Construction of High Order Finite-Element Spaces

a proof of concept...

ANDREAS DEDNER

Mathematics Institute, University of Warwick, Coventry UK
A.S.Dedner@warwick.ac.uk,
www2.warwick.ac.uk/fac/sci/maths/people/staff/andreas_dedner

Co-worker:
Martin Nolte

Chemnitz, 27th of September 2011

Overview

A finite element discretization requires

1. construct partitioning $\mathcal{G}$ of Ω
2. construct basis $\mathcal{B}$ of $V_{\mathcal{G}}$
3. implement of quadratic forms $a(\cdot, \cdot)$
4. solve linear system (find root, evolve in time)

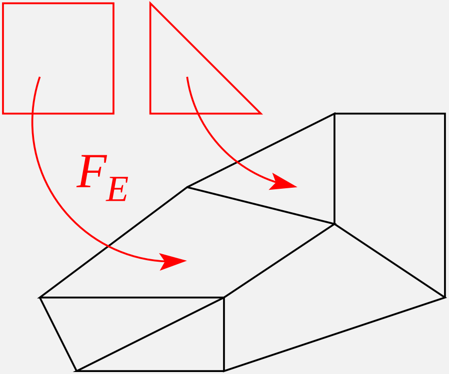
Need to:

1. evaluate basis function and derivatives
2. efficient evaluation in given set of points (quadrature)
3. associate basis functions with subentities (mapper)
4. interpolation or projection into discrete function space

Overview

Given: grid $\mathcal{G}$ with entities E and a small set of reference elements $\mathcal{R}$

Assumption: for each entity E there is a reference element $\hat{E} \in \mathcal{R}$ from a small set $\mathcal{R}$ and a bijective smooth mapping $F_E : \hat{E} \to E$



Values of basis functions are needed on given set of points in R
(can be computed at start up).

Overview

Finite element construction is based on families of finite elements

$$(R, V_R, \mathbf{\Lambda}_R)$$

where

- R is from a small set of reference elements $\mathcal{R}$
- V_R is a finite dimensional function space
- $\mathbf{\Lambda}_R = (\lambda_i)$ is a basis of the dual to V_R

The construction now requires

1. building the primal basis, i.e., a basis $\mathcal{B}_R = (\phi_i)_i$ satisfying

$$\lambda_j(\phi_i) = \delta_{ij} \; .$$

2. by associating each functional $\lambda_j \in \mathbf{\Lambda}_R$ with a subentity (edge, face,...) of R a global space is build.

Goal

Use this (abstract) idea to automatically construct finite-element spaces of arbitrary order on arbitrary reference elements, *without coding polynomials*.

Why bother?

1. Basis function are often quite tedious to code, e.g., 5th order Raviart-Thomas on prisms.
2. Want to compare different choices of functionals, i.e., same sapce V_R different dual basis.
3. We often use matrix free methods (in implicit ODE solvers for example) so the conditioning of the mass matrix is important (no preconditioning).
4. Want construction also in higher dimension.
5. We found it interesting...

Shape function sets

Construction based on dual basis or nodal variables $\mathbf{\Lambda}_R = (\lambda_j)_j$ i.e. the shape functions $\mathcal{B}_R = (\phi_i)_i$ satisfy

$$\lambda_j(\phi_i) = \delta_{ij} \; .$$

Idea for Construction:

Given

1. any basis $\boldsymbol{B}_R = (\psi_1, \ldots, \psi_{N_R})$ of the space V_R
2. set of nodal variables $\mathbf{\Lambda}_R = (\lambda_1, \ldots, \lambda_{N_R})$

Then the basis $\mathcal{B}_R = \{\phi_1, \ldots, \phi_{N_R}\}$ with $\lambda_j(\phi_i) = \delta_{ij}$ is given by

$$\mathcal{B}_R = (A_R)^{-T} \boldsymbol{B}_R, \quad \text{with} \quad A_R = \big(\lambda_j(\psi_i)\big)_{ij} \in \mathbb{R}^{N_R \times N_R}$$

Shape function sets

Similar: Orthonormal basis functions:

Given

1. any basis $\boldsymbol{B}_R = (\psi_1, \dots, \psi_{N_R})$ of the space V_R
2. scalar product $a_R(\cdot, \cdot)$

Seek orthonormal shape functions $\mathcal{B}_R = (\phi_i)_i$ w.r.t. a_R, given by

$$\mathcal{B}_R = (A_R)^{-T}\boldsymbol{B}_R, \quad \text{with} \quad A_R A_R^T = M_R := \big(a_R(\psi_j, \psi_i)\big)_{ij} .$$

Functionals are now given by $\lambda_i(u) = a(u, \varphi_i)$.

