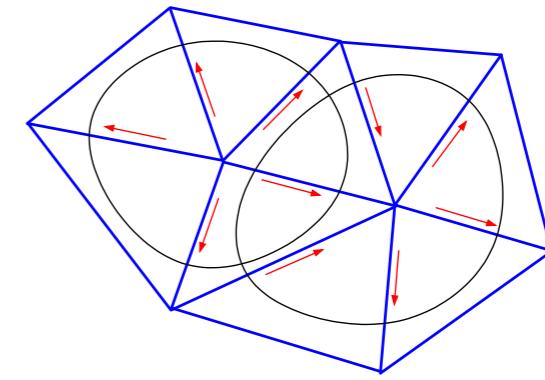
$$\forall j \exists i : \quad \nabla W_j \subset V_i,$$

where the scalar space is decomposed as $W_h = \sum W_j$.

Gradient of
vertex shape function:
Hiptmair blocks



Arnold-Falk-Winther:
Use large blocks:



Hiptmair [99]: one iteration is cheaper, less memory

Arnold-Falk-Winther [00]: less iterations, simpler implementation

J. Schöberl: A multilevel decomposition result in $H(\text{curl})$. in Multigrid, Multilevel and Multiscale Methods, EMG 2005: using commuting quasi-interpolation operators

Schwarz-Preconditioning for High order $\mathbf{H}(\text{curl})$ elements

The global stiffness matrix is split into the according unknowns:

$$A = \begin{pmatrix} A_{\mathcal{N}_0 \mathcal{N}_0} & A_{\mathcal{N}_0 E} & A_{\mathcal{N}_0 F} & A_{\mathcal{N}_0 C} \\ A_{E \mathcal{N}_0} & A_{EE} & A_{EF} & A_{EC} \\ A_{F \mathcal{N}_0} & A_{FE} & A_{FF} & A_{FC} \\ A_{C \mathcal{N}_0} & A_{CE} & A_{CF} & A_{CC} \end{pmatrix}.$$

A cheap preconditioner is the \mathcal{N}_0 -E-F-I block Jacobi-preconditioner

$$C = \begin{pmatrix} A_{\mathcal{N}_0 \mathcal{N}_0} & 0 & 0 & 0 \\ 0 & \tilde{A}_{EE} & 0 & 0 \\ 0 & 0 & \tilde{A}_{FF} & 0 \\ 0 & 0 & 0 & A_{CC} \end{pmatrix}.$$

The Nedelec-0 block plays a special role: It is solved exactly, or, an h -version preconditioner is applied.

Space splitting and local exact sequence property

The potential FE-space is split into Vertex-Edge-Face-Cell blocks

$$W_{p+1} = W_V + \sum_E W_E + \sum_F W_F + \sum_C W_C \subset H^1$$

and the $H(\text{curl})$ by lowest order Nedelec-(high order)Edge-Face-Cell based hierachic spaces

$$V_p = V_{\mathcal{N}_0} + \sum_E V_E + \sum_F V_F + \sum_C V_C \subset H(\text{curl})$$

The $H(\text{curl})$ -splitting is compatible with the kernel $V_0 = \nabla W_{p+1}$, if the sequences associated with the edge, face, and cell nodes are exact ([local exact sequence property](#)):

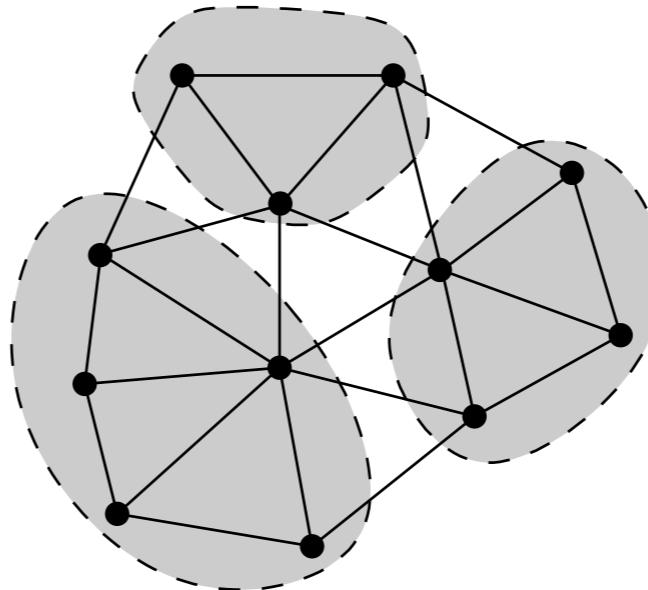
$$\begin{array}{ccccccc} W_{h,p+1=1}^V & \xrightarrow{\nabla} & V_h^{\mathcal{N}_0} & \xrightarrow{\text{curl}} & Q_h^{\mathcal{RT}_0} & \xrightarrow{\text{div}} & S_{h,0} \\ W_{p_E+1}^E & \xrightarrow{\nabla} & V_{p_E}^E & & & & \\ W_{p_F+1}^F & \xrightarrow{\nabla} & V_{p_F}^F & \xrightarrow{\text{curl}} & Q_{p_F-1}^F & & \\ W_{p_I+1}^C & \xrightarrow{\nabla} & V_{p_I}^C & \xrightarrow{\text{curl}} & Q_{p_I-1}^C & \xrightarrow{\text{div}} & S_{p_I-2}^C \end{array}$$

Algebraic coarsening based on Agglomeration

[S. Reitzinger + J. S., 2002]

Coarse grid vertices are defined by the mapping

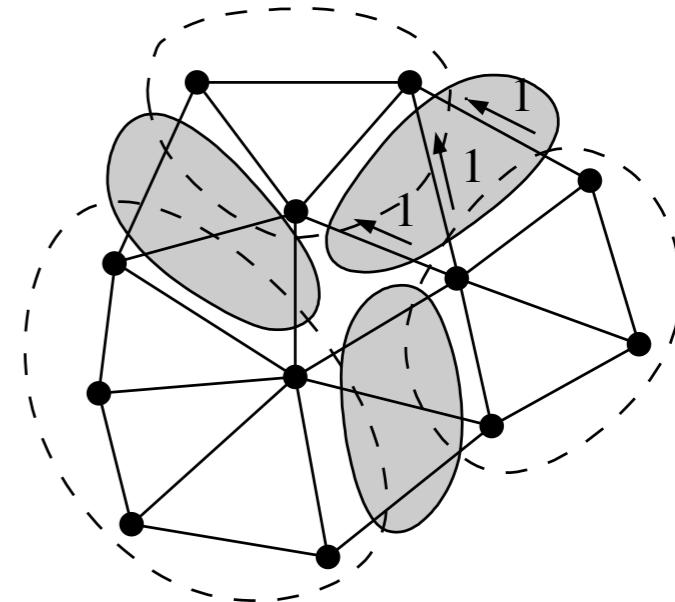
$$\text{Ind}(\cdot) : \text{Vertex} \rightarrow \text{Cluster}$$



Allows to define the full coarse grid topology:

E_{IJ} is a coarse grid edge if and only if $I \neq J$, and there are fine grid vertices i and j s.t.:

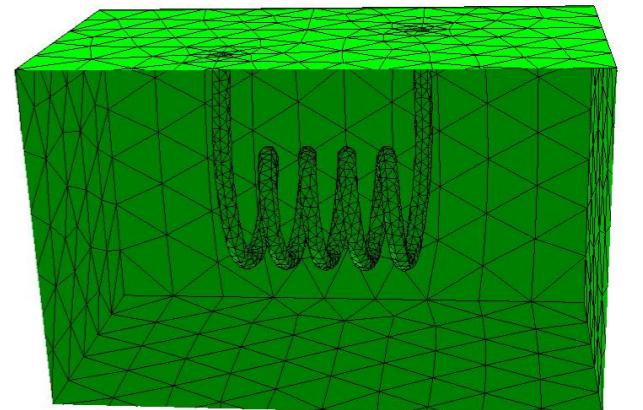
$$I = \text{Ind}(i), \quad J = \text{Ind}(j), \quad E_{ij} \text{ is a fine grid edge}$$



The 2-Level de Rham diagram:

$$\begin{array}{ccccccc}
 V^v & \xrightarrow{B_\nabla} & V^e & \xrightarrow{B_{\text{curl}}} & V^f & \xrightarrow{B_{\text{div}}} & V^c \\
 \downarrow \Pi^W & & \downarrow \Pi^V & & \downarrow \Pi^Q & & \downarrow \Pi^S \\
 V_{\text{coarse}}^v & \xrightarrow{B_\nabla} & V_{\text{coarse}}^e & \xrightarrow{B_{\text{curl}}} & V_{\text{coarse}}^f & \xrightarrow{B_{\text{div}}} & V_{\text{coarse}}^c
 \end{array}$$

- The algebraically constructed coarse spaces form a exact sequence. Thus, Hiptmair / Arnold+Falk+Winther smoothers can be applied.
- There are commuting interpolation operators. This is essential for the two-level analysis.
- Don't change the topology, i.e. don't close tunnels (weak/strong connections) ! Example from T. Kolev, LLNL



Model Problem

$$\Omega = (0, 1)^3, V = H_0(\text{curl}), f = (1, 0, 0),$$

Variational form:

$$\int \text{curl } u \text{ curl } v \, dx + 10^{-3} \int uv \, dx = \int fv \, dx$$

Variable V cycle:

N_h^e	setup (sec)	solver (sec)	iteration
4184	0.15	0.31	11
31024	1.32	6.21	15
238688	11.39	63.93	17

Computation with Stefan Reitzinger's AMG code Pebbles, CPU = PIII 1 GHz (2002)

TEAM 20 Benchmark problem

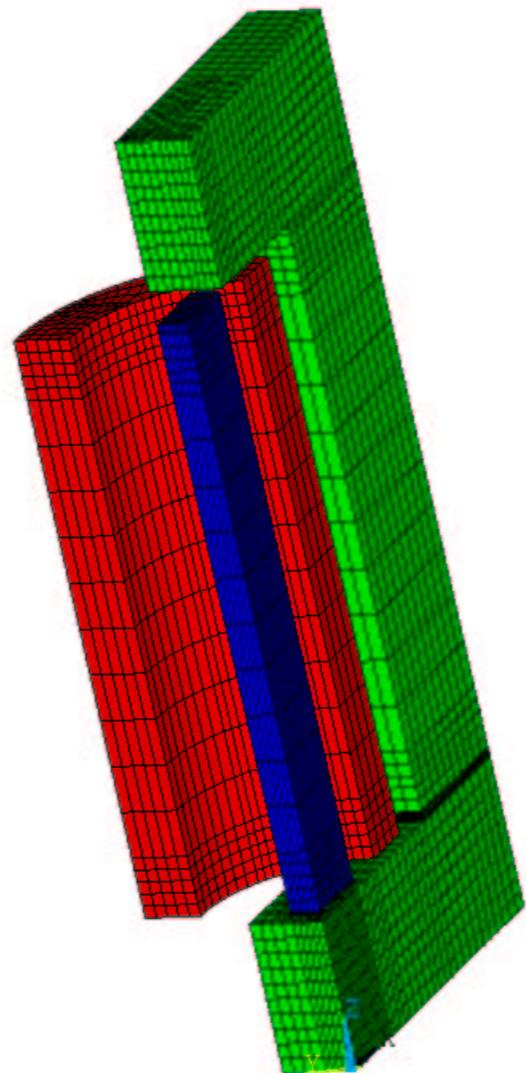
Coil and Iron core, small air gap.

Unknowns: 240E3

Iterations: 26

Solution time: 90 sec

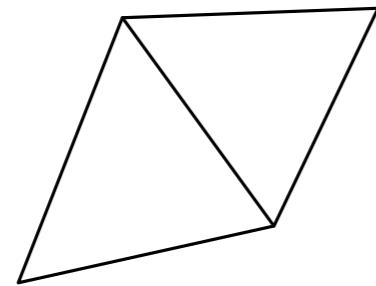
Computations by Manfred Kaltenbacher,
University Erlangen, Germany
Using the code Pebbles (2003)



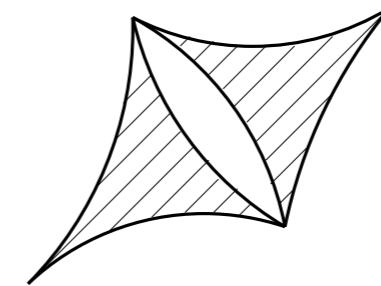
Parallel BDDC Preconditioning

[**Dohrmann**, Mandel, Tezaur, Li, **Widlund**, Tu, Brenner, Sung, Klawonn, Pavarino, Rheinbach, ... 2003+]

Model problem: Conforming high order finite elements for 2D Laplace:



continuous fe-space V
matrix A



discontinuous fe-space \tilde{V} with vertex constraints
matrix \tilde{A}

averaging operator $R : \tilde{V} \rightarrow V$

BDDC preconditioning action:

$$C_{BDDC}^{-1} = R \tilde{A}^{-1} R^t$$

averaging at interfaces is done such that $u - Ru$ is discrete harmonic

BDDC - Analysis

There holds

$$\kappa(C_{BDDC}^{-1}A) = \|R\|_{\tilde{A} \rightarrow A}^2$$

BDDC for p-version $H(\text{curl})$: Keep \mathcal{N}_0^{II} dofs continuous

Robust in κ and h , but grows with $\log^2 p$.

