WARWICK
THE UNIVERSITY OF WARWICK

A General Approach to Implementing Virtual Element Methods

Andreas Dedner
Mathematics Institute, University of Warwick

with Alice Hodson (now University of Prague)

September 19, 2023

Overview of dune-vem

Virtual elements

- General construction of spaces for, e.g., elliptic, forth order problems, $H(\mathrm{div})$, $H(\mathrm{curl})$, divergence free etc
- No restriction on the element shapes

Currently available in dune-vem

- wide range of spaces for second order problems
- wide range of spaces for forth order problems
- bounding box discontinuous Galerkin spaces
- curl-free space
- divergence compatible spaces (piecewise constant divergence)
- $H(div)$ and $H(curl)$ conforming spaces (in progress)

arbitrary order for general elements in 2D.

Starting point

Given set of local dofs Λ_E on element E

VEM space: $(E, ?, \Lambda_E)$ instead of FEM space: (E, V_E, Λ_E)

Problem with FEM:

finding a suitable basis M_E of the space V_E (pre-basis in dune-localfunctions implementation).

Problem with VEM:

we don't know too much about V_E - it's virtual...

[1] Dedner and Hodson, *Implementing general virtual element spaces*.

Starting point

Given set of local dofs Λ_E on element E

VEM space: $(E, ?, \Lambda_E)$ instead of FEM space: (E, V_E, Λ_E)

Problem with FEM:

finding a suitable basis M_E of the space V_E (pre-basis in dune-localfunctions implementation).

Problem with VEM:

we don't know too much about V_E - it's virtual...

VEM idea:

define projections into a simple polynomial space(s) P ($P^\star, P^\sharp$):

$$v_E \approx \mathcal{G}^{E,0} v_E \in P\,, \qquad \nabla v_E \approx \mathcal{G}^{E,\nabla} v_E \in P^\star\,, \qquad \nabla^2 v_E \approx \mathcal{G}^{E,H} v_E \in P^\sharp\,,$$

Our approach:[1]

$\mathcal{G}^{E,0} v_E$ depends on Λ_E (constraint least squares problem).

Other projections are generic but in general

$$\mathcal{G}^{E,\nabla} v_E \neq \nabla \mathcal{G}^{E,0} v_E$$

Advantage: no special projection for e.g. $\mathrm{div}\, v_E$, use $\mathrm{trace}\,\mathcal{G}^{E,\nabla} v_E$.

[1] Dedner and Hodson, *Implementing general virtual element spaces*.

General set of dofs for 2nd/4th order PDE

Local VEM space defined by moments tuple

$$\Big(\underbrace{(\delta_{p0} \quad\quad)}_{\text{vertex dofs}} , \underbrace{(\delta_{e0} \quad\quad)}_{\text{edge dofs}} , \underbrace{\delta_i}_{\text{inner dofs}} \Big)$$

with $\delta_{p0}, \quad \in \{-1, 0\}$ and $\delta_{e0}, \quad \delta_i \leq l$ encoding set of dofs

For 2nd order PDEs:

$$L^i := \{\lambda^E_{m_\beta} : v^E \mapsto \frac{1}{|E|} \int_E v^E m_\beta\} , \qquad (m_\beta)_\beta \text{ basis of } \mathbb{P}_{\delta_i}(E)$$

$$L^{e,0} := \{\lambda^{e,0}_{\psi_\beta} : v^E \mapsto \frac{1}{|e|} \int_e v^E \psi_\beta\} , \qquad e \in E^1 , \qquad (\psi_\beta)_\beta \text{ basis of } \mathbb{P}_{\delta_{e0}}(e)$$

$$L^{p,0} := \{\lambda^{p,0} : v^E \mapsto v^E(p)\} , \qquad p \in E^0 , \qquad \text{if } \delta_{p0} = 0$$