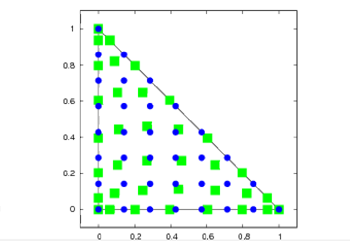
Shape function sets

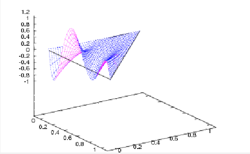
$$\mathcal{B}_R = (A_R)^{-T}\boldsymbol{B}_R, \quad \text{with} \quad A_R = \big(\lambda_j(\psi_i)\big)_{ij} \in \mathbb{R}^{N_R\times N_R}$$

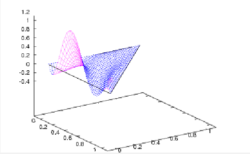
1. Description of set of reference elements $\mathcal{R}$.
2. Description of pre-basis $\boldsymbol{B}_R = (\psi_1, \ldots, \psi_{N_R})$
3. Description of nodal variables $\boldsymbol{\Lambda}_R = (\lambda_1, \ldots, \lambda_{N_R})$

Finally: stable construction of $(A_R)^{-T}$.

Example: Lagrange spaces based on Lagrange interpolation $\lambda_i^p(u) = u(\mathbf{x}_i^p)$ with point set $(\mathbf{x}_i^p)$, $\boldsymbol{B}_R$ is set of monomials (or bimonomials).







Equidistant and Lobatto type point set (Luo and Pozrikidis, 2006).

Reference elements

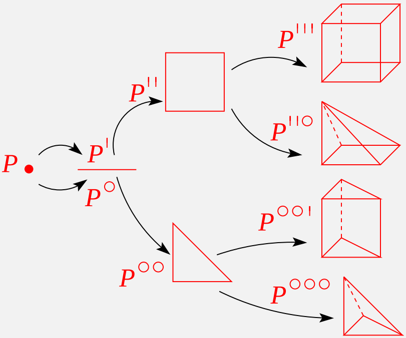
Generic Reference Elements

Given set of reference elements $\mathcal{R}^d$ with $R \subset \mathbb{R}^d, R \in \mathcal{R}^d$ we define

$$\mathcal{R}^{d+1} = \{R^|, R^\circ : R \in \mathcal{R}^{d+1}\}$$

where for $R \in \mathcal{R}^d$:

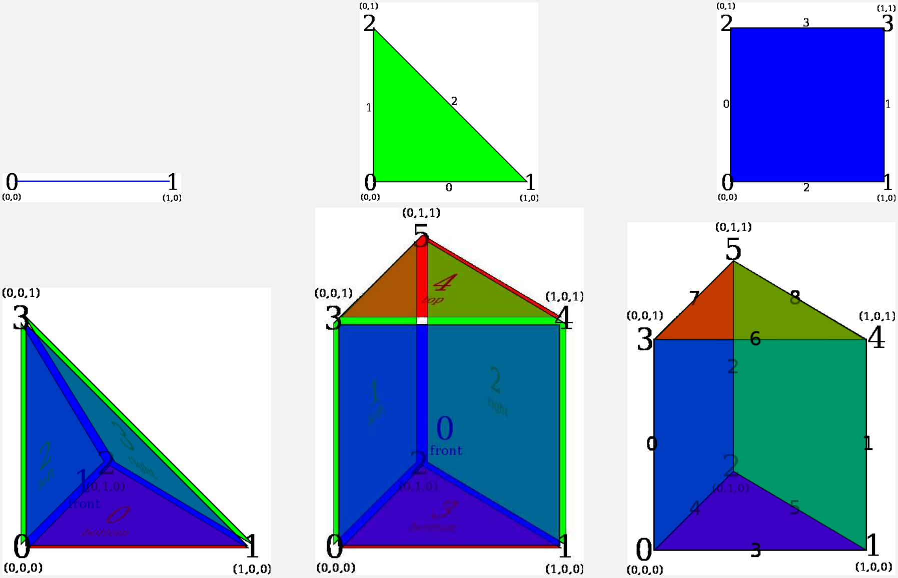
$$\begin{aligned} R^| &= \{(x,z) : z \in [0,1], x \in R\} \\ R^\circ &= \{(x(1-z), z) : z \in [0,1], x \in R\} \end{aligned}$$



