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
dune-functions: Functionality

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:
  \[
  \text{function} + \text{basis} \Rightarrow \text{coefficient vector}
  \]
- 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: \( B_{TH} = (P_2 \otimes P_2 \otimes P_2) \otimes P_1 \)

Tree representation

Systematic construction of

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

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{x,0}$</td>
<td>0</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0, 0)</td>
</tr>
<tr>
<td>$b_{x,1}$</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(0, 0, 1)</td>
</tr>
<tr>
<td>$b_{x,2}$</td>
<td>2</td>
<td>(0, 2)</td>
<td>(0, 2)</td>
<td>(0, 0, 2)</td>
</tr>
<tr>
<td>$b_{x,3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_{y,0}$</td>
<td>$n$</td>
<td>(0, $n$)</td>
<td>(1, 0)</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>$b_{y,1}$</td>
<td>$n + 1$</td>
<td>(0, $n + 1$)</td>
<td>(1, 1)</td>
<td>(0, 1, 1)</td>
</tr>
<tr>
<td>$b_{y,2}$</td>
<td>$n + 2$</td>
<td>(0, $n + 2$)</td>
<td>(1, 2)</td>
<td>(0, 1, 2)</td>
</tr>
<tr>
<td>$b_{y,3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_{z,0}$</td>
<td>$2n$</td>
<td>(0, 2$n$)</td>
<td>(2, 0)</td>
<td>(0, 2, 0)</td>
</tr>
<tr>
<td>$b_{z,1}$</td>
<td>$2n + 1$</td>
<td>(0, 2$n + 1$)</td>
<td>(2, 1)</td>
<td>(0, 2, 1)</td>
</tr>
<tr>
<td>$b_{z,2}$</td>
<td>$2n + 2$</td>
<td>(0, 2$n + 2$)</td>
<td>(2, 2)</td>
<td>(0, 2, 2)</td>
</tr>
<tr>
<td>$b_{z,3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_0$</td>
<td>$3n$</td>
<td>(1, 0)</td>
<td>$n$</td>
<td>(1, 0)</td>
</tr>
<tr>
<td>$p_1$</td>
<td>$3n + 1$</td>
<td>(1, 1)</td>
<td>$n + 1$</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$3n + 2$</td>
<td>(1, 2)</td>
<td>$n + 2$</td>
<td>(1, 2)</td>
</tr>
</tbody>
</table>

Possible index types for a Taylor–Hood basis with lexicographic ordering of the velocity basis functions.
Taylor–Hood basis: interleaved ordering

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{x,0}$</td>
<td>1</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>(0, 0, 1)</td>
</tr>
<tr>
<td>$b_{y,0}$</td>
<td>2</td>
<td>(0, 2)</td>
<td>(0, 2)</td>
<td>(0, 0, 2)</td>
</tr>
<tr>
<td>$b_{z,0}$</td>
<td>3</td>
<td>(0, 3)</td>
<td>(1, 0)</td>
<td>(0, 1, 0)</td>
</tr>
<tr>
<td>$b_{x,1}$</td>
<td>4</td>
<td>(0, 4)</td>
<td>(1, 1)</td>
<td>(0, 1, 1)</td>
</tr>
<tr>
<td>$b_{y,1}$</td>
<td>5</td>
<td>(0, 5)</td>
<td>(1, 2)</td>
<td>(0, 1, 2)</td>
</tr>
<tr>
<td>$b_{z,1}$</td>
<td>6</td>
<td>(0, 6)</td>
<td>(2, 0)</td>
<td>(0, 2, 0)</td>
</tr>
<tr>
<td>$b_{x,2}$</td>
<td>7</td>
<td>(0, 7)</td>
<td>(2, 1)</td>
<td>(0, 2, 1)</td>
</tr>
<tr>
<td>$b_{y,2}$</td>
<td>8</td>
<td>(0, 8)</td>
<td>(2, 2)</td>
<td>(0, 2, 2)</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
<tr>
<td>$p_0$</td>
<td>$3n$</td>
<td>(1, 0)</td>
<td>$n$</td>
<td>(1, 0)</td>
</tr>
<tr>
<td>$p_1$</td>
<td>$3n + 1$</td>
<td>(1, 1)</td>
<td>$n + 1$</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>$p_2$</td>
<td>$3n + 2$</td>
<td>(1, 2)</td>
<td>$n + 2$</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
</tr>
</tbody>
</table>

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
- \texttt{const FiniteElement& finiteElement()} const
- \texttt{size_type localIndex(size_type i) const}

Inner nodes
- \texttt{PowerNode}: Combines identical subtrees
- \texttt{CompositeNode}: Combines differing subtrees

Node access
- \texttt{tree.child(a,b,c,...)},
  with \(a,b,c,...\) either \texttt{int} or \texttt{std::integral_constant<size_type,>}
- Example: \texttt{tree.child(_0,0)}: first component of velocity basis
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 $u : \Omega \to \mathbb{R}^d$, pressure $p : \Omega \to \mathbb{R}$.
- The pressure is therefore usually normalized such that $\int_\Omega p \, dx = 0$.

Weak form
- Spaces
  \[
  H^1_D(\Omega) := \{ v \in H^1(\Omega) : \text{tr} \, v = u_D \},
  \]
  \[
  L_{2,0}(\Omega) := \left\{ q \in L^2(\Omega) : \int_\Omega q \, dx = 0 \right\},
  \]

- 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