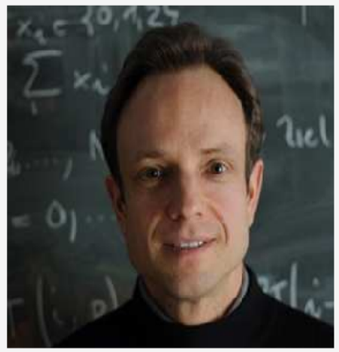
ETH Lecture 401-0674-00L Numerical Methods for Partial Differential Equations

Homework Problem Video Tutorial
Problem 3-2: Debugging Finite Element Codes

Prof. R. Hiptmair, SAM, ETH Zurich

Date: March 29, 2019



$$u \in H^1(\Omega): \quad a(u,v) := \int_\Omega \mathbf{grad}u \cdot \mathbf{grad}v \, dx = \ell(v) := \int_\Omega f(x)v(x)\,dx \quad \forall v \in H^1(\Omega) \qquad (3.2.1)$$

$S_2^0(\mathcal{M})$ - FEM ENTITY_MATRIX_PROVIDERS

↑ LocalLaplacQFE* , * = 1,2,3

triangular meshes

a)

Implement a C++ function

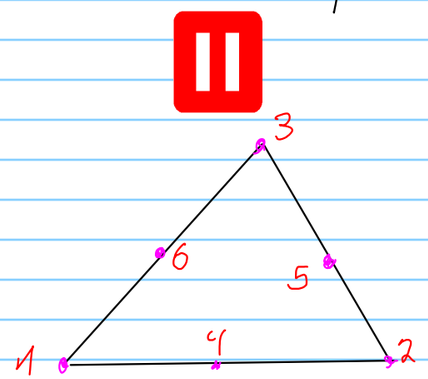
```
template <typename FUNCTOR>
  Eigen::VectorXd interpolateOntoQuadFE(const
    lf::assemble::DofHandler &dofh, FUNCTOR &&f);
```

that accepts a **lf::assemble::DofHandler** argument `dofh` for the finite element space $\mathcal{S}_2^0(\mathcal{M})$ and a functor object for a continuous function $f \in C^0(\overline{\Omega})$. It should return a vector with the basis coefficients of the nodal interpolant $I_2 f \in \mathcal{S}_2^0(\mathcal{M})$. This particular piecewise quadratic interpolation is presented in [Lecture → Ex. 2.6.1.2].

Do not forget to equip your function with checks for consistency using the `LF_ASSERT_MSG()` macro.

$$I_2 f = \sum_{p \in \mathcal{N}} f(p)\, b_h^p \Rightarrow I_2 f(p) = f(p) \quad \forall p \in \mathcal{N}$$

due to cardinal basis property



Here: No assembly!

Set entries of coeff. vector instead of adding them

(2)

C++11 code 3.2.2: Sub-problem (3-2.a): function `interpolateOntoQuadFE()` → GITLAB

```
template <typename FUNCTOR>
Eigen::VectorXd interpolateOntoQuadFE(const lf::assemble::DofHandler &dofh,
                                      FUNCTOR &&f) {
  // get mesh and set up finite element space
  auto mesh = dofh.Mesh();

  // initialize result vector
  const size_type N_dofs(dofh.NoDofs());
  Eigen::VectorXd result = Eigen::VectorXd::Zero(N_dofs);

  /* SOLUTION_BEGIN */
  for (const lf::mesh::Entity &cell : mesh->Entities(0)) {
    // get local to global map for the cell
    auto glob_ind = dofh.GlobalDofIndices(cell);
    for (int i = 0; i < 6; i++) {
      // update the result vector
      auto coords = globalCoordinate(i, cell);
      result(glob_ind[i]) = f(coords);
    }
  }
  /* SOLUTION_END */
  return result;
}
```

← coordinates of local interpolations node

↑ no "+=" !

Theorem 3.2.4. Interpolation error estimate for quadratic Lagrangian FEM

Let $\Omega \subset \mathbb{R}^d$, $d = 1,2,3$, be a bounded polygonal/polyhedral domain equipped with a simplicial mesh $\mathcal{M}$. Then the following interpolation error estimate holds for the nodal interpolation operator I_2 onto $\mathcal{S}_2^0(\mathcal{M})$

$$\|u - I_2 u\|_{H^1(\Omega)} \leq C h^{\min\{3,k\}-1} |u|_{H^k(\Omega)} \quad \forall u \in H^k(\Omega), \quad k = 2,3\,,$$

with a constant $C > 0$ depending only on k and the shape regularity measure $\rho_{\mathcal{M}}$.