For $d = 0$ we set
$\mathcal{R}^0 = \{P\}$ with $P = \{0\} \in \mathbb{R}^0$.
Note: R° is the Duffy transform of R.

Recursion also provides numbering of subentities $\hat{e}$ and corresponding mappings from $\hat{e} \to R$.

Examples



All embeddings of higher codimension subentities respect ordering.

Pre Basis

Monomial Basis Function

Ansatz: we assume each base function is a polynomial in d variables.

Example on simplex topology $S^0 = P$, $S^{d+1} = (S^d)^\circ$

We construct Ψ_k^d of all monomials in d variables of exactly order k:

dim.	monomials	recursion relation		
0	$\Psi_0^0 = 1$	$\Psi_0^0 = 1$		
1	$\Psi_0^1 = 1$	$\Psi_0^0 = 1$		
	$\Psi_1^1 = x$	$\Psi_1^0 = \emptyset$	$x\Psi_0^0 = x$	
	$\Psi_2^1 = x^2$	$\Psi_2^0 = \emptyset$	$x\Psi_1^0 = \emptyset$	$x(x\Psi_0^0) = x^2$
2	$\Psi_0^2 = 1$	$\Psi_0^1 = 1$		
	$\Psi_1^2 = \{x, y\}$	$\Psi_1^1 = x$	$= y$	
	$\Psi_2^2 = \{x^2, xy, y^2\}$	$\Psi_2^1 = x^2$	$y\Psi_1^1 = xy$	$y \quad = y^2$
d+1	$\Psi_0^{d+1} = \{\dots\}$	Ψ_0^d		
	$\Psi_1^{d+1} = \{\dots\}$	Ψ_1^d	$z\Psi_0^d$	
	$\Psi_2^{d+1} = \{\dots\}$	Ψ_2^d	$z\Psi_1^d$	$z(z\Psi_0^d)$

Monomial Basis Function

Ansatz: we assume each base function is a polynomial in d variables.

Example on simplex topology $S^0 = P$, $S^{d+1} = (S^d)^\circ$

We construct Ψ_k^d of all monomials in d variables of exactly order k:

dim.	monomials	recursion relation		
0	$\Psi_0^0 = 1$	$\Psi_0^0 = 1$		
1	$\Psi_0^1 = 1$	$\Psi_0^0 = 1$		
	$\Psi_1^1 = x$	$\Psi_1^0 = \emptyset$	$x\Psi_0^0 = x$	
	$\Psi_2^1 = x^2$	$\Psi_2^0 = \emptyset$	$x\Psi_1^0 = \emptyset$	$x(x\Psi_0^0) = x^2$
2	$\Psi_0^2 = 1$	$\Psi_0^1 = 1$		
	$\Psi_1^2 = \{x, y\}$	$\Psi_1^1 = x$	$y\Psi_0^1 = y$	
	$\Psi_2^2 = \{x^2, xy, y^2\}$	$\Psi_2^1 = x^2$	$y\Psi_1^1 = xy$	$y(y\Psi_0^1) = y^2$
d+1	$\Psi_0^{d+1} = \{\dots\}$	Ψ_0^d		
	$\Psi_1^{d+1} = \{\dots\}$	Ψ_1^d	$z\Psi_0^d$	
	$\Psi_2^{d+1} = \{\dots\}$	Ψ_2^d	$z\Psi_1^d$	$z(z\Psi_0^d)$

Monomial Basis Function

Ansatz: we assume each base function is a polynomial in d variables.