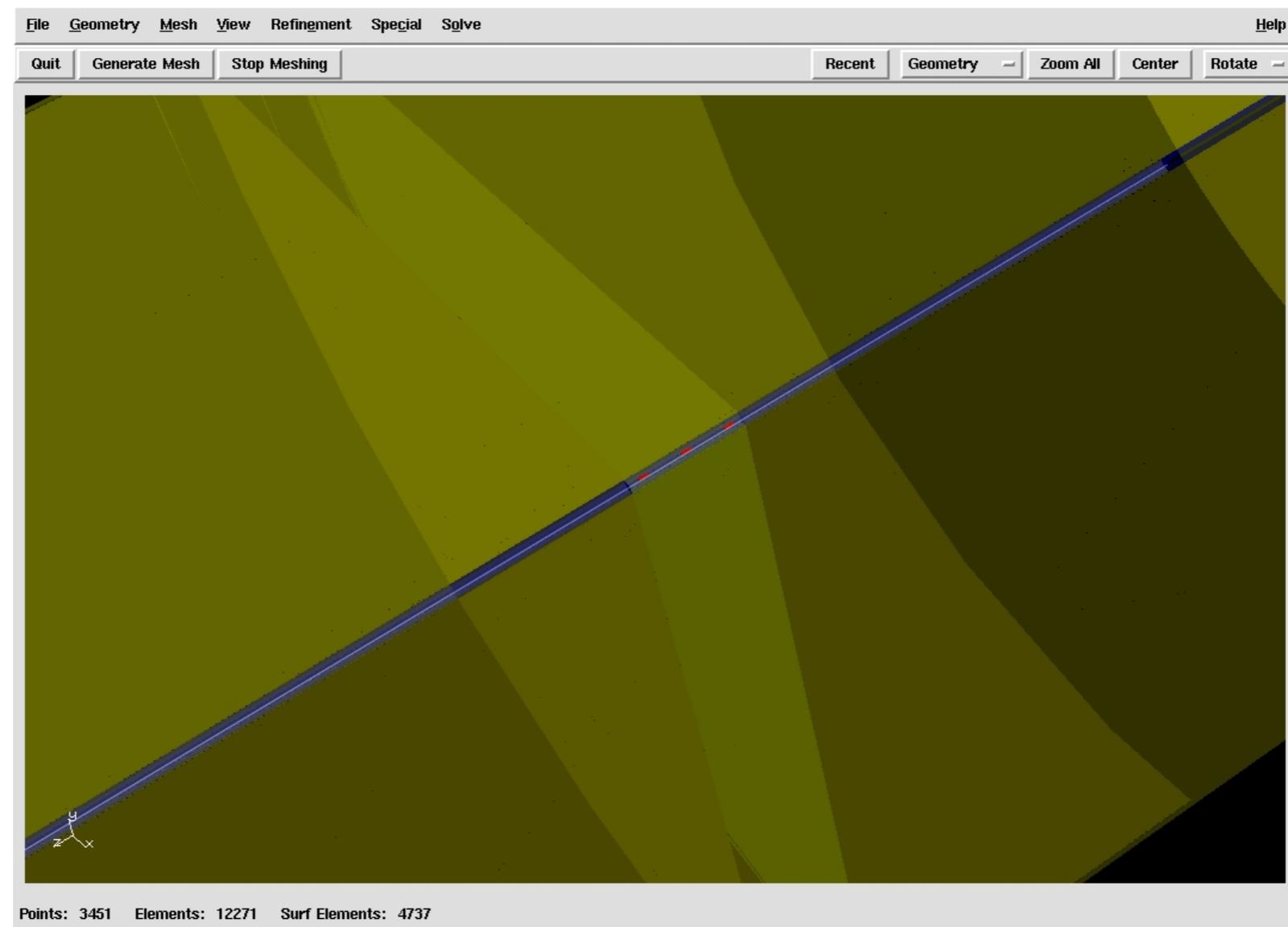
BDDC timings

- Poisson equation, HDG discretization, BDDC preconditioner
- Unstructured mesh of 496 K tets
- Coarse space: mean value on facets (1.01 M), solved with MUMPS, $\text{eps} = 10^{-8}$
- On 4x10 core server, 512 GB shared memory

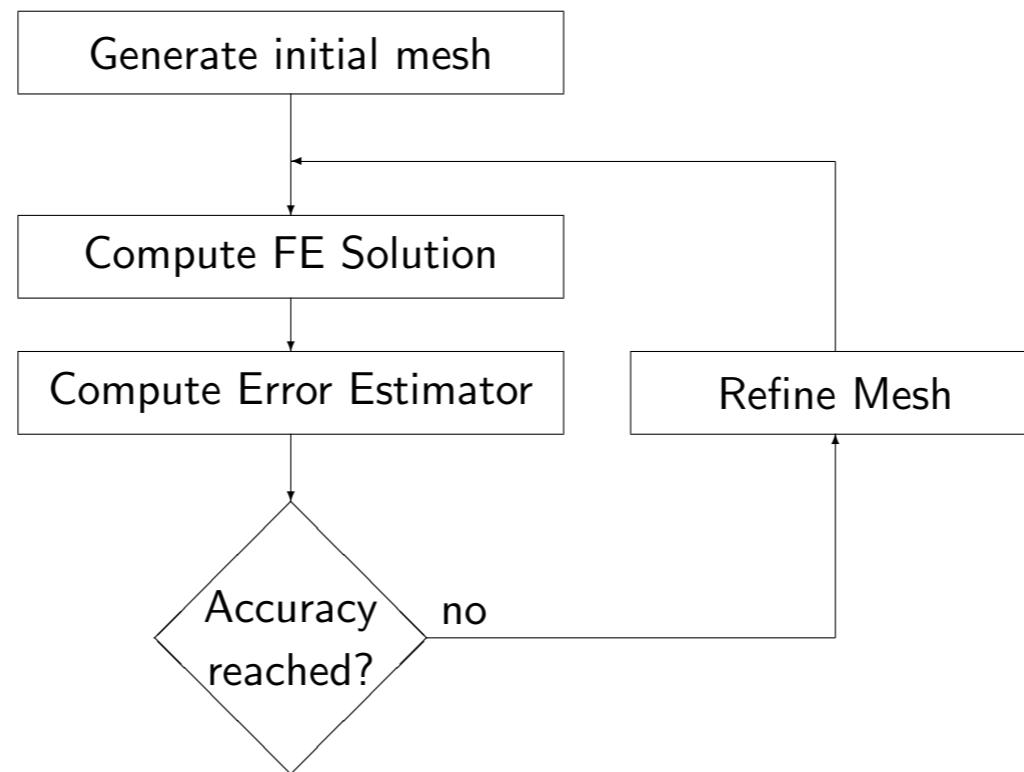
order	N	its	time[sec]	N / (time × cores)
2	11.1 M	38	15.4	18 K
3	20.0 M	47	23.9	20 K
4	32.5 M	55	43.6	18 K
5	49.1 M	61	69.7	17 K
6	70.0 M	66	121	14 K
7	96.0 M	70	196	12 K

[JS - C. Lehrenfeld 2012] cond number estimate $O(\log^3 p)$

Bore-hole Electromagnetics



Local mesh refinement based on a posteriori error estimators



Energy norm error estimators

Infinite dimensional variational problem: Find $A \in V$ such that

$$B(A, v) = f(v) \quad \forall v \in V,$$

Finite element problem: Find $A_h \in V_h$ such that

$$B(A_h, v_h) = f(v_h) \quad \forall v_h \in V_h,$$

Element-wise energy norm error estimators:

$$\eta^2(A_h) = \sum_T \eta_T^2(A_h),$$

such that

$$\|A - A_h\|_B \simeq \eta(A_h).$$

Goal driven error estimates

One is interested in some quantity $y(A)$ depending on the solution, e.g., the closed loop voltage. R. Rannacher's feedback ee and T. Oden's goal driven ee focus on the error

$$y(A) - y(A_h)$$

The key is to define the dual problem with the functional as r.h.s:

$$B(w, v) = y(v)$$

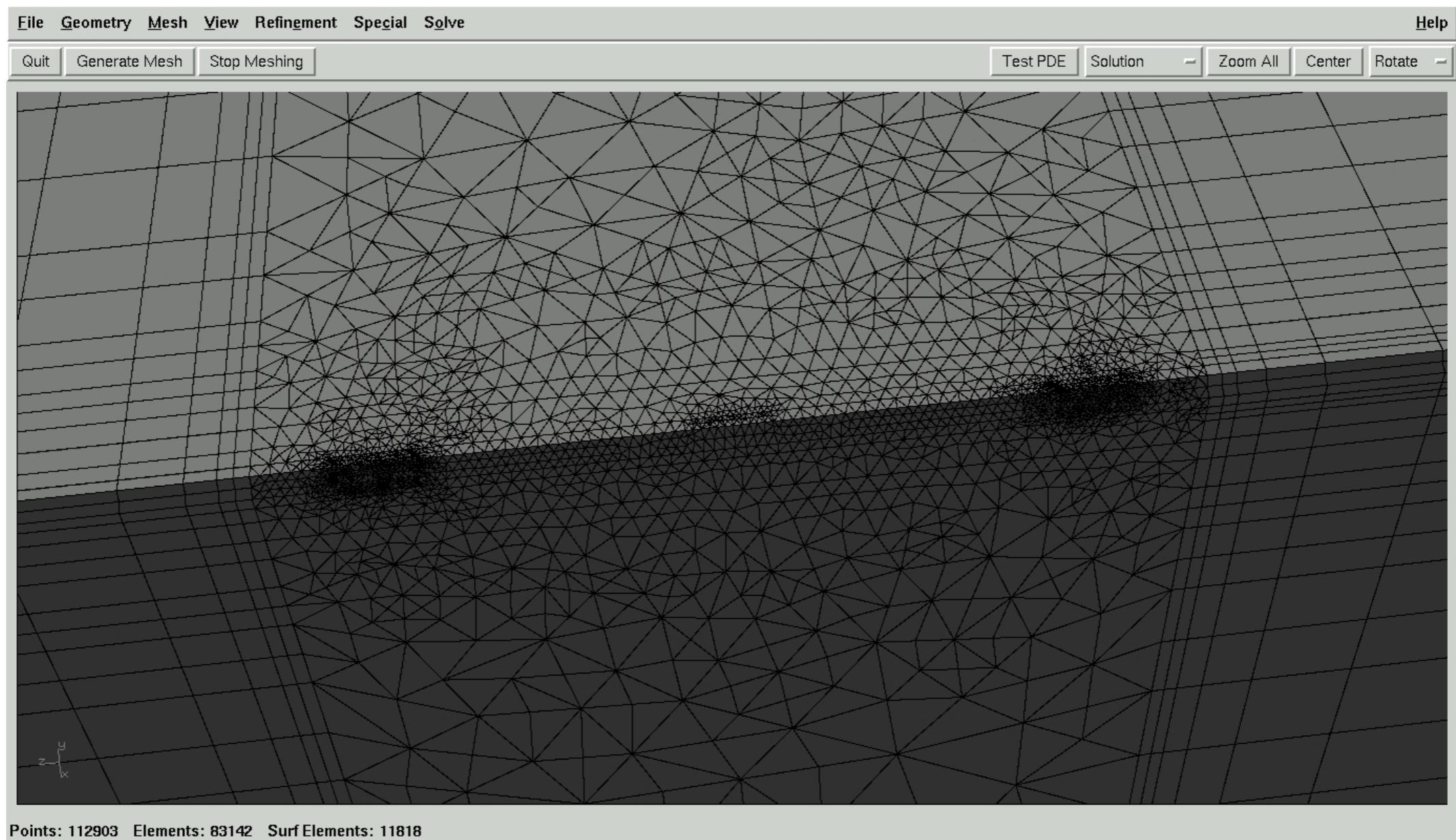
One observes

$$y(A) - y(A_h) = y(A - A_h) = B(w, A - A_h) = B(w - w_h, A - A_h)$$

One version of the goal driven error estimator is

$$y(A) - y(A_h) \simeq \sum_T \eta_T(w_h) \eta_T(A_h)$$

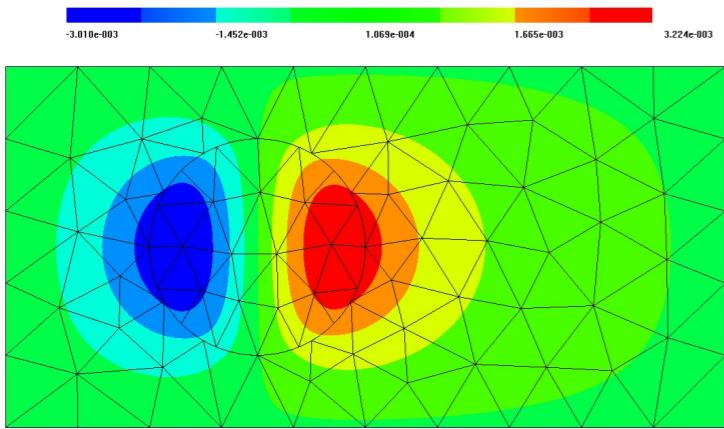
Goal: Compute closed loop voltage in coil 3



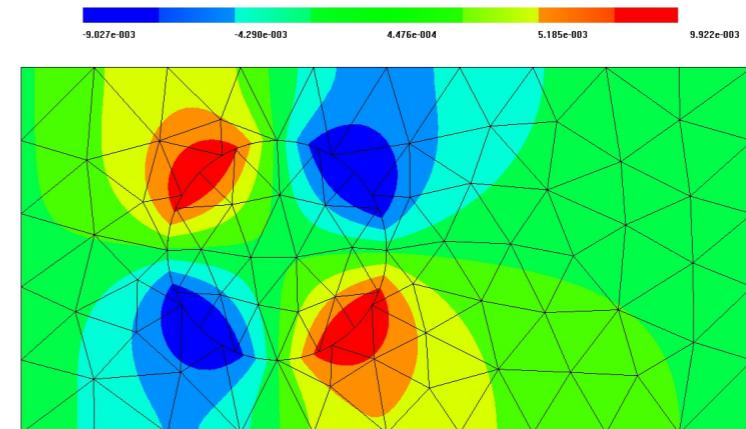
Coupling on non-matching meshes

$f = x$ in circle, else $f = 0$.