General set of dofs for 2nd/4th order PDE

Local VEM space defined by moments tuple

$$\Big(\underbrace{(\delta_{p0},\delta_{p1})}_{\text{vertex dofs}},\ \underbrace{(\delta_{e0},\delta_{e1})}_{\text{edge dofs}},\ \underbrace{\delta_i}_{\text{inner dofs}}\Big)$$

with $\delta_{p0}, \delta_{p1} \in \{-1,0\}$ and $\delta_{e0}, \delta_{e1}, \delta_i \leq l$ encoding set of dofs

For 2nd order PDEs:

$$L^i := \{\lambda^E_{m_\beta} : v^E \mapsto \frac{1}{|E|}\int_E v^E m_\beta\}\ , \qquad (m_\beta)_\beta \text{ basis of } \mathbb{P}_{\delta_i}(E)$$

$$L^{e,0} := \{\lambda^{e,0}_{\psi_\beta} : v^E \mapsto \frac{1}{|e|}\int_e v^E \psi_\beta\}\ , \qquad e \in E^1\ , \qquad (\psi_\beta)_\beta \text{ basis of } \mathbb{P}_{\delta_{e0}}(e)$$

$$L^{p,0} := \{\lambda^{p,0} : v^E \mapsto v^E(p)\}\ , \qquad p \in E^0\ , \qquad \text{if } \delta_{p0} = 0$$

additonal for 4th order PDEs: add δ_{p1}, δ_{e1}

$$L^{e,1} := \{\lambda^{e,1}_{\psi_\beta} : v^E \mapsto \int_e \nabla v^E \cdot n_e \psi_\beta\} \quad e \in E^1\ , \quad (\psi_\beta)_\beta \text{ basis of } \mathbb{P}_{\delta_{e1}}(e)$$

$$L^{p,1} := \{\lambda^{p,1} : v^E \mapsto \nabla v^E(p)\} \qquad p \in E^0\ , \quad \text{if } \delta_{p1} = 0$$

General set of dofs: examples with $l = 4$

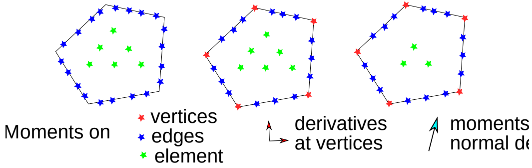
Dof tuple:

$$\Big(\underbrace{(\delta_{p0} \quad)}_{\text{vertex dofs}} , \underbrace{(\delta_{e0} \quad)}_{\text{edge dofs}} , \underbrace{\delta_i}_{\text{inner dofs}} \Big)$$

$\delta_{p0}, \delta_{p1} \in \{-1, 0\}, \delta_{e0}, \delta_{e1}, \delta_i \leq l$

```python
# non-conforming C^1 space
from dune.vem import vemSpace
spc = vemSpace(grid, order=4,
           testSpaces=[[0,-1],[2,2],0])
```

non conforming H^1	conforming H^1	Serendipity conforming H^1
$((-1,-1),(3,-1),2)$	$((0,-1),(2,-1),2)$	$((0,-1),(2,-1),1)$



General set of dofs: examples with $l = 4$

Dof tuple:

$$\Big(\underbrace{(\delta_{p0}, \delta_{p1})}_{\text{vertex dofs}}, \underbrace{(\delta_{e0}, \delta_{e1})}_{\text{edge dofs}}, \underbrace{\delta_i}_{\text{inner dofs}} \Big)$$

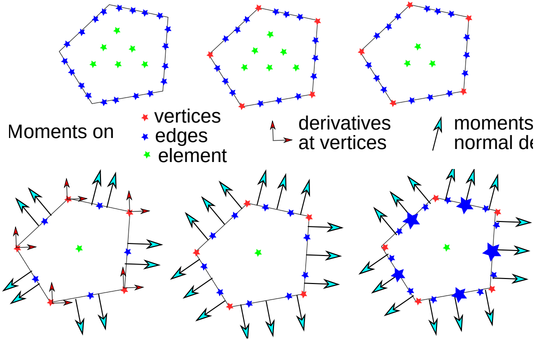
$\delta_{p0}, \delta_{p1} \in \{-1, 0\}$, $\delta_{e0}, \delta_{e1}, \delta_i \leq l$

```
# non-conforming C^1 space
from dune.vem import vemSpace
spc = vemSpace(grid, order=4,
        testSpaces=[[0,-1],[2,2],0])
```