Example on simplex topology $S^0 = P$**,** $S^{d+1} = (S^d)^\circ$

We construct Ψ_k^d of all monomials in d variables of exactly order k:

dim.	monomials	recursion relation
0	$\Psi_0^0 = 1$	$\Psi_0^0 = 1$
1	$\Psi_0^1 = 1$	$\Psi_0^0 = 1$
	$\Psi_1^1 = x$	$\Psi_1^0 = \emptyset \quad x\Psi_0^0 = x$
	$\Psi_2^1 = x^2$	$\Psi_2^0 = \emptyset \quad x\Psi_1^0 = \emptyset \quad x(x\Psi_0^0) = x^2$
2		$\Psi_0^1 = 1$
	$\Psi_0^2 = 1$	
	$\Psi_1^2 = \{x, y\}$	$\Psi_1^1 = x \quad y\Psi_0^1 = y$
	$\Psi_2^2 = \{x^2, xy, y^2\}$	$\Psi_2^1 = x^2 \quad y\Psi_1^1 = xy \quad y(y\Psi_0^1) = y^2$
d+1	$\Psi_0^{d+1} = \{\dots\}$	Ψ_0^d
	$\Psi_1^{d+1} = \{\dots\}$	$\Psi_1^d \quad z\Psi_0^d$
	$\Psi_2^{d+1} = \{\dots\}$	$\Psi_2^d \quad z\Psi_1^d \quad z(z\Psi_0^d)$

Generic Monomials

Example on cube topology $Q^0 = P$, $Q^{d+1} = (Q^d)^{|}$

We construct Ψ_k^d of all bi-monomials in d variables of exactly order k:

dim.	bi-monomials	recursion relation
0	$\Psi_0^0 = 1$	$\Psi_0^0 = 1$
1	$\Psi_0^1 = 1$	$\Psi_0^0 = 1$
	$\Psi_1^1 = x$	$\Psi_1^0 = \emptyset \quad x\Psi_1^0 = \emptyset \quad x\Psi_0^0 = x$
2	$\Psi_0^2 = 1$	$\Psi_0^1 = 1$
	$\Psi_1^2 = \{x, xy, y\}$	$\Psi_1^1 = x \quad y\Psi_1^1 = yx \quad y\Psi_0^1 = y$
d+1	$\Psi_0^{d+1} = \{\dots\}$	Ψ_0^d
	$\Psi_1^{d+1} = \{\dots\}$	$\Psi_1^d \quad z\Psi_1^d \quad z\Psi_0^d$
	$\Psi_2^{d+1} = \{\dots\}$	$\Psi_2^d \quad z\Psi_2^d \quad z(z\Psi_2^d) \quad z(z\Psi_1^d) \quad z(z\Psi_0^d)$

Recursion correct for any reference element $R^{|}$ (e.g., prisms).

Note: recursion also works to compute arbitrary derivatives.

Nodal variables

Example Spaces

Lagrange space

Pre basis: $\boldsymbol{B}_R = \mathcal{M}_R^k$

Functionals: $\lambda_p(u) = u(p) \; p \in P_L$

Orthonormal shape functions

Pre basis: $\boldsymbol{B}_R = \mathcal{M}_{S^d}^k$ or $\mathcal{M}_R^k$

Bilinear form: $a_R(u,v) = \int_R uv$

Raviart-Thomas space

Pre basis: $\boldsymbol{B}_R = (\mathcal{M}_R^k)^d + x\bar{\mathcal{M}}_R^k$

Functionals:
$$\begin{aligned} \lambda_{\hat{e},p}(u) &:= \int_{\hat{e}} u \cdot n_{\hat{e}}\, p \quad \text{for all } \hat{e} \in \hat{E}_R^1 \text{ and } p \in B_k(\hat{e}) \\ \lambda_{R,p,j}(u) &:= \int_R u \cdot e_j\, p \quad \text{for all } p \in B_{k-1}(R) \text{ and } j = 1, \ldots, d \end{aligned}$$

Raviart-Thomas space

Pre basis: $\boldsymbol{B}_R = (\mathcal{M}_R^k)^d + x\bar{\mathcal{M}}_R^k$

Functionals:
$$\begin{aligned} \lambda_p(u) &:= u(p) \cdot n_{\hat{e}(p)} \quad \text{for all } p \in P_L^1 \\ \lambda_{p,j}(u) &:= u(p) \cdot e_j \quad \text{for all } p \in P_L^0 \text{ and } j = 1, \ldots, d \end{aligned}$$