Solution u :



Solution $\partial u / \partial x$:



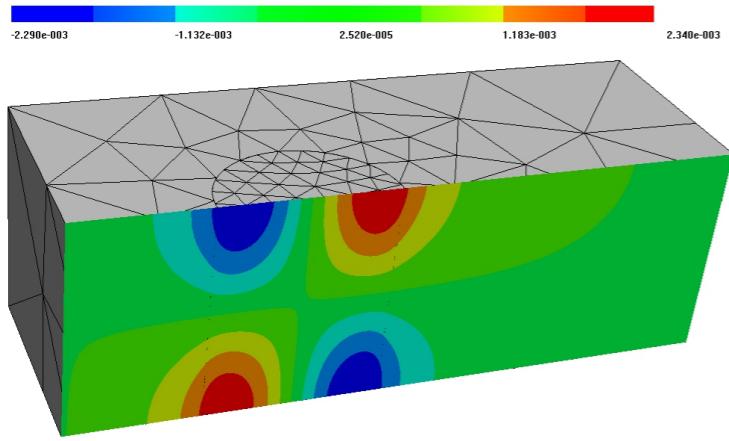
Finite element order $p = 5$.

[Thesis Feldengut, 2010], [Hollaus,Feldengut,JS,Wabro,Omeragic 2011]

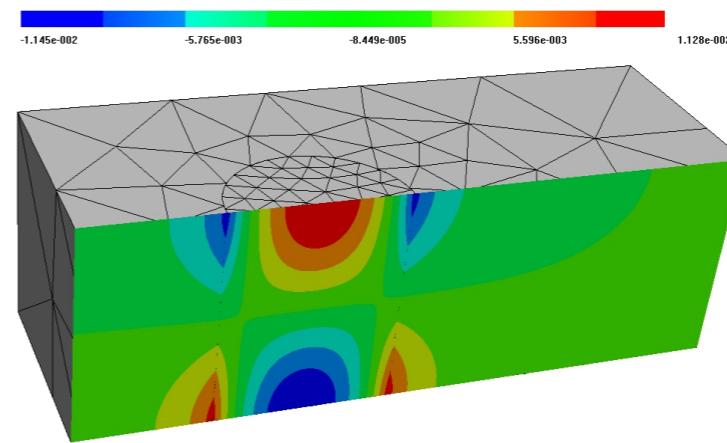
Numerical experiments in 3D

$f = xz$ in cylinder, else $f = 0$.

Solution u :



Solution $\partial u / \partial x$:



Finite element order $p = 4$.

Nitsche / Discontinuous Galerkin method

Allows discontinuous approximation by keeping extra boundary terms:

Find $u \in H_{0,D}^1(\Omega_1) \times H_{0,D}^1(\Omega_1)$ such that

$$\int_{\Omega_1 \cup \Omega_2} \nabla u \cdot \nabla v - \frac{1}{2} \int_{\Gamma} \partial_n u [v] - \frac{1}{2} \int_{\Gamma} \partial_n v [u] + \alpha \int_{\Gamma} [u] [v] = \int f v \quad \forall v$$

with soft penalty term $\alpha \sim p^2/h$ sufficiently large.

No extra stability condition is required.

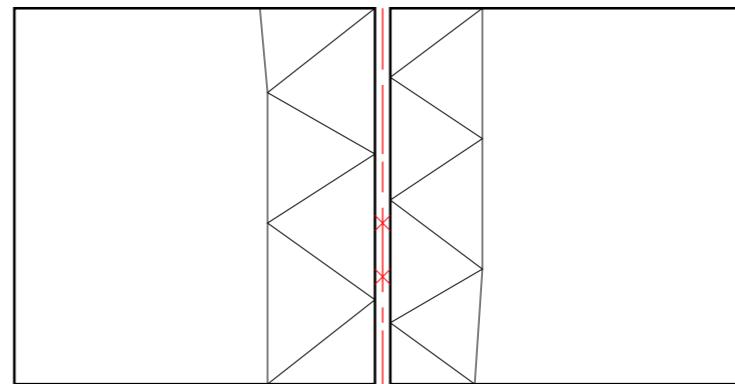
Leads to a symmetric positive definite stiffness matrix.

Integration of boundary terms

Both methods require to compute integrals of finite element functions from different meshes:

Mortar method: $\int_{\Gamma} u_2 \mu$

Nitsche method: $\int_{\Gamma} v_1 \partial_n u_2$



Requires the calculation of an intersection mesh

Complicated implementation, in particular on curved interfaces in 3D

Hybrid Nitsche method - derivation

Introduce a new variable for the primal unknown on the interface:

$$\lambda := u|_{\Gamma}$$

Multiply by test-functions, and integrate by parts:

$$\sum_i \int_{\Omega_i} \nabla u \cdot \nabla v - \int_{\Gamma} \partial_n u v = \int f v \quad \forall v \in H^1(\Omega_1) \times H^1(\Omega_2)$$

Use continuity of $\partial_n u$ and introduce single-valued test function μ on interface:

$$\sum_i \int \nabla u \cdot \nabla v - \int_{\Gamma} \partial_n u (v - \mu) = \int f v \quad \forall v \forall \mu$$

Use $u = \lambda$ on Γ to symmetrize and stabilize with $\alpha \sim p^2/h$.

$$\sum_i \int \nabla u \cdot \nabla v - \int_{\Gamma} \partial_n u (v - \mu) - \int_{\Gamma} \partial_n v (u - \lambda) + \alpha \int_{\Gamma} (u - \lambda)(v - \mu) = \int f v \quad \forall v \forall \mu$$

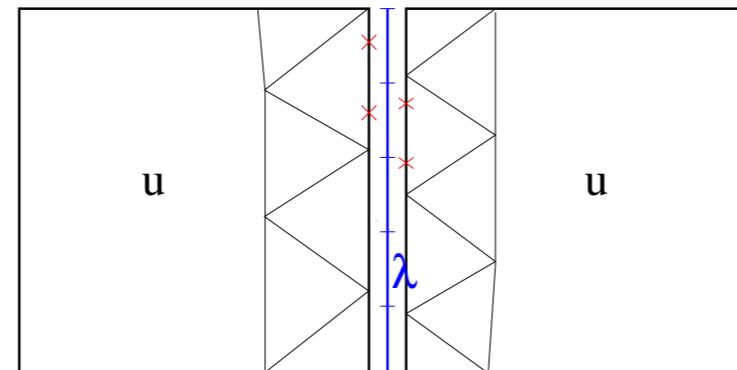
For α chosen right, the discrete formulation is stable independent of the choice of fe spaces for u and λ .

Discretizing and numerical integration

In general, numerical integration is still difficult.

We propose to use smooth B-spline functions for discretizing the hybrid variable λ . This allows

- evaluation in global coordinates
- efficient numerical integration by Gauss-rules on the surface elements



Hybrid Nitsche formulation for Maxwell

proceed as in the scalar case:

$$\int_{\Omega_i} \{\mu^{-1} \operatorname{curl} u \cdot \operatorname{curl} v + \kappa u \cdot v\} + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} u \cdot (v \times n) = \int_{\Omega_i} j \cdot v$$

add symmetry and penalty terms: find (u, λ) such that

$$\sum_{i=1}^2 \left\{ \int_{\Omega_i} \mu^{-1} \{\operatorname{curl} u \cdot \operatorname{curl} v + \kappa u \cdot v\} + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} u \cdot [(v - \mu) \times n] \right. \\ \left. + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} v \cdot [(u - \lambda) \times n] + \frac{\alpha p^2}{\mu h} \int_{\partial\Omega_i} [(u - \lambda) \times n] \cdot [(v - \mu) \times n] \right\} = \int_{\Omega} j \cdot v,$$

where $u, v \in H(\operatorname{curl}, \Omega_1) \times H(\operatorname{curl}, \Omega_2)$, and λ, μ are tangential vector valued fields on the interface.

Overpenalization of gradient fields

The natural energy norm is

$$\|u\|^2 = \mu^{-1} \|\operatorname{curl} u\|_{L_2}^2 + |\kappa| \|u\|_{L_2}^2$$

For gradient fields $u = \nabla\phi$, this norms scales as

$$\|\nabla\phi\|^2 = O(\kappa)$$

This is small for small frequencies/conductivities.

The norm for the Nitsche method is

$$\|(u, \lambda)\|^2 = \sum_{i=1}^2 \left\{ \mu^{-1} \|\operatorname{curl} u\|_{\Omega_i}^2 + \kappa \|u\|_{\Omega_i}^2 + \alpha \mu^{-1} \|(u - \lambda) \times n\|_{\Gamma}^2 \right\}$$

But, the last term of this norm does not scale with κ for gradient fields.

Thus, the penalty term $\|u - \lambda\|$ leads to an overpenalization of the jump for gradient fields.

Scalar potential at the boundary

Goal: Want to replace the continuity condition

$$(u_i - \lambda) \times n_i = 0 \quad i = 1, 2$$

by

$$\begin{aligned} (u_i - \nabla \phi_i - \lambda) \times n_i &= 0 \\ \phi_i - \phi_\Gamma &= 0 \end{aligned}$$

with arbitrary scalar fields $\phi_1 = \phi_2 = \phi_\Gamma$ on the boundary.

This allows to scale the penalty terms for gradients and rotations differently.

Variational formulation

$$\sum_{i=1}^2 \left\{ \int_{\Omega_i} \{\mu^{-1} \operatorname{curl} u \cdot \operatorname{curl} v + \kappa u v\} + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} u [(v - \mu) \times n] + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} v [(u - \nabla\phi - \lambda) \times n] + \alpha \int_{\partial\Omega_i} \mu^{-1} [(u - \nabla\phi - \lambda) \times n][(v - \nabla\psi - \mu) \times n] + \alpha \int_{\partial\Omega_i} \kappa(\phi - \phi_\Gamma)(\psi - \psi_\Gamma) \right\} = \int_{\Omega} j v$$

A boundary identity

Testing the weak form with $v = \nabla\psi$ gives

$$\int_{\Omega_i} \kappa u \cdot \nabla\psi + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} u \cdot (\nabla\psi \times n) = \int_{\Omega} j \cdot \nabla\psi$$

Taking the divergence in the strong form, and integrating by parts leads to

$$\begin{aligned} \int_{\Omega_i} \operatorname{div}(\kappa u) \psi &= \int_{\Omega_i} \operatorname{div} j \psi \\ - \int_{\Omega_i} \kappa u \cdot \nabla\psi + \int_{\partial\Omega_i} \kappa u_n \psi &= - \int_{\Omega_i} j \cdot \nabla\psi + \int_{\partial\Omega_i} j_n \psi \end{aligned}$$

Adding up leads to the boundary relation

$$\int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} u \cdot (\nabla\psi \times n) + \int_{\partial\Omega_i} \kappa u_n \psi = \int_{\partial\Omega_i} j_n \psi$$