non conforming H^1 $((-1,-1),(3,-1),2)$

conforming H^1 $((0,-1),(2,-1),2)$

Serendipity conforming H^1 $((0,-1),(2,-1),1)$



conforming H^2 $((0,0),(0,1),0)$

non conforming H^2 $((0,-1),(1,2),0)$

C^0-conf, H^2-nc (4^{th} order perturbation) $((0,-1),(2,2),0)$

Implementation details[1]

1. Grid View T_h: consisting of elements E
2. Local Mapper μ_E: local dofs to global dofs
3. Local Nodal Basis B_E: evaluate values, jacobians, hessians, ...
4. Local Operators L_E: assemble local contributions

[1] Dedner and Hodson, *Implementing general virtual element spaces*.

Implementation details[1]

1. Grid View T_h: consisting of elements E
   - Any (dune) grid
   - Agglomerated triangular grid: a triangle T knows polygon $E \supset T$.
   - Alternative: use dune-polygongrid (not yet added)

[1] Dedner and Hodson, *Implementing general virtual element spaces*.

Implementation details[1]

1. Grid View T_h: consisting of elements E
   - Any (dune) grid
   - Agglomerated triangular grid: a triangle T knows polygon $E \supset T$.
   - Alternative: use dune-polygongrid (not yet added)
2. Local Mapper μ_E: local dofs to global dofs
   - Based on local key approach (easy e.g. for dof tuple)
   - In agglomerated approach $\mu_T = \mu_E$ for all $T \subset E$.

[1] Dedner and Hodson, *Implementing general virtual element spaces*.

Implementation details[1]

1. Grid View T_h: consisting of elements E
   - Any (dune) grid
   - Agglomerated triangular grid: a triangle T knows polygon $E \supset T$.
   - Alternative: use dune-polygongrid (not yet added)
2. Local Mapper μ_E: local dofs to global dofs
   - Based on local key approach (easy e.g. for dof tuple)
   - In agglomerated approach $\mu_T = \mu_E$ for all $T \subset E$.
3. Local Nodal Basis B_E: evaluate values, jacobians, hessians, ...
   In FE can start with any basis M_E of local space V_E.
   Use dofs to construct basis transform matrix A_E so that

$$\texttt{eval}(\texttt{E}, \texttt{x}) = A_E M_E(x) , \quad \texttt{jac}(\texttt{E}, \texttt{x}) = A_E \nabla M_E(x) .$$

   This does not work for VEM.

[1] Dedner and Hodson, *Implementing general virtual element spaces*.

Implementation details[1]

1. Grid View T_h: consisting of elements E
   - Any (dune) grid
   - Agglomerated triangular grid: a triangle T knows polygon $E \supset T$.
   - Alternative: use dune-polygongrid (not yet added)
2. Local Mapper μ_E: local dofs to global dofs
   - Based on local key approach (easy e.g. for dof tuple)
   - In agglomerated approach $\mu_T = \mu_E$ for all $T \subset E$.
3. Local Nodal Basis B_E: evaluate values, jacobians, hessians, ...
   In FE can start with any basis M_E of local space V_E.
   Use dofs to construct basis transform matrix A_E so that

$$\texttt{eval}(\texttt{E},\texttt{x}) = A_E M_E(x)\ , \quad \texttt{jac}(\texttt{E},\texttt{x}) = A_E \nabla M_E(x)\ .$$

   This does not work for VEM.
   Construct A_E^0, A_E^1, A_E^2

$$\texttt{eval}(\texttt{E},\texttt{x}) = \mathcal{G}^{E,0} B_E = A_E^0 M(x)\ , \quad \texttt{jac}(\texttt{E},\texttt{x}) = \mathcal{G}^{E,\nabla} B_E = A_E^1 M(x)\ .$$

   Instead of $\nabla^p A_E M_E$ implement function for $A_E^p M$ (same interface).
   Currently: basis in physical space.