Example Spaces

Raviart-Thomas space

Pre basis:
$$\boldsymbol{B}_R = (\mathcal{M}_R^k)^d + x\bar{\mathcal{M}}_R^k$$

Functionals:
$$\begin{aligned}\lambda_{\hat{e},p}(u) &:= \int_{\hat{e}} u \cdot n_{\hat{e}}\, p \quad \text{for all } \hat{e} \in \hat{E}_R^1 \text{ and } p \in B_k(\hat{e}) \\ \lambda_{R,p,j}(u) &:= \int_R u \cdot e_j\, p \quad \text{for all } p \in B_{k-1}(R) \text{ and } j = 1, \dots, d\end{aligned}$$

Raviart-Thomas space

Pre basis:
$$\boldsymbol{B}_R = (\mathcal{M}_R^k)^d + x\bar{\mathcal{M}}_R^k$$

Functionals:
$$\begin{aligned}\lambda_p(u) &:= u(p) \cdot n_{\hat{e}(p)} \quad \text{for all } p \in P_L^1 \\ \lambda_{p,j}(u) &:= u(p) \cdot e_j \quad \text{for all } p \in P_L^0 \text{ and } j = 1, \dots, d\end{aligned}$$

Definition of functionals require

- Definition of pointsets (on subentities)
- Quadrature rules (on subentities)

Everything can be implmented using recursive definition of reference elements and subentity embeddings.

Construction of shape function set

Usage of high precision field type

1. Construction phase

Evaluate *prebasis* $\boldsymbol{B}$ for $A_R = \big(\lambda_j(\psi_i)\big)_{ij} \in \mathbb{R}^{N_R \times N_R}$ and compute A_R^{-T}.
Note: Setting up the *sparse* matrix is only done once.

2. Evaluation phase

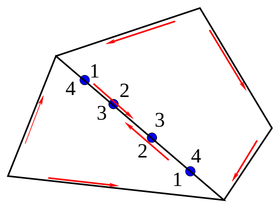
Evaluation of *prebasis* $\boldsymbol{B}$ to compute $\mathcal{B}_R = (A_R)^{-T}\boldsymbol{B}_R$.
Note: For any basis this is always the same matrix-vector multiplication (even for derivatives).
Note: If caching on quadrature points is used then this step is also start up.

Usage of different field types:

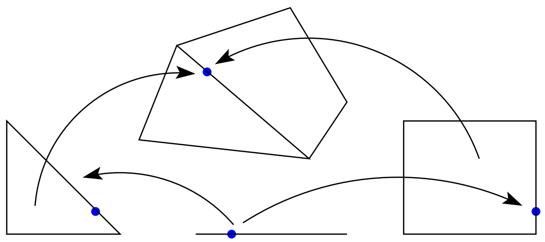
We use *high precision floating point* arithmetics (alglib based on mpfr, gmp).
ComputeField: used to setup matrix and during inversion/QR.
StorageField: used for storing the matrix B^{-T} and during the evaluation phase.
Note: Final Caching is done in standard double precision.

Construction of the global space

Problem: twist in grid



Idea of twist free



Definition (Twist-free grids)

Grids are twist free when this does not happen (an even more technical definition is possible...)

1. Cartesian grids in any dimension d are twist free
2. Simplex grids in any dimension d can be made twist free due to the construction of the reference elements: use global numbering of verticies to sort verticies $p_{n_0}, \ldots, p_{n_d}$ in simplex T, i.e., $n_i < n_{i+1}$. Now construct F_T with $F_T(\hat{p}_k) = p_{n_k}$.

Construction of the global space

Note: in all reference elements edges are oriented from low to high index, similar for faces...
General case: Sort verticies of T as before: $p_{n_0}, \ldots, p_{n_N}$ with $n_i < n_{i+1}$.
Construct mapping τ from reference element $\hat{T}$ to itself taking $\hat{p}_k \to \hat{p}_{n_k}$ (simple).
Use τ in fuctionals during basis construction, e.g,

$$\text{Lagrange } : \lambda_p(u) = u(\tau(p)) \; p \in P_L$$

$$\text{RT } : \lambda_{\hat{e},p}(u) = \int_{\hat{e}} u \circ \tau \cdot n_{\hat{e}} \, p$$

Result: new set of basis function for each occuring ”twist” in grid (small number, can use caching).

