MATH 676

–

Finite element methods in scientific computing

Wolfgang Bangerth, Texas A&M University

Lecture 19:

Problems with more than one solution variable

(“Vector-valued” problems)

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 Laplace equation:

$$\begin{aligned} K^{-1}u + \nabla p &= 0 \\ -\nabla \cdot u &= -f \end{aligned}$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 1: Write the solution as a vector and the operator as a matrix:

$$\begin{array}{rcl} K^{-1}u + \nabla p & = & 0 \\ -\nabla \cdot u & = & -f \end{array} \longrightarrow \begin{pmatrix} K^{-1} & \nabla \\ -\nabla \cdot & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} 0 \\ -f \end{pmatrix} \longrightarrow \begin{pmatrix} K^{-1} & \nabla \\ -\nabla \cdot & 0 \end{pmatrix} U = F$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 1: Write the solution as a vector and the operator as a matrix.

Note: We can consider *U* to be a vector that consists of several components. We can either see it as

$$U = \begin{pmatrix} u \\ p \end{pmatrix}$$

where *u* is itself a vector, or as

$$U = \begin{pmatrix} u_x \\ u_y \\ p \end{pmatrix} \qquad \text{or} \qquad U = \begin{pmatrix} u_x \\ u_y \\ u_z \\ p \end{pmatrix}$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 1: Write the solution as a vector and the operator as a matrix.

Note: We will express this composition via *subscripts*. Let "*velocities*" and "*pressure*" be symbolic names, then

$$U[\text{velocities}] = u$$
$$U[\text{pressure}] = p$$

Here,

$$U[\text{velocities}] \text{ is a dim-dimensional vector}$$
$$U[\text{pressure}] \text{ is a scalar}$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 2: Multiply from the left with a vector-valued test function and integrate:

$$\int_\Omega \Phi^T \begin{pmatrix} K^{-1} & \nabla \\ -\nabla\cdot & 0 \end{pmatrix} U = \int_\Omega \Phi^T F$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 2: Multiply from the left with a vector-valued test function and integrate:

$$\int_\Omega \Phi^T \begin{pmatrix} K^{-1} & \nabla \\ -\nabla\cdot & 0 \end{pmatrix} U = \int_\Omega \Phi^T F$$

Note: We can again consider the test function to be composed:

$$\Phi = \begin{pmatrix} \varphi_u \\ \varphi_p \end{pmatrix}$$

And with subscription:

$$\Phi[\text{velocity}] = \varphi_u$$
$$\Phi[\text{pressure}] = \varphi_p$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 3: Multiply out the various terms:

$$(\varphi_u, K^{-1} u) + (\varphi_u, \nabla p) - (\varphi_p, \nabla \cdot u) = (\varphi_p, -f)$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 3: Multiply out the various terms:

$$(\varphi_u, K^{-1}u) + (\varphi_u, \nabla p) - (\varphi_p, \nabla \cdot u) = (\varphi_p, -f)$$

Step 4: Integrate by parts where necessary:

$$(\varphi_u, K^{-1}u) - (\nabla \cdot \varphi_u, p) - (\varphi_p, \nabla \cdot u) = (\varphi_p, -f)$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 5: Expand the solution as

$$U_h = \sum_j U_j \Phi_j(x)$$

And test with a particular test function, namely shape function *i*:

$$\sum_j \left[(\varphi_{i,u}, K^{-1} \varphi_{j,u}) - (\nabla \cdot \varphi_{i,u}, \varphi_{j,p}) - (\varphi_{i,p}, \nabla \cdot \varphi_{j,u}) \right] U_j = (\varphi_{i,p}, -f)$$

Vector-valued problems

A systematic way to treat vector-valued problems:

Step 6: Translate the bilinear form (and similarly: the rhs)

$$A_{ij} = (\varphi_{i,u}, K^{-1}\varphi_{j,u}) - (\nabla\cdot\varphi_{i,u}, \varphi_{j,p}) - (\varphi_{i,p}, \nabla\cdot\varphi_{j,u})$$

into source code using these identities:

$$\varphi_{i,u}(x_q) \;=\; \text{fe\_values[velocities].value(i,q)}$$
$$\varphi_{i,p}(x_q) \;=\; \text{fe\_values[pressure].value(i,q)}$$
$$\nabla\cdot\varphi_{i,u}(x_q) \;=\; \text{fe\_values[velocities].divergence(i,q)}$$

Vector-valued problems

A systematic way to treat vector-valued problems:

```
for (unsigned int q=0; q<n_q_points; ++q)
  for (unsigned int i=0; i<dofs_per_cell; ++i)  {
      const Tensor<1,dim> phi_i_u  = fe_values[velocities].value (i, q);
      const double        div_phi_i_u = fe_values[velocities].divergence (i, q);
      const double        phi_i_p       = fe_values[pressure].value (i, q);

      for (unsigned int j=0; j<dofs_per_cell; ++j) {
        const Tensor<1,dim> phi_j_u = fe_values[velocities].value (j, q);
        const double        div_phi_j_u = fe_values[velocities].divergence (j, q);
        const double        phi_j_p       = fe_values[pressure].value (j, q);

        local_matrix(i,j) += (phi_i_u * k_inverse_values[q] * phi_j_u
                              - div_phi_i_u * phi_j_p
                              - phi_i_p * div_phi_j_u)
                              * fe_values.JxW(q);
      }
```

What else needs to be adjusted

Part 1: 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

$$U=\begin{pmatrix} u_x \\ u_y \\ p \end{pmatrix} \qquad \Rightarrow \qquad U \in Q_2 \times Q_2 \times Q_1$$

is represented by:

```
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
```

What else needs to be adjusted

Part 1: 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 generic Stokes

$$U=\begin{pmatrix} u \\ p \end{pmatrix} \qquad \Rightarrow \qquad U \in Q_2^{\text{dim}} \times Q_1$$

is represented by:

```
FESystem<dim> stokes_element (FE_Q<dim>(2), dim,  // dim copies of FE_Q(2) for u
                              FE_Q<dim>(1), 1);    // one copy of FE_Q(1) for p
```

What else needs to be adjusted

Part 1: 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 2: Raviart-Thomas for the mixed Laplace where

$$U = \begin{pmatrix} u \\ p \end{pmatrix} \qquad \Rightarrow \qquad U \in RT_k \times Q_k$$

is represented by:

```
FESystem<dim> mixed_laplace_element (FE_RaviartThomas<dim>(k), 1,  // u
                                     FE_Q<dim>(k), 1);              // p
```

Note: The RT element itself has *dim* components.

What else needs to be adjusted

Part 2: Describing logical connections for graphical output

- *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

What else needs to be adjusted

Part 2: Describing logical connections for graphical output

For example, for mixed Laplace or Stokes (from step-22):

```
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);
```

MATH 676

–

Finite element methods in scientific computing

Wolfgang Bangerth, Texas A&M University