[1] Dedner and Hodson, *Implementing general virtual element spaces*.

Implementation details[1]

1. Grid View T_h: consisting of elements E
   - Any (dune) grid
   - Agglomerated triangular grid: a triangle T knows polygon $E \supset T$.
   - Alternative: use dune-polygongrid (not yet added)
2. Local Mapper μ_E: local dofs to global dofs
   - Based on local key approach (easy e.g. for dof tuple)
   - In agglomerated approach $\mu_T = \mu_E$ for all $T \subset E$.
3. Local Nodal Basis B_E: evaluate values, jacobians, hessians, ...
   Construct A_E^0, A_E^1, A_E^2

   $$\texttt{eval}(\texttt{E},\texttt{x}) = \mathcal{G}^{E,0} B_E = A_E^0 M(x)\,, \quad \texttt{jac}(\texttt{E},\texttt{x}) = \mathcal{G}^{E,\nabla} B_E = A_E^1 M(x)\,.$$

   Instead of $\nabla^p A_E M_E$ implement function for $A_E^p M$ (same interface).
   Currently: basis in physical space.
4. Local Operators L_E: assemble local contributions
   No changes to FEM code or code generation.

   Only need `eval(E,x)`, `jac(E,x)`, ... at quadrature (currently given by agglomerated triangles).

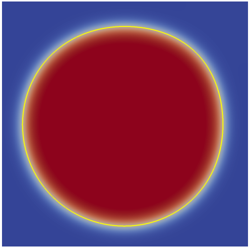
[1] Dedner and Hodson, *Implementing general virtual element spaces*.

Cahn-Hilliard Equation

Backward Euler in time with time step τ

$$\int_\Omega u^n + \tau\varepsilon^2 D^2 u^n : D^2 v + \tau \nabla \Psi'(u^n) \cdot \nabla v = \int_\Omega u^{n-1} v$$

with $\nabla u \cdot n = 0, \nabla(\varepsilon^2 \triangle u - \Psi'(u)) \cdot n = 0$ on $\partial\Omega$ and $\Psi(u) = (u^2-1)^2$



final state

Extension to some vector valued spaces

Example curl free space: used for mixed formulations of Laplace problem requiring space for $\sigma = \nabla u$ so $\mathrm{curl}\sigma = 0$.
Other application: Eigenvalue problem for $\int_\Omega \mathrm{div} u \, \mathrm{div} v$

Define subset of $H(\mathrm{div})$ so that on each element E

- $\mathrm{div} u^E \in \mathbb{P}_l(E)$
- $\mathrm{curl} u^E = 0$
- in addition $u^E\restriction_e \cdot n \in \mathbb{P}_l(e)$ on each edge.

Dofs: similar to Raviart-Thomas FEM spaces (but fewer)

$$\int_E u^E \cdot \nabla m \,, \; m \in \mathbb{P}_l(E) \setminus \mathbb{P}_0(E) \,, \qquad \int_e u^E \cdot nq \,, \; q \in \mathbb{P}_l(e)$$

Navier Stokes for (u, p)

$p \in \mathrm{DG}_0$ and u in compatible space i.e., $\mathrm{div} u \in \mathrm{DG}_0$ (order $l \geq 2$).

Top two rows: VEM velocity and pressure $l = 2$ (left) and $l = 4$ (right).

Lower row: Taylor-Hood of order $l = 2, 4$ for velocity on same grid.
Velocity dofs:

$$\int_E u^E \cdot m^\perp \;,\; m^\perp \in x^\perp \mathbb{P}_{l-3}(E)\,, \qquad \int_e u^E \cdot n q \;,\; q \in \mathbb{P}_{l-2}(e)\,, \qquad u^E(v)$$

Spaces useful for porous media (work in progress...)