"$a^2 - b^2 = (a+b)(a-b)$"

$$\begin{aligned}
|a(u,u) - a(I_2u, I_2u)| &= |a(u + I_2u, u - I_2u)| \\
&\leq \|u + I_2u\|_A \|u - I_2u\|_A && \text{(Cauchy-Schwarz)} \\
&\leq (\|u\|_A + \|I_2u\|_A)\|u - I_2u\|_A && (\Delta\text{-ineq.}) \\
&= (\|u\|_A + \|I_2u - u + u\|_A)\|u - I_2u\|_A && \text{(Add zero trick)} \\
&\leq (2\|u\|_A + \|u - I_2u\|_A)\|u - I_2u\|_A && (\Delta\text{-ineq. and } \|-a\| = \|a\|) \\
&\leq C_1 |u|_{H^1(\Omega)} |u|_{H^3(\Omega)} h^2 + C_2 |u|^2_{H^3(\Omega)} h^4 && \text{Use theorem above} \\
&\leq C_3 |u|_{H^1(\Omega)} |u|_{H^3(\Omega)} h^2. = O(h_{\mathcal{M}}^2) = O(N^{-1})
\end{aligned}$$

$N := \dim \mathcal{S}_2^0(\mathcal{M}) \approx h_{\mathcal{M}}^{-2}$

c) Debugging strategy monitor

$|a(u,u) - a(I_2u, I_2u)|$

$u \in C^\infty(\overline{\Omega})$ known analytically

for a sequence of meshes generated by uniform regular refinement

(3) In the file `qfe_provider_tester.h` flesh out the implementation of the templated class

```
template <typename ENTITY_MATRIX_PROVIDER>
class QFEProviderTester {
public:
  QFEProvideTester(lf::assemble::DofHandler &dofh,
                   ENTITY_MATRIX_PROVIDER &element_matrix_provider);

  template <typename FUNCTOR>
    double energyOfInterpolant(FUNCTOR &&u) const;
private:
  lf::assemble::DofHandler &dofh_;
  ENTITY_MATRIX_PROVIDER &element_matrix_provider_;
  /* Further private data members */
};
```

→ $a(I_2u, I_2u)$

The template parameter **ENTITY_MATRIX_PROVIDER** must be a type that fits the local Laplace element matrix provider classes **LocalLaplaceQFEX**, $X = 1,2,3$.
The method `energyOfInterpolant()` should accepts a functor object `u` encoding a continuous function u defined on the domain covered by the mesh associated with the `dofh` object with which the current instance of **QFEProviderTester** was initialized. The method should return $a(I_2u, I_2u)$ based on the element matrix provider object with which the current object of type **QFEProviderTester** was set up.

C++11 code 3.2.7: Implementation of member function `energyOfInterpolant()` of class QFEProvideTester ➽ GITLAB

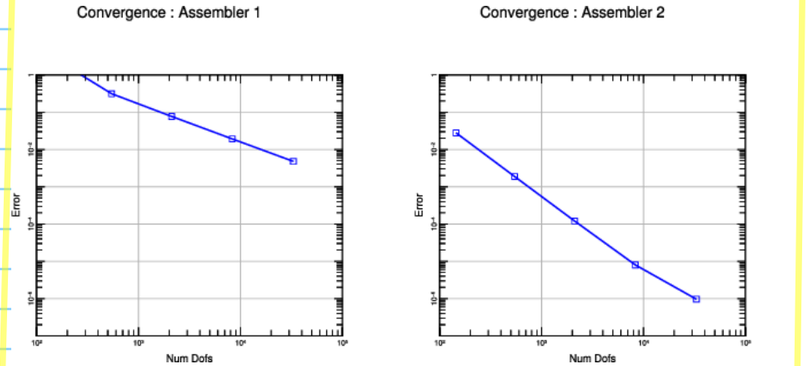
```
template <typename ENTITY_MATRIX_PROVIDER>
template <typename FUNCTOR>
double QFEProviderTester<ENTITY_MATRIX_PROVIDER>::energyOfInterpolant(
    FUNCTOR &&u) const {
  double energy;
  /* SOLUTION_BEGIN */
  Eigen::VectorXd eta = DebuggingFEM::interpolateOntoQuadFE(dofh_, u);
  energy = eta.dot(A_ * eta);
  /* SOLUTION_END */
  return energy;
}
```

↑ $S_2^0(M)$ - Galerkin matrix from ~~using the underlying ENTITY_MATRIX_PROVIDER~~ for assembly

$$a(I_2u, I_2u) = \vec{\eta}^T A \vec{\eta}$$

↳ coeff. vec. $\vec{\eta} = [f(p)]_{p \in \mathcal{N}} \in \mathbb{R}^N$

e)



Alg. cvg. $O(N^{-1}) = O(h_M^2)$

Alg. cvg. $O(N^{-2}) = O(h_M^4)$

no cvg.

f) Assembler 3 ⇒ flawed

Assembler 1 & 2 ⇒ pass test

[estimate from c) too crude to rule out that Assembler 1 is correct]