Final variational formulation

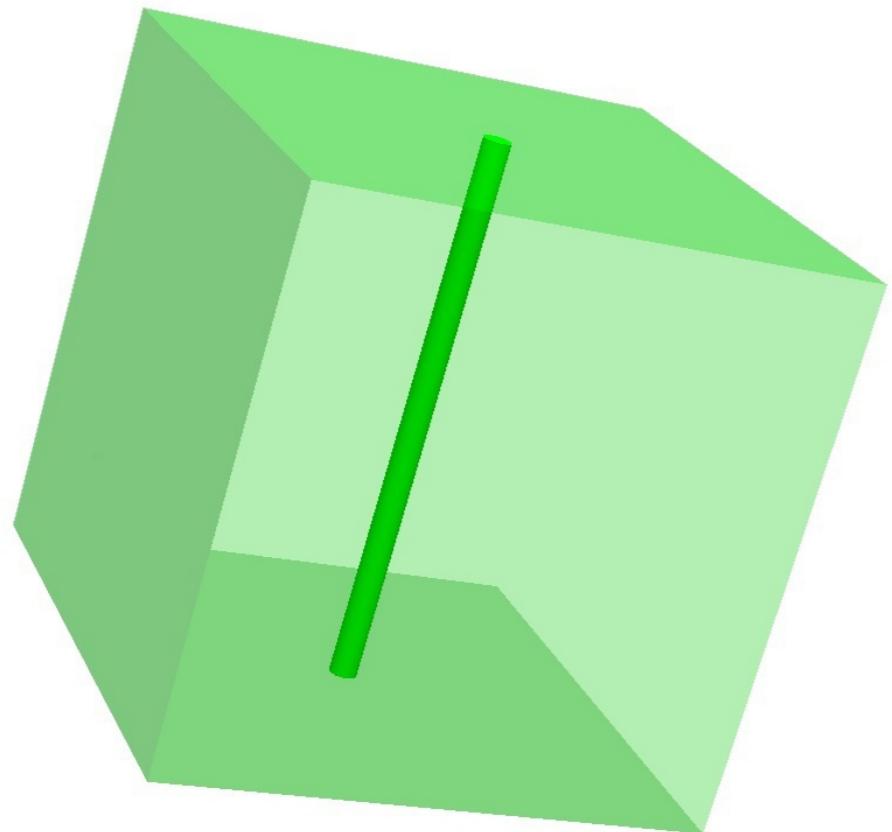
$$\begin{aligned}
& \sum_{i=1}^2 \left\{ \int_{\Omega_i} \{\mu^{-1} \operatorname{curl} u \cdot \operatorname{curl} v + \kappa u v\} + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} u [(v - \nabla\psi - \mu) \times n] \right. \\
& \quad + \int_{\partial\Omega_i} \mu^{-1} \operatorname{curl} v [(u - \nabla\phi - \lambda) \times n] + \alpha \int_{\partial\Omega_i} \mu^{-1} [(u - \nabla\phi - \lambda) \times n][(v - \nabla\psi - \mu) \times n] \\
& \quad \left. - \int_{\partial\Omega_i} \kappa u_n (\psi - \psi_\Gamma) - \int_{\partial\Omega_i} \kappa v_n (\phi - \phi_\Gamma) + \alpha \int_{\partial\Omega_i} \kappa (\phi - \phi_\Gamma) (\psi - \psi_\Gamma) \right\} = \\
& \quad \sum_{i=1}^2 \left\{ \int_{\Omega_i} j v - \int_{\partial\Omega_i} j_n \psi \right\}
\end{aligned}$$

- $u, v \dots H(\operatorname{curl})$ conforming element basis functions on Ω_i
- $\phi, \psi \dots H^1$ conforming element basis functions on $\Omega_i \cap \Gamma$
- $\lambda, \mu \dots$ tangential vector valued spline functions on Γ
- $\phi_\Gamma, \psi_\Gamma \dots$ scalar spline functions on Γ

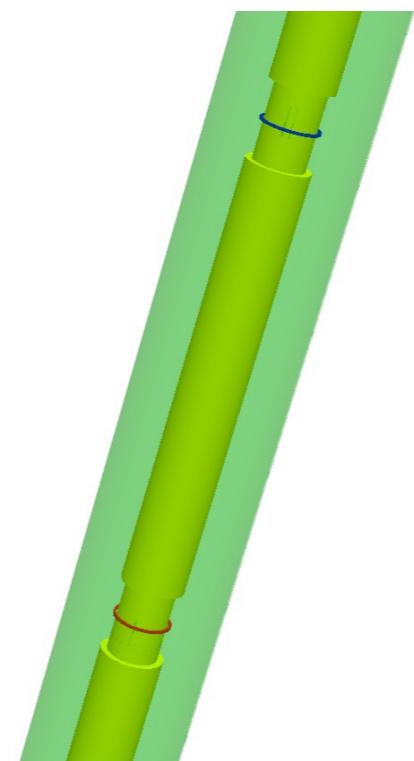
LWD-Tool

project with Schlumberger

borehole with soil

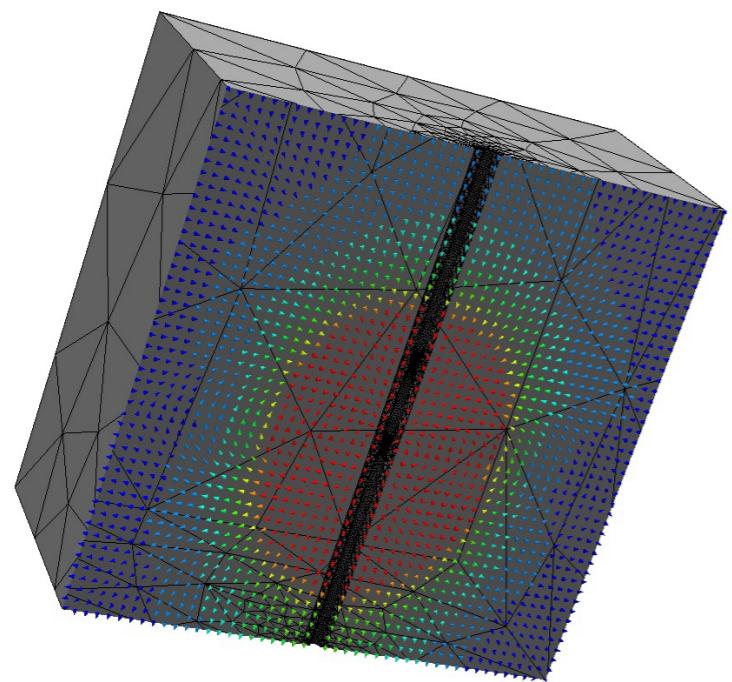


tool with antennas

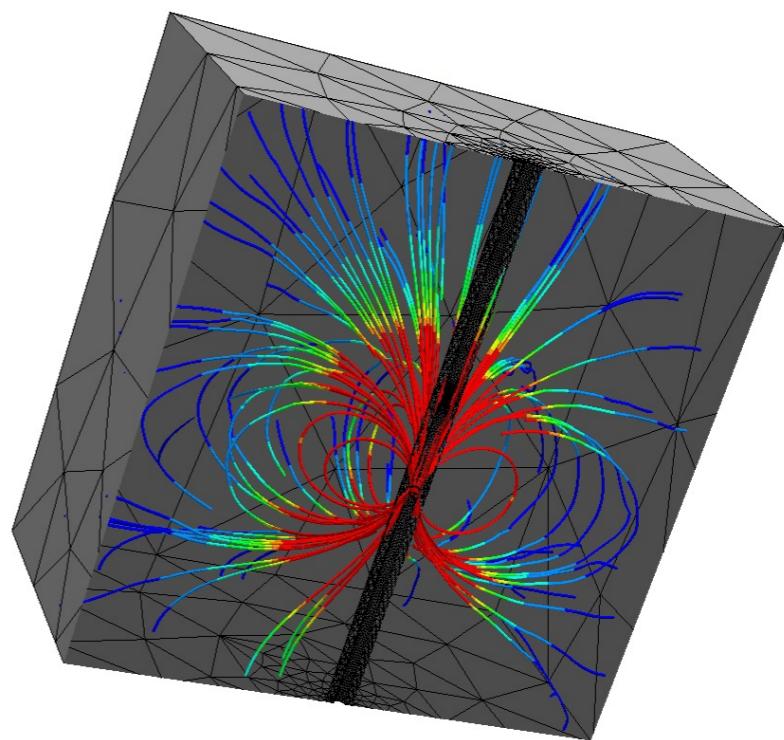


LWD-Tool

B-field



B-field, field-lines:



LWD-Tool

Numerical results for first order elements:

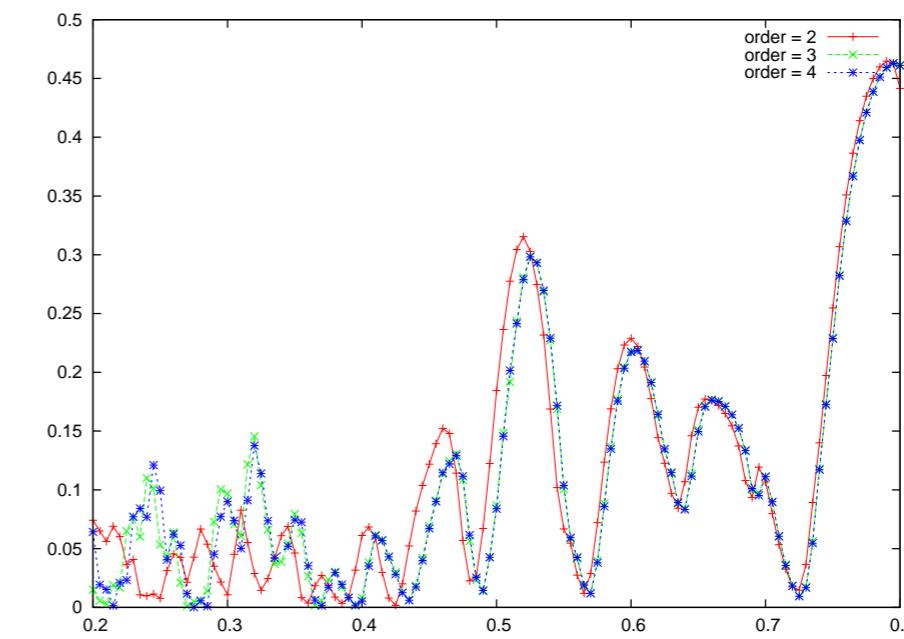
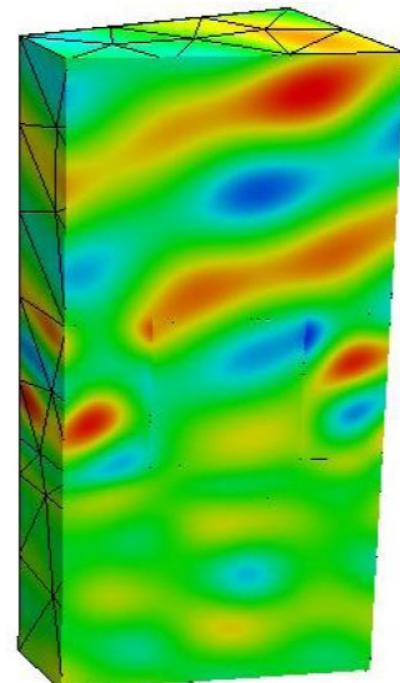
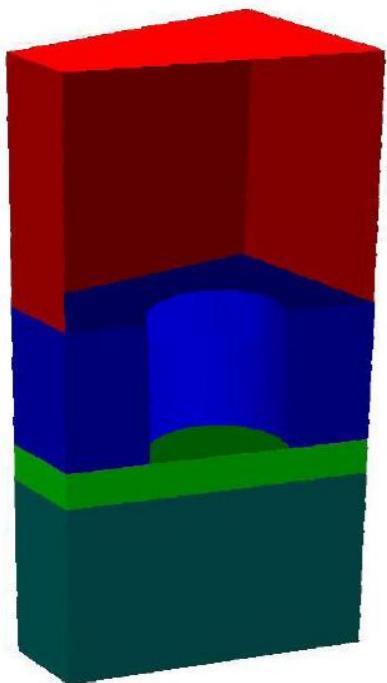
frequency [kHz]	standard, rec volt [nV] 185 810 dofs	Nitsche, rec volt [nV] 195 383 dofs
20	$25.44 - i 18.38$	$25.43 - i 18.37$
100	$71.68 - i 197.5$	$71.65 - i 197.3$
400	$124.9 - i 963.0$	$124.9 - i 962.3$
2000	$-635.9 - i 5295$	$-634.8 - i 5255$

Numerical results for second order elements:

frequency [kHz]	standard, rec volt [nV] 733 881 dofs	Nitsche, rec volt [nV] 736 939 dofs
20	$24.99 - i 18.47$	$24.98 - i 18.47$
100	$70.25 - i 196.7$	$70.23 - i 196.7$
400	$121.7 - i 957.9$	$121.7 - i 957.7$
2000	$-648.1 - i 5256$	$-647.9 - i 5255$

Diffraction on bi-periodic gratings

- Weak coupling of hp-FEM and plane waves by the method of Nitsche / Discontinuous Galerkin
- Homogeneous layers below the grating
- Compute the intensity of the 0^{th} order reflection for varying λ .