Commercial break...

Now to something completely different

VTK reader (pickling support)

Recently we added pickling (backup/restore) to dune-common:

```
grid = dune.grid.structuredGrid( [-2,-2,-2],[2,2,2],[4,4,4] )
space = dune.fem.space.lagrange(grid, order=4)
df = space.interpolate(..., name="df")
with open(fileName+".dbf","wb") as f:
    dune.common.pickle.dump([df],f)
```

- uses `Dune::BackupRestoreFacility` for grids
- writes dofs of discrete functions (few lines of extra binding code)
- dump full Python object hierarchy (standard pickle)
- Extra: write source code for required generate modules
  Nice: can load onto any machine (with a dune installation).

VTK reader (pickling support)

Using paraview

- Class derived from `VTKPythonAlgorithmBase`
- Exporting env variable `export PV_PLUGIN_PATH=...`
- Can p- or h-refine the grid in paraview

Nice alternative (also for teaching):

`pyvista` e.g. in a notebook `https://pyvista.org`

VTK reader (pickling support)

Using paraview

- Class derived from `VTKPythonAlgorithmBase`
- Exporting env variable `export PV_PLUGIN_PATH=...`
- Can p- or h-refine the grid in paraview
- Can transform functions (using ufl) e.g. compute the error

```
def error(gv,t,df,dfs):
    ldf = dfs[0].localFunction()
    @gridFunction(gv,name="error",order=6)
    def _error(element,xLocal):
        ldf.bind(element)
        xGlobal = element.geometry.toGlobal(xLocal)
        exact = numpy.sin(numpy.pi*x.two_norm2)
        return abs( ldf(x) - exact )
    return [_error,*dfs]
```

Nice alternative (also for teaching):

`pyvista` e.g. in a notebook `https://pyvista.org`

Commercial break...

And now back to the main program...

Summary and Outlook

Summary

- General approach for constructing/implementing VEM spaces
- VEM spaces available in Dune with Python bindings (with UFL)
- Many versions for nonlinear second and forth order problems
- Extension to compatible VEM spaces (deRham complex)
- ... divergence/curl free spaces

Summary and Outlook

Summary

- General approach for constructing/implementing VEM spaces
- VEM spaces available in Dune with Python bindings (with UFL)
- Many versions for nonlinear second and forth order problems
- Extension to compatible VEM spaces (deRham complex)
- ... divergence/curl free spaces

Work in Progress

- Looking at fluid flow problems, e.g., CH-NS
- Looking at Eigenvalue problems (with L Alzaben, D. Boffi)
- Looking at isoparametric VEM (with A. Cangiani, H. Wells)

Summary and Outlook

Summary

- General approach for constructing/implementing VEM spaces
- VEM spaces available in Dune with Python bindings (with UFL)
- Many versions for nonlinear second and forth order problems
- Extension to compatible VEM spaces (deRham complex)
- ... divergence/curl free spaces

Work in Progress

- Looking at fluid flow problems, e.g., CH-NS
- Looking at Eigenvalue problems (with L Alzaben, D. Boffi)
- Looking at isoparametric VEM (with A. Cangiani, H. Wells)

Open

- 3D: concepts should carry over easily but
- ... implementation needs to be done
- Efficiency: so far code is mostly proof of concept
- ... some (threading/mpi) parallelization available
- ... want to add reference element caching where possible
- Adaptivity: refine polygons and prolongation/restriction missing

References

A. Dedner and A. Hodson. *Implementing general virtual element spaces*. 2022. arXiv: 2208.08978 [math.NA].

— . "Robust nonconforming virtual element methods for general fourth-order problems with varying coefficients". In: *IMA Journal of Numerical Analysis* (Mar. 2021). drab003. eprint: https://academic.oup.com/imajna/advance-article-pdf/doi/10.1093/imanum/drab003/36627651/drab003.pdf.