FEniCS Course

Lecture 5: Happy hacking
Tools, tips and coding practices

Contributors
André Massing

Post-processing

Function evaluation

`Expression` and `Function` objects `f` can be evaluated at arbitrary points:

Python code

```
# 1D
x = 0.5
f(x)
# 2D
x = (0.5,0.3) # tuple
# x = [0.5,0.3] is also valid
f(x)
# 3D
x = (0.5,0.2,1.0) # tuple
# x = [0.5,0.2,1.0] is also valid
f(x)
print f(x)
```

Short-hand

Python code

```
f(0.5,0.5)
```

Exercise: Try it out! Use one of your existing codes and evaluate the solution at some point.

Function evalution vs. Function representation

Question: What about plotting $\sin(u_h)$? And ∇u_h and $|\nabla u_h|$?

Experiment: Try it out! Use

Python code

```python
sqrt(grad(u)**2)
```

for $|\nabla u|$. What happens if you plot these function? Have a closer look at the terminal output. Anything suspicious?

Question: What happened now? Why is there a

`> Object cannot be plotted directly, projecting to piecewise linears.`

Answer:

- $\sin(u_h(x))$ is the evaluation of the built-in function sin at a *given* value $u_h(x)$, which in turn results from a FEM function evalution.
- $\sin \circ u_h$ is a composition of the built-in function sin and a FEM function u_h. The composition is a symbolic `UFL` (Unified Form Language) expression.

Building FE representations via L^2 projection

Define $f = \sin \circ u_h$ and choose a FEM function space $\widetilde{V}_h \subset L^2(\Omega)$ which is "suitable" for your post-process.

Find $w_h \in \widetilde{V}_h \subset L^2(\Omega)$ such that for all $v_h \in \widetilde{V}_h$

$$\underbrace{\int_\Omega w_h v \,\mathrm{d}x}_{a(u,v)} = \underbrace{\int_\Omega f v \,\mathrm{d}x}_{L(v)}$$

Exercise: Compute $|\nabla(u)|$ for the solution from one of your existing solvers. Start with adding

Python code

```
abs_grad_V = FunctionSpace(mesh,"DG",0)
f = sqrt(grad(u)**2)
```

to your original Python script.

A hack to plot $\nabla(u)$ only on $\partial\Omega$

Python code

```
V_ag = FunctionSpace(mesh,"Lagrange",1)
#V_ag = FunctionSpace(mesh,"DG",0)
f = sqrt(grad(u)**2)

# Do the Projection only on the boundary
u_ag = TrialFunction(V_ag)
v = TestFunction(V_ag)
a = u_ag*v*ds
L = f*v*ds
A = assemble(a)
b = assemble(L)

# Set dofs not located on the boundary to
# zero by adding ones in the diagonal of A
A.ident_zeros()
u_ag = Function(V_ag)
solve(A, u_ag.vector(), b)

plot(u_ag, title="|grad(u)| on boundary")
interactive()
```

Simple code validation

Theory can help you to validate your implementation!

A priori estimates for the Poisson problem

If

- $u \in H_0^1(\Omega) \cap H^{k+1}(\Omega)$
- $V_h = \{v_h \in C(\Omega) : v_h \in P^k(T)\ \forall\, T \in \mathcal{T}\}$

then

$$E_1(h) := \|u - u_h\|_{1,\Omega} \leq C h^k \|u\|_{k+1,\Omega}$$

$$E_0(h) := \|u - u_h\|_{0,\Omega} \leq C h^{k+1} \|u\|_{k+1,\Omega}$$

where $\|\cdot\|_{l,\Omega} = \|\cdot\|_{H^l(\Omega)}$ for $l = 0, 1, k+1$.

Theory can help you to validate your implementation!

A priori estimates for the Poisson problem

If

- $u \in H_0^1(\Omega) \cap H^{k+1}(\Omega)$
- $V_h = \{v_h \in C(\Omega) : v_h \in P^k(T) \ \forall\, T \in \mathcal{T}\}$

then

$$E_1(h) := \|u - u_h\|_{1,\Omega} \leq Ch^k \|u\|_{k+1,\Omega}$$

$$E_0(h) := \|u - u_h\|_{0,\Omega} \leq Ch^{k+1} \|u\|_{k+1,\Omega}$$

where $\|\cdot\|_{l,\Omega} = \|\cdot\|_{H^l(\Omega)}$ for $l = 0, 1, k+1$.
Taking log on each side

$$\log(E_1(h)) \leq \log(Ch^k \|u\|_{k+1,\Omega}) = k\log(h) + \log(C\|u\|_{k+1,\Omega})$$

Theory can help you to validate your implementation!

A priori estimates for the Poisson problem

If

- $u \in H_0^1(\Omega) \cap H^{k+1}(\Omega)$
- $V_h = \{v_h \in C(\Omega) : v_h \in P^k(T) \; \forall \, T \in \mathcal{T}\}$

then

$$E_1(h) := \|u - u_h\|_{1,\Omega} \leq Ch^k \|u\|_{k+1,\Omega}$$
$$E_0(h) := \|u - u_h\|_{0,\Omega} \leq Ch^{k+1} \|u\|_{k+1,\Omega}$$

where $\|\cdot\|_{l,\Omega} = \|\cdot\|_{H^l(\Omega)}$ for $l = 0, 1, k+1$.
Take the log of each side:

$$\underbrace{\log(E_1(h))}_{y} \leq \log(Ch^k \|u\|_{k+1,\Omega}) = k \underbrace{\log(h)}_{x} + \underbrace{\log(C\|u\|_{k+1,\Omega})}_{c}$$

Method of manufactured solutions

Recipe

1. Take a suitable function u
2. Compute $-\Delta u$ to obtain f
3. Compute boundary values (trivial if only Dirichlet boundary conditions are used)
4. Solve the corresponding variational problem

$$a(u_h, v) = L(v)$$

   for a sequence of meshes $\mathcal{T}_h$ and compute the error $E_i(h) = \|u - u_h\|_{i,\Omega_i}$ for $i = 0, 1$
5. Plot $\log(E_i(h))$ against $\log(h)$ and determine k

Homework

Try this by taking $u = \sin(2\pi x)\sin(2\pi y)$ on the unit square. Solve the problem for $N = 2, 4, 8, 16, 64, 128$ and compute both the L^2 and H^1 errors for $P1$, $P2$ and $P3$ elements as a function of h. Can you determine the convergence rate?