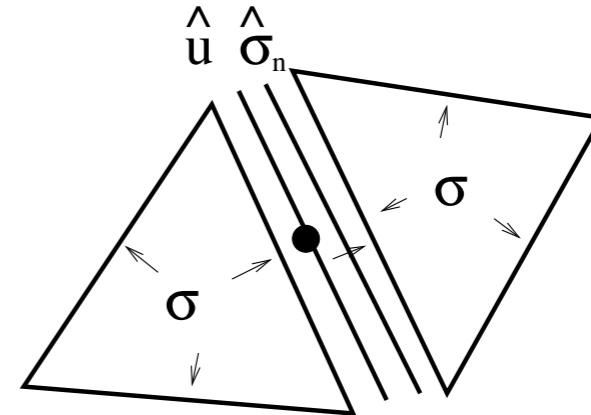
[M. Huber, J. Schöberl, A. Sinwel, S. Zaglmayr: SISC 2009]

Hybrid Discretization of the Helmholtz equation

Add facet variables

$$\hat{u} := u|_F \quad \hat{\sigma}_n = \sigma|_F \cdot n_F$$

and stabilize by adding $0 = \int_{\partial T} (\sigma_n - \hat{\sigma}_n)(\tau_n - \hat{\tau}_n)$



$$\begin{aligned} \sum_T \frac{i}{\omega} \int_T \operatorname{div} \sigma \operatorname{div} \tau - i\omega \int_T \sigma \tau + \int_{\partial T} \sigma_n \tau_n &+ \sum_T \int_{\partial T} \tau_n \hat{u} - \tau_n \hat{\sigma}_n = 0 \quad \forall \tau \\ \sum_T \int_{\partial T} \sigma_n \hat{v} - \sigma_n \hat{\tau}_n &+ \sum_T \int_{\partial T} \hat{\sigma}_n \hat{\tau}_n + \int_{\partial \Omega} \hat{u} \hat{v} = \int_{\partial \Omega} g_{in} \hat{v} \quad \forall \hat{v} \end{aligned}$$

The element-variable σ can be eliminated from the stable local equation (Robin - b.c.):

$$\frac{i}{\omega} \int_T \operatorname{div} \sigma \operatorname{div} v - i\omega \int_T \sigma \tau + \int_{\partial T} \sigma_n \tau_n = \int_{\partial T} (\hat{\sigma}_n - \hat{u}) \tau_n$$

[Monk-Sinwel-Schöberl], [Huber-Schöberl], [Phd Huber 2013]

Infrastructure

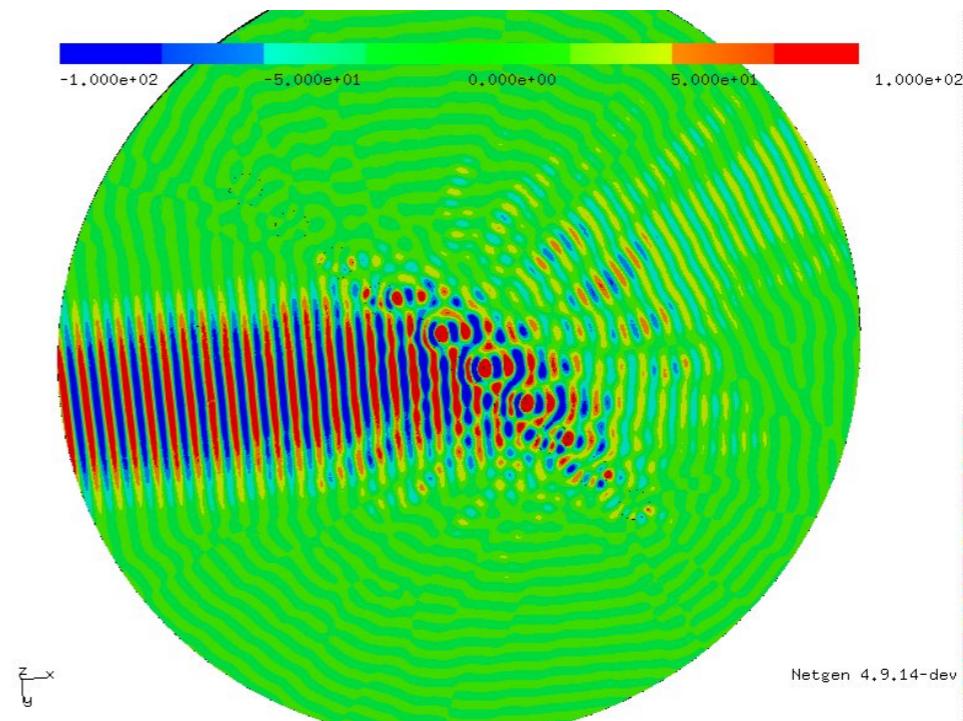
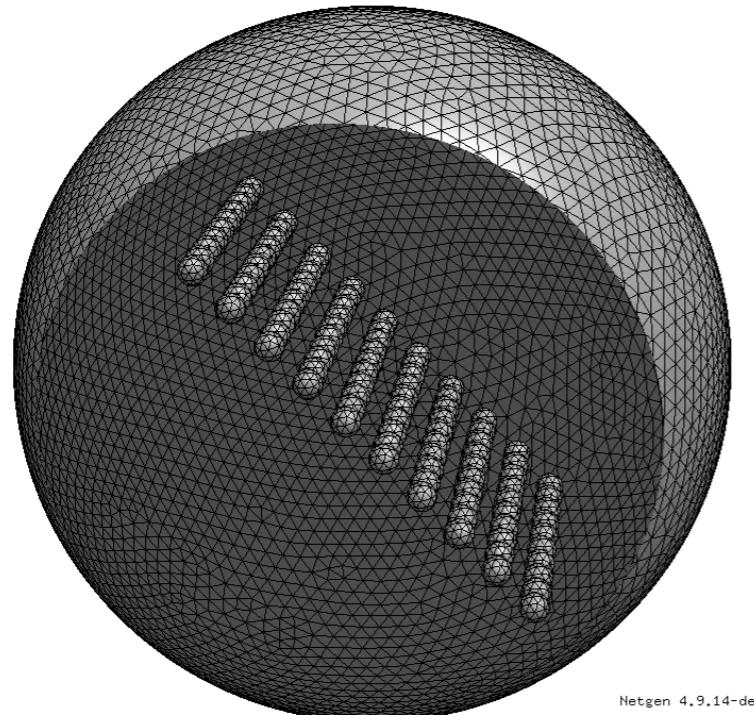
- general purpose hp - Finite Element Code Netgen/NGSolve
C++, open source
MPI - parallel
- Dell R 910 Server (costs: 20k Euro in 2011)
4 Xeon E7 CPUs 10 core @ 2.2 GHz
512 GB RAM shared memory (NUMA)
- Vienna Scientific Cluster 2 (ranked 56 in top500, 2011)
20 000 cores, 2 GB RAM / core

Diffraction from a grating

Sphere with $D = 40\lambda$, $127k$ elements, $h \approx \lambda$, $p = 5$, $39M$ dofs (corresponds to $9.4M$ primal dofs)

78 sub-domains / processes, no coarse grid

$T_{ass} = 9m$, $T_{pre} = 12m$, $T_{solve} = 21m$, 156 its

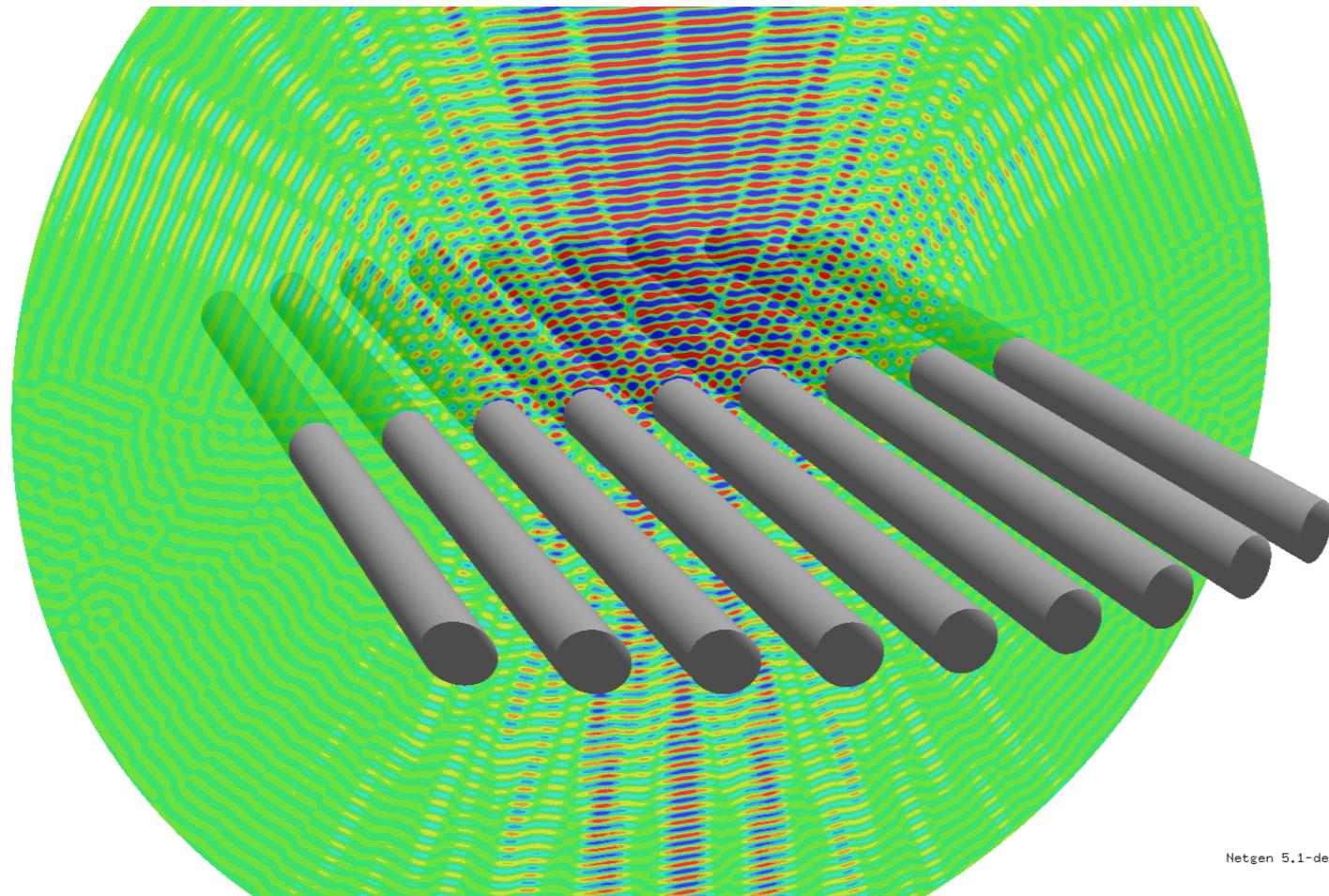


Diffraction from rods

Sphere with $D = 80\lambda$, $2.24M$ elements, $h \approx 0.8\lambda$, $p = 4$, $536M$ dofs

2000 sub-domains / processes, no coarse grid

$T_{ass} = 50s$, $T_{pre} = 30s$, $T_{solve} = 12m$, 496 its



Hybrid Discontinuous Galerkin (HDG) Method

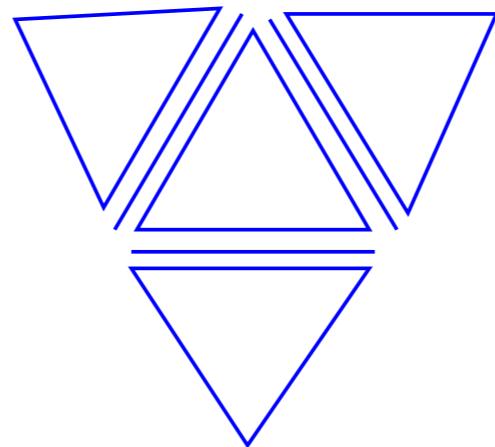
Model problem: $-\Delta u = f$

A mesh consisting of elements and facets (= edges in 2D and faces in 3D)

$$\mathcal{T} = \{T\} \quad \mathcal{F} = \{F\}$$

Goal: Approximate u with piece-wise polynomials on elements and additional polynomials on facets:

$$u_N \in P^p(\cup T) \quad \lambda_N \in P^p(\cup F)$$



HDG - Derivation

Exact solution u , traces on element boundaries: $\lambda := u|_{\cup F}$

Integrate against discontinuous test-functions $v \in H^1(\cup T)$, element-wise integration by parts:

$$\sum_T \left\{ \int_T \nabla u \nabla v - \int_{\partial T} \frac{\partial u}{\partial n} v \right\} = \int_{\Omega} f v$$

Use continuity of $\frac{\partial u}{\partial n}$, and test with single-valued functions $\mu \in L_2(\cup F)$:

$$\sum_T \left\{ \int_T \nabla u \nabla v - \int_{\partial T} \frac{\partial u}{\partial n} (v - \mu) \right\} = \int_{\Omega} f v$$

Use consistency $u = \lambda$ on ∂T to symmetrize, and stabilize ...

$$\sum_T \left\{ \int_T \nabla u \nabla v - \int_{\partial T} \frac{\partial u}{\partial n} (v - \mu) - \int_{\partial T} \frac{\partial v}{\partial n} (u - \lambda) + \frac{\alpha p^2}{h} \int_{\partial T} (u - \lambda)(v - \mu) \right\} = \int_{\Omega} f v$$

Dirichlet b.c.: Imposed on λ , Neumann b.c.: $\int_{\Gamma_N} g \mu$

Relation to standard Interior Penalty DG method

DG - space

$$V_N := P^p(\cup T)$$

Bilinearform

$$A^{DG}(u, v) = \sum_T \left\{ \int_T \nabla u \nabla v - \int_{\partial T} \frac{\partial u}{\partial n}[v] - \int_{\partial T} \frac{\partial v}{\partial n}[u] + \frac{\alpha p^2}{h} \int_{\partial T} [u][v] \right\}$$

Hybrid DG has

- even more unknowns, but less matrix entries
- allows element-wise assembling
- allows static condensation of element unknowns

Hybridization of standard DG methods [Cockburn+Gopalakrishnan+Lazarov]

Comparison to mixed hybrid system

HDG method needs facet variable of one order higher ???

$\lambda \in P^{p-1}(\cup F)$ is enough when inserting a projector:

$$\begin{aligned} A^{HDG}(u, \lambda; v, \mu) = & \sum_T \left\{ \int_T \nabla u \nabla v - \int_{\partial T} \frac{\partial u}{\partial n} (v - \mu) \right. \\ & \left. - \int_{\partial T} \frac{\partial v}{\partial n} (u - \lambda) + \frac{\alpha p^2}{h} \int_{\partial T} \Pi^{p-1}(u - \lambda) \Pi^{p-1}(v - \mu) \right\} \end{aligned}$$

Implementation of the projector by an EAS - like method.

