Function space bases in the dune-functions module

Oliver Sander

Heidelberg, 28. 9. 2015
Discretization modules

**dune-fem**
- Focus on adaptivity, parallelism, and efficiency

**dune-pdelab**
- Very flexible and powerful
- Steep learning curve

**dune-fufem**
- Easy to use
- Less powerful
New module: dune-functions

The idea:
- Standardize on parts of the functionality

The team
- Carsten
- Christian
- Steffen
- Yours truly

History
- First meeting: Aug. 2013 in Münster (with Christoph Gersbacher and Stefan Girke)
- Further meetings every six months
- First actual users in March 2015
Functions

- Interface for functions $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, differentiable functions, grid functions, etc.
- Based on callables, concepts and type erasure
- Talk by Carsten

Function space bases

- Content of this talk

Infrastructure

- Interpolation:
  
  function + basis $\Rightarrow$ coefficient vector

- VTK output of grid functions
The case for bases

- Grid function spaces are *not* the right abstraction
- More than one basis for the same space
  - E.g., P2 nodal basis vs. hierarchical basis
  - Orthogonal vs. Lagrange DG basis
- Basis + coefficients = discrete function

Functionality of a basis For any given grid element

- ...get restrictions of relevant basis functions to this element
  - i.e., the shape functions
  - use dune-localfunctions interfaces
- ...get local shape function numbers
- ...get global basis function numbers
Tree representation of composite bases

Systematic construction of basis for vector-valued spaces

- Tensor products of simpler basis
- Taylor–Hood: $B_{\text{TH}} = (P_2 \otimes P_2 \otimes P_2) \otimes P_1$

Tree representation

Systematic construction of

- orderings
- multi-indices
### Taylor–Hood basis: lexicographic ordering

|       | 0      | (0,0)       | (0,0)       | (0,0,0)       |
|-------|--------|-------------|-------------|---------------|
| $b_{x,0}$ | 1      | (0,1)       | (0,1)       | (0,0,1)       |
| $b_{x,1}$ | 2      | (0,2)       | (0,2)       | (0,0,2)       |
| $b_{x,2}$ | ...    | ...         | ...         | ...           |
| $b_{y,0}$ | $n$    | (0,$n$)     | (1,0)       | (0,1,0)       |
| $b_{y,1}$ | $n+1$  | (0,$n+1$)   | (1,1)       | (0,1,1)       |
| $b_{y,2}$ | $n+2$  | (0,$n+2$)   | (1,2)       | (0,1,2)       |
| $b_{z,0}$ | $2n$   | (0,2$n$)    | (2,0)       | (0,2,0)       |
| $b_{z,1}$ | $2n+1$ | (0,2$n+1$)  | (2,1)       | (0,2,1)       |
| $b_{z,2}$ | $2n+2$ | (0,2$n+2$)  | (2,2)       | (0,2,2)       |
| $p_0$   | 3$n$   | (1,0)       | $n$         | (1,0)         |
| $p_1$   | 3$n+1$ | (1,1)       | $n+1$       | (1,1)         |
| $p_2$   | 3$n+2$ | (1,2)       | $n+2$       | (1,2)         |

Possible index types for a Taylor–Hood basis with lexicographic ordering of the velocity basis functions.
Orderings and indices

**Taylor–Hood basis: interleaved ordering**

|      | $b_{x,0}$ | $b_{y,0}$ | $b_{z,0}$ | $b_{x,1}$ | $b_{y,1}$ | $b_{z,1}$ | $b_{x,2}$ | $b_{y,2}$ | $b_{z,2}$ | $p_0$ | $p_1$ | $p_2$ |
|------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-------|-------|-------|
|      | 0         | 1         | 2         | 3         | 4         | 5         | 6         | 7         | 8         | 3n   | 3n+1 | 3n+2 |
|      | $(0,0)$   | $(0,1)$   | $(0,2)$   | $(0,3)$   | $(0,4)$   | $(0,5)$   | $(0,6)$   | $(0,7)$   | $(0,8)$   | $(1,0)$| $(1,1)$| $(1,2)$|
|      |           |           |           |           |           |           |           |           |           |       |       |       |
|      |           |           |           |           |           |           |           |           |           |       |       |       |

Possible index types for a Taylor–Hood basis with interleaved ordering of the velocity basis functions
Figure: Overview of the classes making up the interface to finite element space bases
FunctionSpaceBasis

Interface

- size_type size() const
  Total number of basis functions

- size_type size(const SizePrefix& prefix) const
  Number of basis functions with a given multi-index prefix

- LocalView localView() const
  Get a local view object

- LocalIndexSet localIndexSet() const
  Get a local index object
Interface

- **void bind(const Element& e)**
  Bind the view to grid element e

- **const Tree& tree() const**
  Get the shape function tree for the current element

- **size_type size() const**
  Total number of shape functions on the current element

- **size_type maxSize() const**
  Maximum number of shape functions over all elements
Leaf nodes
- const FiniteElement& finiteElement() const
- size_type localIndex(size_type i) const

Inner nodes
- PowerNode: Combines identical subtrees
- CompositeNode: Combines differing subtrees

Node access
- tree.child(a,b,c,...)
  with a,b,c,... either int or std::integral_constant<size_type,>
- Example: tree.child(_0,0): first component of velocity basis
LocalIndexSet

Interface

- `void bind(const LocalView& localView)`
  Bind to `localView` object
- `size_type size() const`  
  Total number of shape functions for the current element
- `MultiIndex index(size_type i) const`  
  Get global (multi-)index for the `i`-th shape function

Open question:

- How to request *different* orderings / index types?
Example: Stokes equation

Setting
- Models a viscous incompressible fluid in a \(d\)-dimensional domain \(\Omega\).
- Unknowns: fluid velocity field \(u : \Omega \rightarrow \mathbb{R}^d\), pressure \(p : \Omega \rightarrow \mathbb{R}\).
- The pressure is therefore usually normalized such that \(\int_{\Omega} p \, dx = 0\).

Weak form
- Spaces
  \[
  H^1_D(\Omega) := \{v \in H^1(\Omega) : \text{tr } v = u_D\},
  L^2,0(\Omega) := \{q \in L^2(\Omega) : \int_{\Omega} q \, dx = 0\},
  \]
- Bilinear forms
  \[
  a(u, v) := \int_{\Omega} \nabla u \nabla v \, dx, \quad \text{and} \quad b(v, q) := \int_{\Omega} \nabla \cdot v \cdot q \, dx.
  \]
- Saddle-point problem: Find \((u, p) \in H^1_D(\Omega) \times L^2,0(\Omega)\) such that
  \[
  a(u, v) + b(v, p) = 0 \quad \text{for all } v \in H^1_0(\Omega)
  \]
  \[
  b(u, q) = 0 \quad \text{for all } q \in L^2,0(\Omega).
  \]
Example: Driven cavity

Figure: Left: setting, right: simulation result. The arrows show the normalized velocity.
Current status

Technology preview

- Most work is done
- Details of the API may still change(!)
- Go use it!

Basis implementations

- PQkNodalBasis
- LagrangeDGBasis
- TaylorHoodBasis
- BSplineBasis
- ...more to come

Further information

- www.dune-project.org/modules/dune-functions