Comparing different shape functions

Raviart-Thomas space

Functionals:

$$\begin{aligned}
\lambda_{\hat{e},p}(u) &:= \int_{\hat{e}} u \cdot n_{\hat{e}}\, p \quad \text{for all } \hat{e} \in \hat{E}_R^1 \text{ and } p \in B_k(\hat{e}) \\
\lambda_{R,p,j}(u) &:= \int_R u \cdot e_j\, p \quad \text{for all } p \in B_{k-1}(R) \text{ and } j = 1, \ldots, d
\end{aligned}$$

Raviart-Thomas space

Functionals:

$$\begin{aligned}
\lambda_p(u) &:= u(p) \cdot n_{\hat{e}(p)} \quad \text{for all } p \in P_L^1 \\
\lambda_{p,j}(u) &:= u(p) \cdot e_j \quad \text{for all } p \in P_L^0 \text{ and } j = 1, \ldots, d
\end{aligned}$$

L^2-ONB:	orthonormal basis for local L^2-projection.
L^2-Lob:	Lagrange functions using Lobatto points for local L^2-projection.
P-Lob:	Lobatto point set for the pointwise evaluation.

We also tested the equidistant point set but results are less satisfactory.

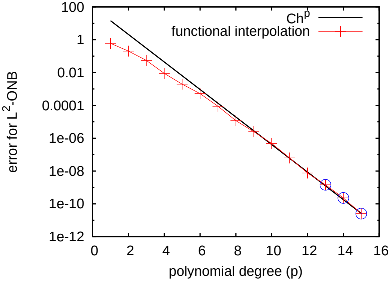
Comparing different shape functions

Functional interpolation: comput DoFs using functionals λ.

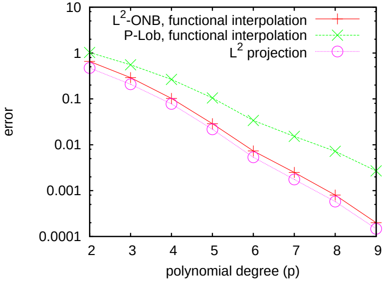
L^2 **projection:** invert mass matrix to cmpute DoFs

Note: the same discrete function but conditioning of mass matrix differes.

Interpolation error (p convergence)



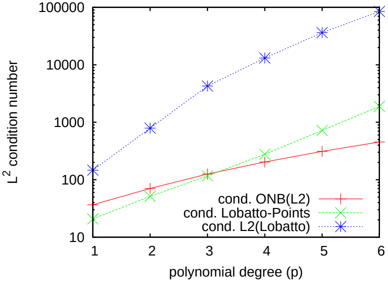
123 triangles
last three using high precision



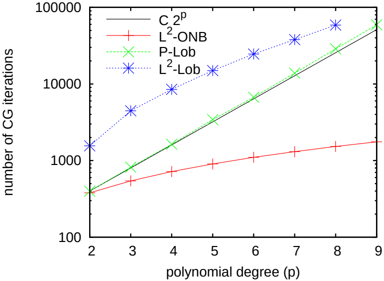
233 tetrahedra
including L^2 projection

Comparing different shape functions

Conditioning of mass matrix (p refinement)



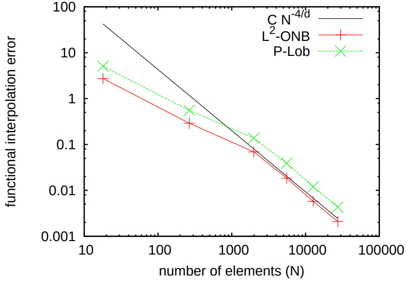
Condition number
(28 triangles)



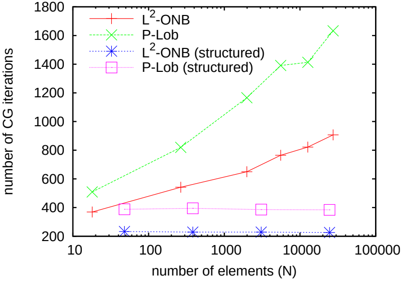
unpreconditioned CG steps
(233 tetrahedra)

