Master in High Performance Computing

Theory and Practice of Finite Element Methods

Handling systems of PDEs

Luca Heltai <luca.heltai@sissa.it>

International School for Advanced Studies (www.sissa.it)
Mathematical Analysis, Modeling, and Applications (math.sissa.it)
Master in High Performance Computing (www.mhpc.it)
SISSA mathLab (mathlab.sissa.it)

sissa mathLab innovating with mathematics

ICTP The Abdus Salam International Centre for Theoretical Physics

SISSA

Vector valued problems

- In reality, most problems are not scalar, i.e., they have more than one solution variable. We call them vector-valued.

- Example: The mixed Poisson equation:

$$\begin{aligned} K^{-1}\mathbf{u} + \nabla p &= 0 \\ \nabla \mathbf{u} &= g \end{aligned} \quad \Longleftrightarrow \quad \begin{aligned} -\nabla \cdot (K \nabla p) &= g \\ -K \nabla p &= \mathbf{u} \end{aligned}$$

Vector valued problems

- In reality, most problems are not scalar, i.e., they have more than one solution variable. We call them vector-valued.

- Example: The Stokes equation

$$-\nabla\cdot(\eta\nabla\mathbf{u})+\nabla p=f$$
$$\nabla\mathbf{u}=0$$

Vector valued problems

- A systematic way to treat vector-valued problems:
  - Write the solution in the product space, using the graph norm, i.e., $\mathbf{u} \in V$, $p \in Q$, and $\psi \in V \times Q \equiv \mathbb{V}$
  - Write the test functions as $\mathbf{v} \in V$, $q \in Q$, and collect them as $\phi \in V \times Q \equiv \mathbb{V}$
  - Write the functionals $f \in V'$ and $g \in Q'$ as $\mathbb{F} \in V' \times Q'$
  - Write the operator as $\mathbb{A} : \mathbb{V} \mapsto \mathbb{V}'$

$$\psi = \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} \qquad \mathbb{A} = \begin{pmatrix} \mathbb{A}_{uu} & \mathbb{A}_{up} \\ \mathbb{A}_{pu} & \mathbb{A}_{pp} \end{pmatrix} \qquad \phi = \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix}$$

$$\langle \mathbb{A}\psi, \phi \rangle = \langle \mathbb{F}, \phi \rangle$$

Stokes problem

$$V = H_0^1(\Omega)^d \qquad Q = L_0^2(\Omega)$$

$$\begin{aligned} (\nabla \mathbf{u}, \nabla \mathbf{v}) - (\nabla \cdot \mathbf{v}, p) &= (f, \mathbf{v}) \qquad \forall \mathbf{v} \in V \\ (\nabla \cdot \mathbf{u}, q) &= 0 \qquad \forall q \in Q \end{aligned} \qquad \Longleftrightarrow \qquad \langle \mathbb{A}\psi, \phi \rangle = \langle \mathbb{F}, \phi \rangle$$

$$\psi = \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} \qquad \mathbb{A} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \qquad \phi = \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix}$$

$$\langle Au, v \rangle := (\nabla \mathbf{u}, \nabla \mathbf{v}) \qquad \langle Bv, q \rangle := (\nabla \cdot \mathbf{v}, q)$$

Accessing subspaces

$$\psi = \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} \qquad \mathbb{A} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \qquad \phi = \begin{pmatrix} \mathbf{v} \\ q \end{pmatrix} \qquad \langle \mathbb{A}\psi, \phi \rangle = \langle \mathbb{F}, \phi \rangle$$

- Use symbolic names (*not indices:* extractor objects) to index subvectors/subspaces:
  - ϕ_u, ϕ_p translate into `phi[velocity], phi[pressure]`
  - ψ_u, ψ_p translate into `psi[velocity], psi[pressure]`

Assembly of vector valued problems

- Assemble systems and rhs using the following identities:
  - $\phi_{i,u}(x_q) \implies$ `fe_values[velocity].value(i,q)`
  - $\phi_{i,p}(x_q) \implies$ `fe_values[pressure].value(i,q)`
  - $\nabla\phi_{i,u}(x_q) \implies$ `fe_values[velocity].gradient(i,q)`
  - $\nabla\cdot\phi_{i,u}(x_q) \implies$ `fe_values[velocity].divergence(i,q)`

Defining the finite element

- We defined solution, shape functions, and test functions as having multiple components.
- Each component is usually built from a simpler element.
- Example 1: The Taylor-Hood element for 2d Stokes

```
FESystem<2> stokes_element (FE_Q<2>(2), 1,     // one copy of FE_Q(2) for u_x
                            FE_Q<2>(2), 1,     // one copy of FE_Q(2) for u_y
                            FE_Q<2>(1), 1);    // one copy of FE_Q(1) for p
```

Describing logical connections

- You know which components logically form a vector or are scalars
- The visualization program doesn't.
- Solution:
  - You need to describe it to the DataOut class when adding a solution vector
  - DataOut can then represent this information in the output file

Describing logical connections

```
std::vector<std::string> solution_names (dim, "velocity");
solution_names.push_back ("pressure");

std::vector<DataComponentInterpretation::DataComponentInterpretation>
      data_component_interpretation
      (dim, DataComponentInterpretation::component_is_part_of_vector);
data_component_interpretation
      .push_back (DataComponentInterpretation::component_is_scalar);

data_out.add_data_vector(solution, solution_names,
                         DataOut<dim>::type_dof_data,
                         data_component_interpretation);
```