Function spaces $H(\text{curl})$ and $H(\text{div})$

$$\begin{aligned} H(\text{curl}) &= \{u \in [L_2]^d : \text{curl } u \in L_2^{d \times d, \text{skew}}\} \\ H(\text{div}) &= \{u \in [L_2]^d : \text{div } u \in L_2\} \end{aligned}$$

Piece-wise smooth functions in

- $H(\text{curl})$ have continuous tangential components,
- $H(\text{div})$ have continuous normal components.

Important for constructing conforming finite elements such as Raviart Thomas, Brezzi-Douglas-Marini, and Nedelec elements.

Natural function space for Maxwell equations: Find $A \in H(\text{curl})$ such that

$$\int_{\Omega} \mu^{-1} \text{curl } A \text{ curl } v + \int_{\Omega} (i\sigma\omega - \varepsilon\omega^2) A v = \int j v \quad \forall v \in H(\text{curl})$$

Mixed Continuous / Hybrid Discontinuous Galerkin method

Vector-valued spaces with partial continuity and partial components on facets:

$$\begin{aligned} V_{\mathcal{T},n} &= \{v \in [P^p(\cup T)]^d : [v_n] = 0\} & V_{\mathcal{T},\tau} &= \{v \in [P^p(\cup T)]^d : [v_\tau] = 0\} \\ V_{\mathcal{F},n} &= \{v \in [P^p(\cup F)]^d : v_\tau = 0\} & V_{\mathcal{F},\tau} &= \{v \in [P^p(\cup F)]^d : v_n = 0\} \end{aligned}$$

$H(\text{curl})$ - based formulation for elasticity: Find $u \in V_{\mathcal{T},\tau}$ and $\lambda \in V_{\mathcal{F},n}$ such that

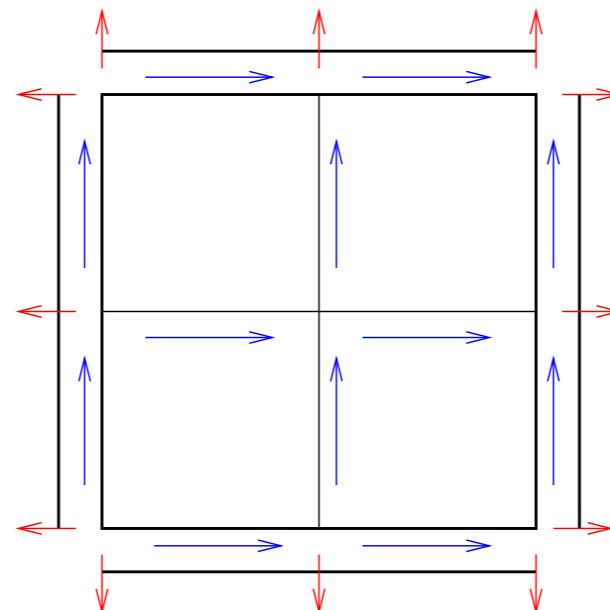
$$A^\tau(u, \lambda; v, \mu) = \int f v \quad \forall v \in V_{\mathcal{T},\tau} \quad \forall \mu \in V_{\mathcal{F},\nu}$$

$$\begin{aligned} A^\tau(u, \lambda; v, \mu) &= \sum_T \left\{ \int_T D\varepsilon(u) : \varepsilon(v) - \int_{\partial T} (D\varepsilon(u))_{nn} (v - \mu)_n \right. \\ &\quad \left. - \int_{\partial T} (D\varepsilon(v))_{nn} (u - \lambda)_n + \frac{\alpha p^2}{h} \int_{\partial T} (u - \lambda)_n (v - \mu)_n \right\} \end{aligned}$$

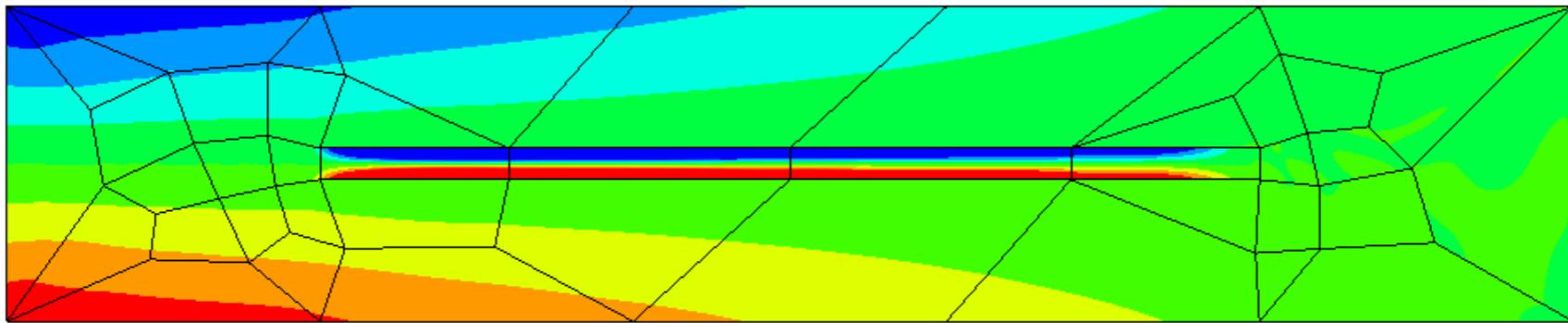
Or, vice versa ...

Degrees of freedom

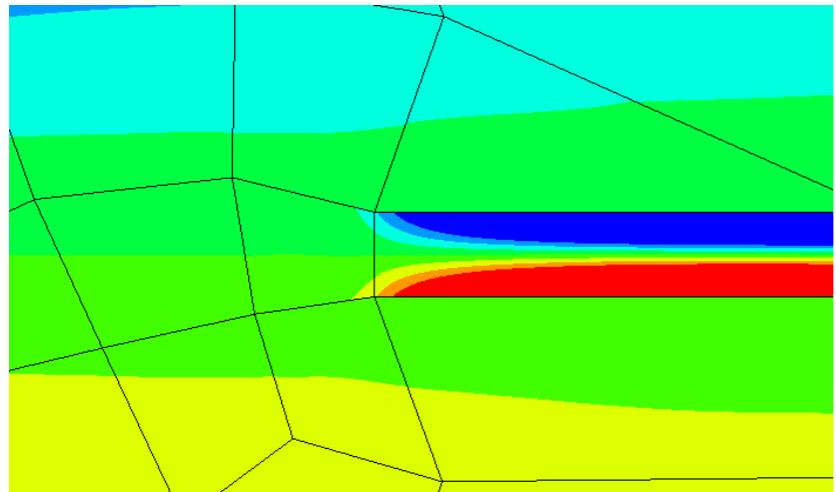
$$u_T \in \mathcal{N}_1 = P^{p,p+1} \times P^{p+1,p}, \quad \lambda_F \in P^{p+1}$$



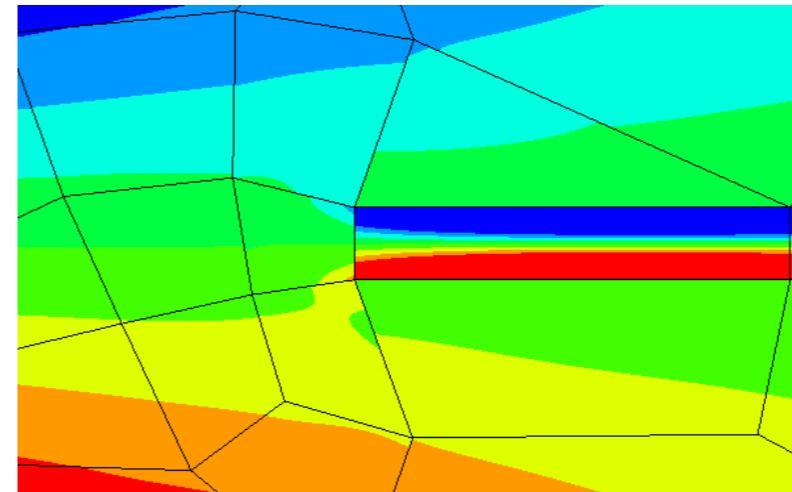
Elasticity: A beam in a beam



Reenforcement with $E = 50$ in medium with $E = 1$.



HDG FEM, $p = 3$



Primal FEM, $p = 3$

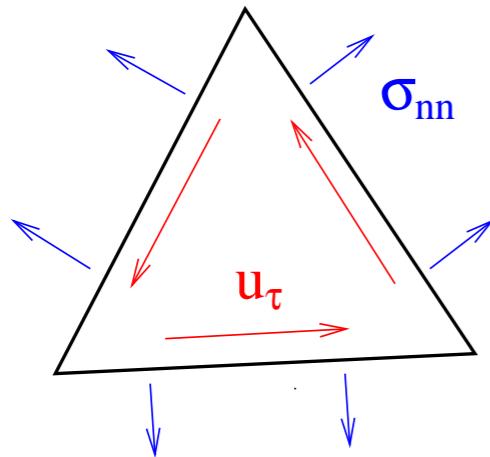
Tangential displacement - normal normal stress continuous mixed method

[Phd thesis Astrid Sinwel (now Pechstein)], [A. Pechstein + JS 2011]

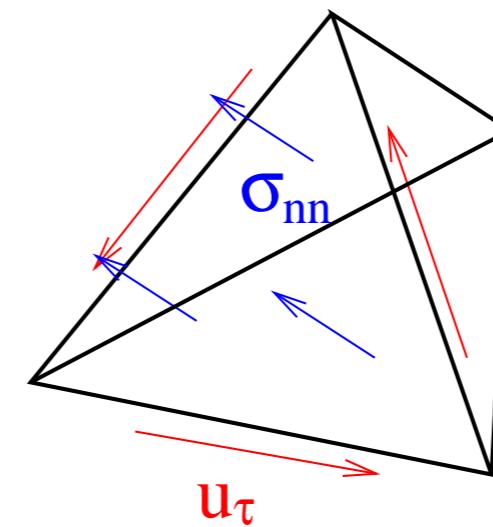
Mixed elements for approximating displacements and stresses.

- tangential components of displacement vector
- normal-normal component of stress tensor

Triangular Finite Element:

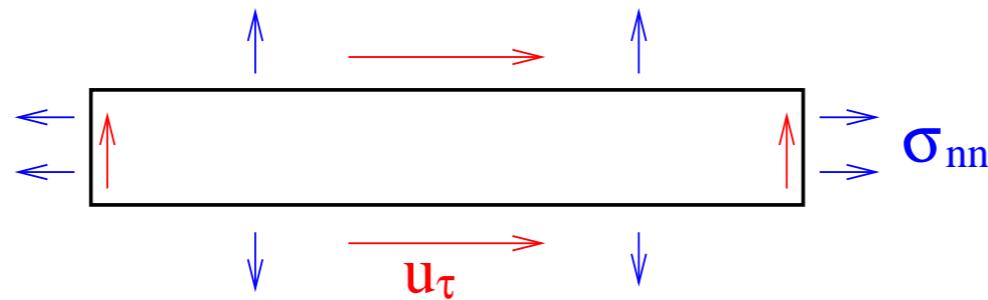


Tetrahedral Finite Element:



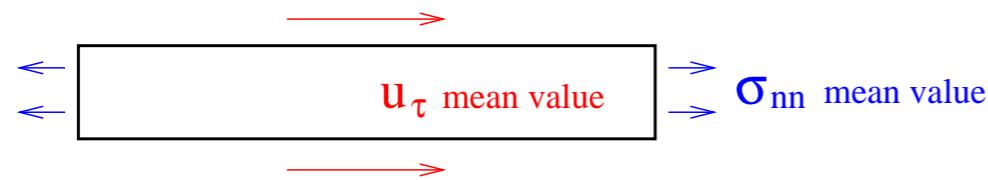
The quadrilateral element

Dofs for general quadrilateral element:

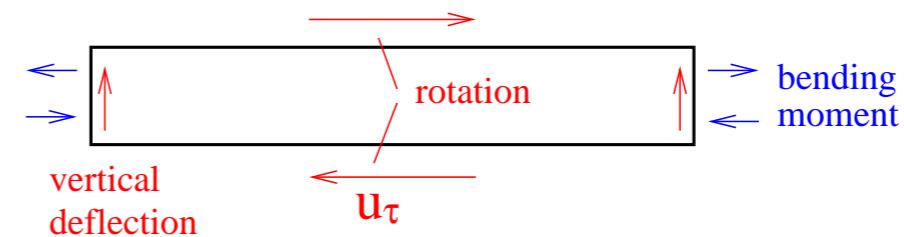


Thin beam dofs ($\sigma_{nn} = 0$ on bottom and top):

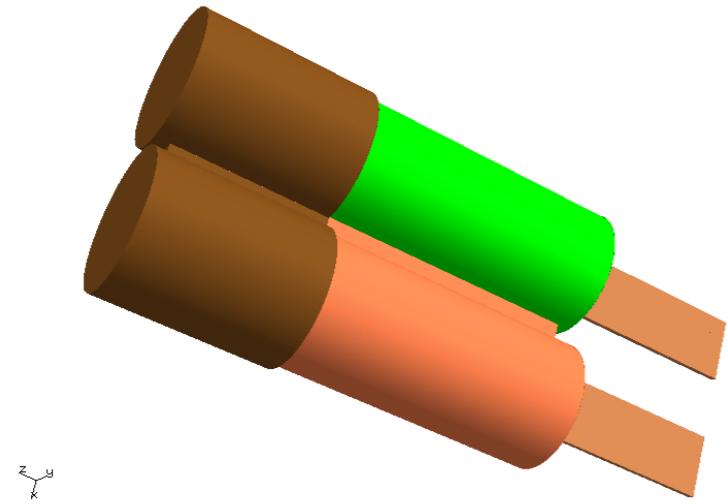
Beam stretching components:



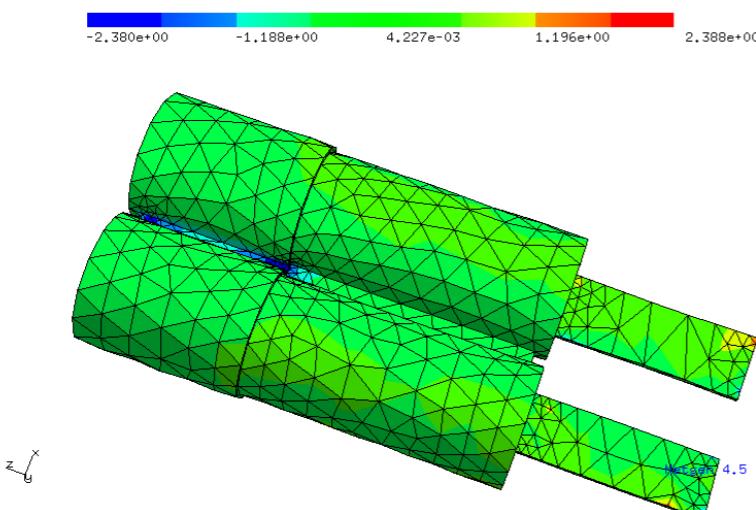
Beam bending components:



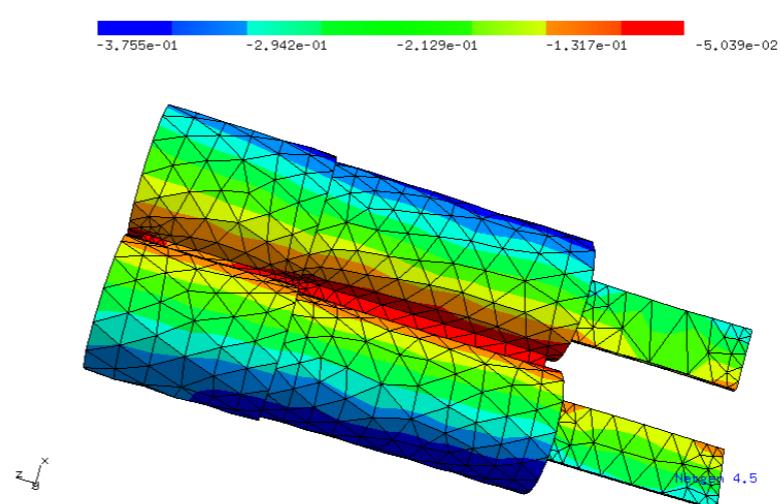
For Hot Days ...



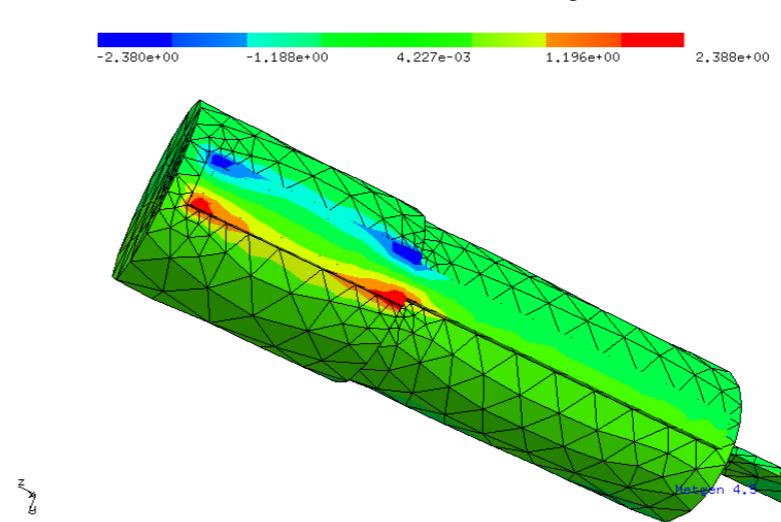
Geometry



Deformed geometry, stress σ_{xx}



Displacement u_y



Interior stress

Anisotropic Estimates

Thm: There holds

$$\sum_T \|\varepsilon(u - u_h)\|_T^2 + \sum_F h_{op}^{-1} \| [u_n] \|_F^2 + \|\sigma - \sigma_h\|^2 \leq c \{ h_{xy}^m \|\nabla_{xy}^m \varepsilon(u)\| + h_z^m \|\nabla_z^m \varepsilon(u)\| \}^2$$

Proof: Stability constants are robust in aspect ratio (for tensor product elements)

Anisotropic interpolation estimates (H^1 : Apel). Interpolation operators commute with the strain operator:

$$\begin{aligned} \|\varepsilon(u - Qu)\|_{L_2} &= \|(I - \tilde{Q})\varepsilon(u)\|_{L_2} \\ &\preceq h_{xy}^m \|\nabla_x^m \varepsilon_{xy,z}(u)\|_0 + h_z^m \|\nabla_z^m \varepsilon_{xy,z}(u)\|_{L_2} \end{aligned}$$

[A. Pechstein + JS, 2012]

Hybridization: Implementation aspects

Both methods are (essentially) equivalent:

- Hybridization of mixed method:

Introduce Lagrange parameter λ_n to enforce continuity of σ_{nn} . Its meaning is the displacement in normal direction.

- Continuous / hybrid discontinuous Galerkin method:

Displacement u is strictly tangential continuous, HDG facet variable (= normal displacement) enforces weak continuity of normal component.

Anisotropic error estimates from mixed methods can be applied for HDG method !

Continuous / hybrid discontinuous Galerkin method for Stokes

(Thesis C. Lehrenfeld 2010, RWTH)

$H(\text{div})$ - based formulation for Stokes:

Find $u \in V_{\mathcal{T},n} \subset H(\text{div})$, $\lambda \in V_{\mathcal{F},\tau}$ and $p \in P^{p-1}(\mathcal{T})$ such that

$$\begin{aligned} A^n(u, \lambda; v, \mu) + \int_{\Omega} \text{div } v q &= \int f v \quad \forall (v, \mu) \\ \int \text{div } u q &= 0 \quad \forall q \end{aligned}$$

Provides exactly divergence-free discrete velocity field u

LBB is proven by commuting interpolation operators for de Rham diagram

[Cockburn, Kanschat, Schötzau 2005]

$H(\text{div})$ -conforming elements for Navier Stokes

$$\begin{aligned}\frac{\partial u}{\partial t} - \operatorname{div}(2\nu\varepsilon(u) - u \otimes u - pI) &= f \\ \operatorname{div} u &= 0 \\ +b.c.\end{aligned}$$

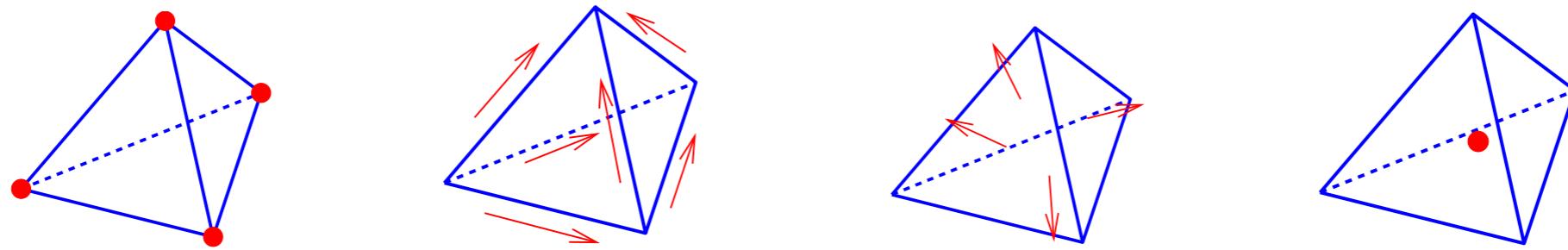
Fully discrete scheme, semi-implicit time stepping (IMEX1)

$$\begin{aligned}\left(\frac{1}{\tau}M + A^\nu\right)\hat{u} + B^T\hat{p} &= f + \frac{1}{\tau}Mu - A^c(u) \\ B\hat{u} &= 0\end{aligned}$$

- u is exactly div-free \Rightarrow non-negative convective term $\int u \nabla v v \geq 0$
- stability for kinetic energy ($\frac{d}{dt}\|u\|_0^2 \leq \frac{1}{\nu}\|f\|_{L_2}^2$)
- convective term by upwinding
- allows kernel-preserving smoothing and grid-transfer for fast iterative solver

The de Rham Complex

$$\begin{array}{ccccccc}
 H^1 & \xrightarrow{\nabla} & H(\text{curl}) & \xrightarrow{\text{curl}} & H(\text{div}) & \xrightarrow{\text{div}} & L^2 \\
 \cup & & \cup & & \cup & & \cup \\
 W_h & \xrightarrow{\nabla} & V_h & \xrightarrow{\text{curl}} & Q_h & \xrightarrow{\text{div}} & S_h
 \end{array}$$



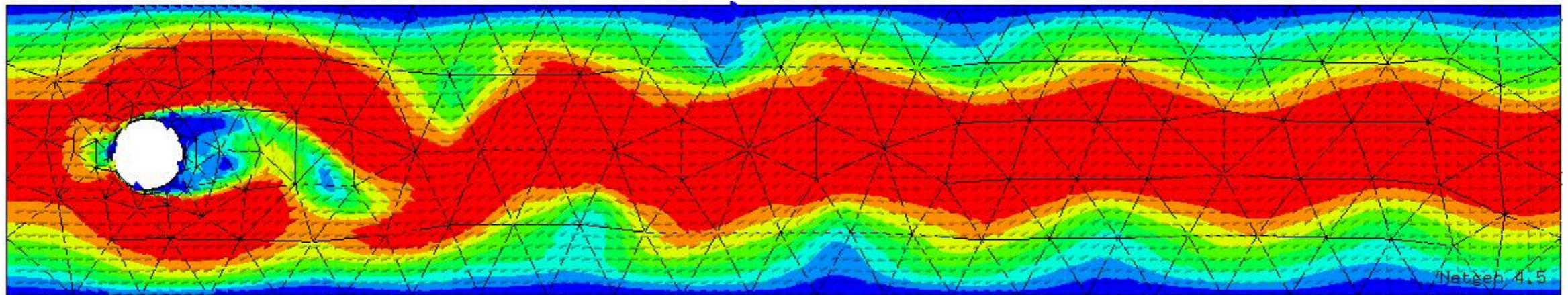
For constructing high order finite elements

$$\begin{aligned}
 W_{hp} &= W_{\mathcal{L}_1} + \text{span}\{\varphi_{h.o.}^W\} \\
 V_{hp} &= V_{\mathcal{N}_0} + \text{span}\{\nabla \varphi_{h.o.}^W\} + \text{span}\{\varphi_{h.o.}^V\} \\
 Q_{hp} &= Q_{\mathcal{R}T_0} + \text{span}\{\text{curl } \varphi_{h.o.}^V\} + \text{span}\{\varphi_{h.o.}^Q\} \\
 S_{hp} &= S_{\mathcal{P}_0} + \text{span}\{\text{div } \varphi_{h.o.}^S\}
 \end{aligned}$$

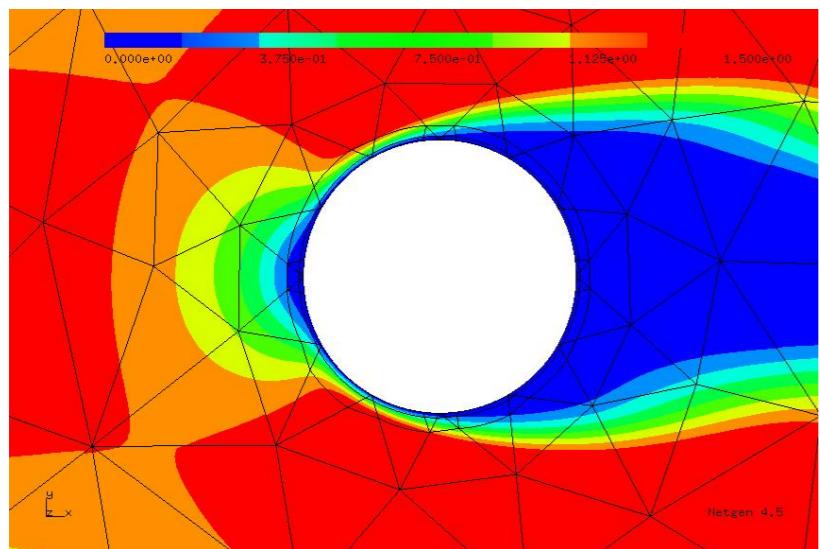
Allows to construct high-order-divergence free elements $\{v \in BDM_k : \text{div } v \in P_0\}$