Comparing different shape functions

Conditioning of mass matrix (*h* refinement)



interpolation error



number of CG iterations

degree $k = 3$ in 3d $d = 3$

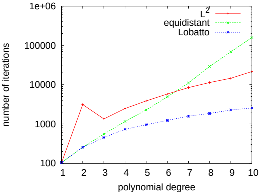
An experiment

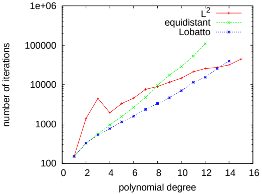
Define Lagrange space through L^2 projection:

L^2 based Lagrange space, (e.g. on triangles)

$$\begin{aligned}
\textit{Functionals:}\quad \lambda_{\hat{e},p}(u) &:= \int_{\hat{e}} u\,p = u(\hat{e}) &&\text{for all } \hat{e} \in \hat{E}^2_R \text{ and } p \in B_0(\hat{e}) \\
\lambda_{\hat{e},p}(u) &:= \int_{\hat{e}} u\,p &&\text{for all } \hat{e} \in \hat{E}^1_R \text{ and } p \in B_{k-2}(\hat{e}) \\
\lambda_{R,p}(u) &:= \int_{R} u\,p &&\text{for all } p \in B_{k-3}(R)
\end{aligned}$$

Test case: $-5\triangle u + u = f$ (2d cube, 2d simplex)





I am still hoping we can find some hidden great property...

To Do

- test more elements: Nedelec, hierarchic...
- use code generator: e.g., let Maple generates optimized code.

  Example:

  tex output Raviart-Thomas on 2D Simplex order 3 (note vector valued)

$\varphi_0(a,b) =$
$(a - 0.1800000000000000000000000000000000000000E2ab + 0.6300000000000000000000000000000000000000E2ab^2 -$
$0.5600000000000000000000000000000000000000E2ab^3, -0.1000000000000000000000000000000000000000E1 +$
$0.1900000000000000000000000000000000000000E2b - 0.8100000000000000000000000000000000000000E2b^2 +$
$0.1190000000000000000000000000000000000000E3b^3 - 0.5600000000000000000000000000000000000000E2b^4)$
$\varphi_1(a,b) =$
$(-0.2424871130596428210938424878108221313725E1a + 0.4156921938165305504465871219614093680664E1a^2 +$
$0.3533383647440509678795990536671979628557E2ab - 0.5819690713431427706252219707459731152920E2a^2b -$
$0.9456997409326070022659857024622063123519E2ab^2 + 0.1163938142686285541250443941491946230584E3a^2b^2 +$
$0.5819690713431427706252219707459731152920E2ab^3, 0.1732050807568877293527446341505872366945E1 -$
$0.3464101615137754587054892683011744733891E1a - 0.2875204340564336307255560926899748129126E2b +$
$0.5403998519614897155805632585498321784868E2ab + 0.1070407399077566167399961839050629122771E3b^2 -$
$0.1600414946193642619219360419551426067056E3ab^2 - 0.1382176544439964080234902180521686148824E3b^3 +$
$0.1163938142686285541250443941491946230584E3ab^3 + 0.5819690713431427706252219707459731152920E2b^4)$
$\varphi_2(a,b) =$
$(0.4919349550499537332100182071208807717973E1a - 0.2146625258399798108552806721982025186026E2a^2 -$
$0.4561578674099570980674714284211803520299E2ab + 0.1878297101099823344983705881734272037771E2a^3 +$
$0.1878297101099823344983705881734272037769E3a^2b + 0.6574039853849381707442970586069952132183E2ab^2 -$
$0.1502637680879858675986964705387417630217E3a^3b - 0.1502637680879858675986964705387417630217E3a^2b^2 -$
$0.2504396134799764459978274508979029383691E2ab^3, -0.2236067977499789696409173668731276235440E1 +$
$0.1341640786499873817845504201238765741264E2a + 0.2638560213449751841762824929102905957824E2b -$
$0.1341640786499873817845504201238765741264E2a^2 - 0.1448972049419863723273144537337867000567E3ab -$
$0.7110696168449331234581172266565458428714E2b^2 + 0.1314807970769876341488594117213990426437E3a^2b +$
$0.2817445651649735017475558822601408056661E3ab^2 + 0.7200138887549322822437539213314709478104E2b^3 -$
$0.1502637680879858675986964705387417630217E3a^2b^2 - 0.1502637680879858675986964705387417630217E3ab^3 -$