Flow around a disk, 2D

$Re = 100$, 5th-order elements

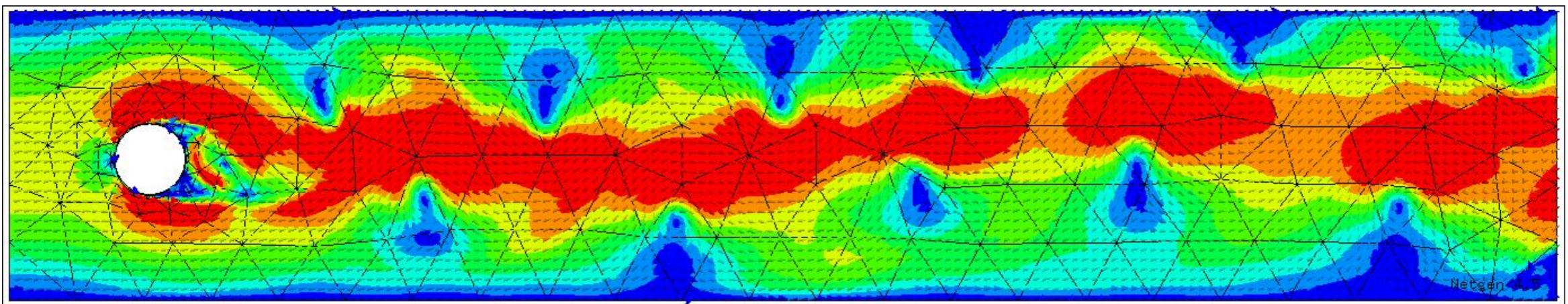


Boundary layer mesh around cylinder:

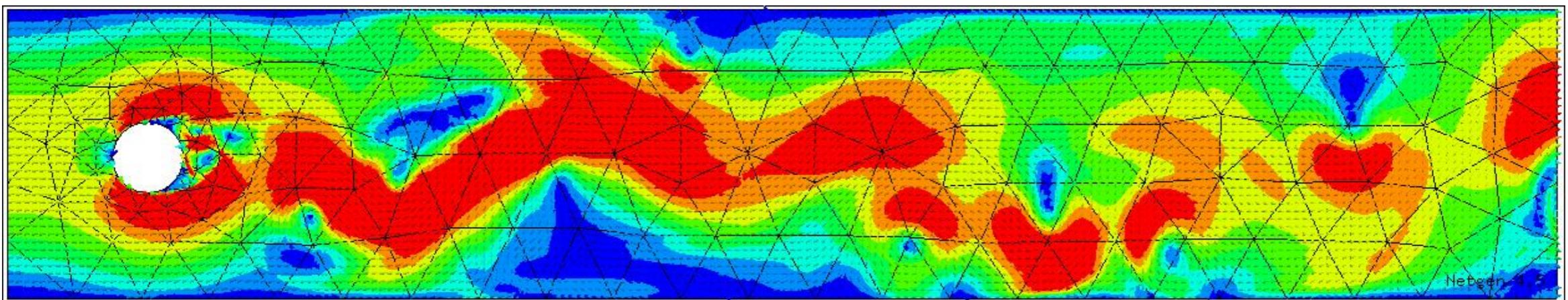


Flow around a disk, 2D

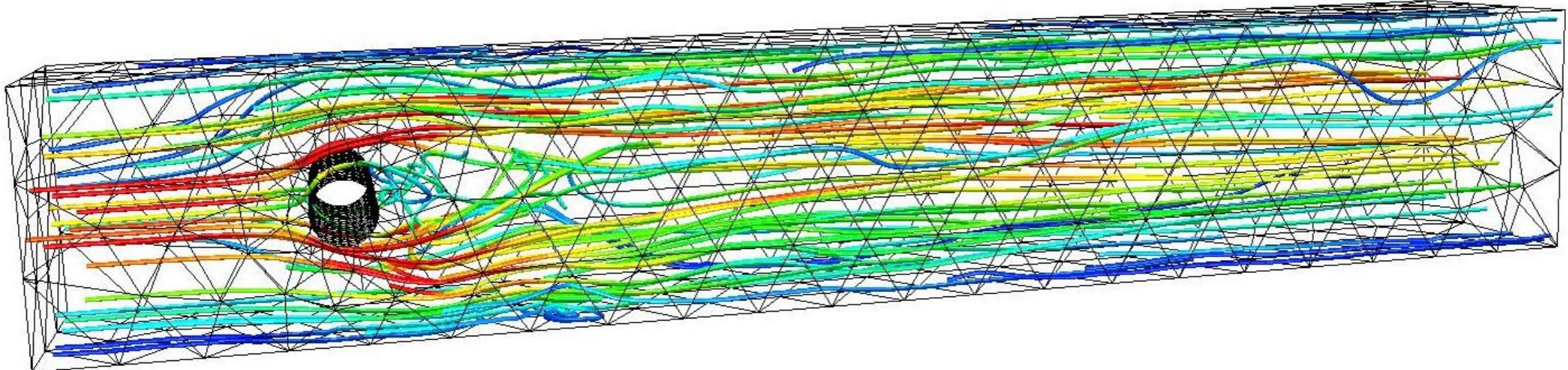
$Re = 1000$:



$Re = 5000$:



Flow around a cylinder, $Re = 100$



Schäfer-Turek benchmark (3D, instationary)

Featflow (Turek et al, 2011): $Q_2 - P_1$ -FEM:

dofs	max c_D	max c_L	min c_L	T (sec x cores)
199 K	3.2207	0.0027	-0.0095	3 220 × 4
1482 K	3.2877	0.0028	-0.010892	17 300 × 8
11432 K	3.2963	0.0028	-0.010992	35 550 × 48
89760 K	3.2978	0.0028	-0.010999	214 473 × 96

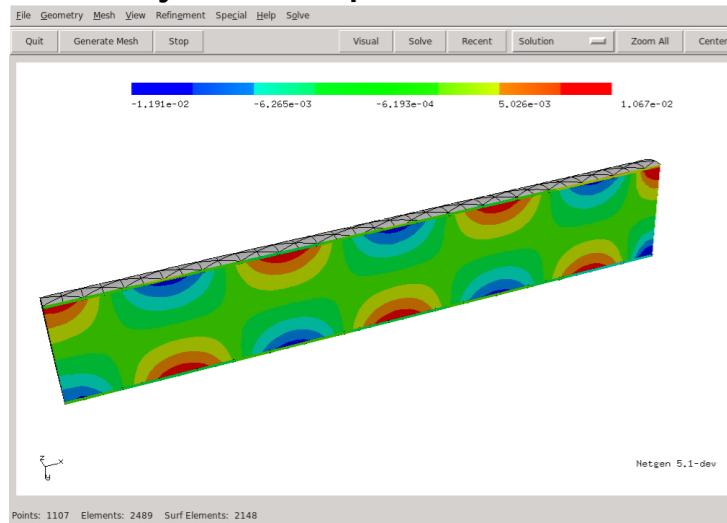
NGSolve, HDG:

p	dofs	max c_D	max c_L	min c_L	T (sec x cores)
3	119 K	3.39935	0.003614	-0.02227	388 × 24
4	206 K	3.31082	0.003067	-0.01320	970 × 24
5	326 K	3.30082	0.002966	-0.01044	2385 × 24
6	480 K	3.29769	0.002964	-0.010826	4374 × 24

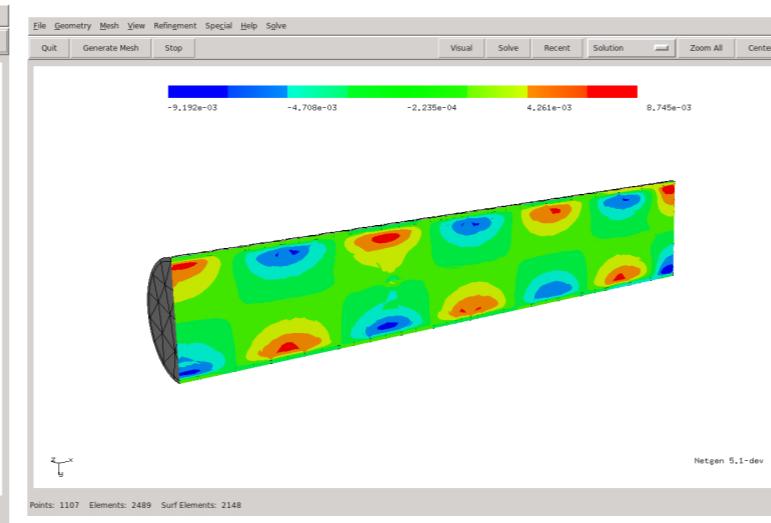
Coriolis mass flow meter (S. Stingelin, Endress-Hauser)

Monolithic fluid structure interaction

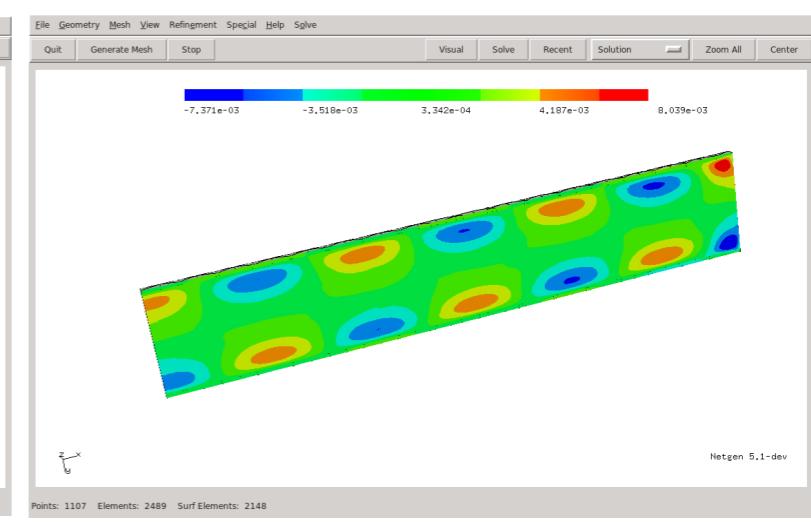
velocity x -component in fluid for different viscosities:



$$\nu = 0.1$$



$$\nu = 10$$



$$\nu = 100$$

$$(A + i\omega\nu B - \omega^2 M)u = f, \quad +\text{div-free constraint}$$

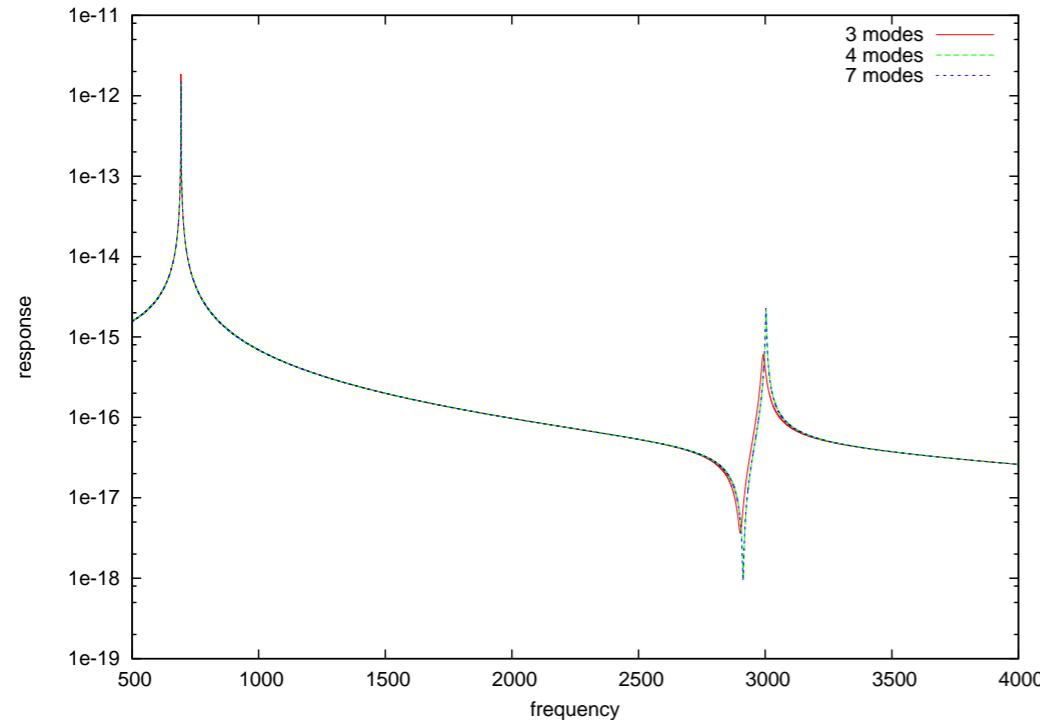
A .. elasticity op, B .. viscosity op, M .. mass op, and small viscosity ν

Simulation with hybrid DG in fluid and solid. $p = 2\ldots 4$, $N = 200k\ldots 500k$, Time = 20 sec ... 2 min

Reduced Basis Method (RBM)

$$(A + i\omega\nu B - \omega^2 M)u = f$$

Compute snapshots $u^j = u(\omega_j)$ for $j = 1 \dots m$, and project onto $V_m = \text{span}\{u^j\}$



Eigenvalues from RBM:

modes	EV 1	EV 2
3	693 - i 0.136	2992 - i 6.62
4	693 - i 0.170	3003 - i 1.85
7	693 - i 0.166	3002 - i 1.73

with Anna Zechner

Concluding remarks

- there are many fancy methods for scalar fields
- and even more games for vector-fields (and tensor-fields !)
- look for qualitative properties of compatible discretization methods
- high order methods are not always faster - but moving to hp-FEM is a one-way road
- implementation becomes easy with a powerful and extendable finite element library - have a look into open-source code NGSolve
- fast and robust solvers are the key for real applications