INSA Toulouse, Applied Mathematics department
Finite Element Methods

Programming Practicals with FEniCS

Jerome.Monnier@insa-toulouse.fr

Part 1. Steady-state models

1 Linear advection-diffusion equation

We consider the following stationary linear advection-diffusion equation :

$$-\mathrm{d}iv(\lambda\ \nabla u(x)) + w(x)\cdot\nabla u(x) \ = \ f(x) \text{ in } \Omega; \quad \Omega = [0, L_x] \times [0, L_y]. \tag{1}$$

Where : $u(x)$ is the unknown scalar field ; $w(x)$ is a given velocity field.
The diffusivity λ may depends on x. One has : $\lambda \in L^\infty(\Omega); \quad \lambda > 0$.

The variable u can represent e.g. a chemical specie concentration in the atmosphere, w the wind.
The velocity expression w is given.Its expression is upon your own choice.

The boundary of Ω is decomposed as follows : $\partial\Omega = \Gamma_{in} \cup \Gamma_{wall} \cup \Gamma_{out}$.
Then equation above is closed with the following B.C. :

$$u \text{ given on } \Gamma_{in};\ \partial_n u = 0 \text{ on } \Gamma_{wall};\ -\lambda\partial_n u = c\ u \text{ on } \Gamma_{out} \tag{2}$$

with c a given parameter, $c > 0$.

For each practical, you can upload on the course Moodle page a Python-FEniCS code.

1.1 Goal : to stabilize the advection term

1.1.1 Mathematical analysis

a) Write the weak formulation of the BVP.
b) Derive conditions such that this BVP is well-posed.

1.1.2 FE solution without stabilization

a) Implement the numerical model in FeniCS by using P_k-Lagrange FE. *(To do so, start from the Python-Fenics code available on the course Moodle page).*
b) Compute and plot out the local values of the dimensionless Peclet number Pe and $Pe_h = hPe$.
c) Highlight the instability phenomena if $Pe_h \geq 1$ and if no stabilisation term is introduced in the weak formulation.

1.1.3 Artificial diffusion & "sharp" test case

a) Implement the SUPG (and the SD) stabilisation methods presented in the course manuscript.
b) Build up a case where the solution u_h presents a sharp boundary layer. Compare the two solutions obtained by using SUPG and SD.

1.2 Goal-oriented mesh refinement

a) Consider (and implement) a particular model output.
b) Highlight the accuracy improvement when using a mesh refinement procedure related to this model ouput.
The latter will be based on the method already implemented in a Fenics code (see Moodle page).

2 Non-linear model

In this section, we consider the same stationary advection-diffusion as before but the diffusivity λ depends here on the solution u....
The equation reads :

$$-\mathrm{d}iv(\lambda(u)\ \nabla u(x)) + w(x) \cdot \nabla u(x) \ = \ f(x) \text{ in } \Omega; \quad \Omega = [0, L_x] \times [0, L_y]. \quad (3)$$

The equation is closed with the same BC as before, see (2).

The diffusivity λ is a differentiable function.
We have : $\lambda : u \in V \mapsto \lambda(u) \in L^\infty(\Omega), \quad \lambda(u) > 0$.

The diffusivity expression can be for example :

$$\lambda(u) = \lambda_0 u^m \text{ with } m \geq 1$$

.

2.1 Build a non linear solver

a) Write the (non linear) weak formulation of the BVP.

b) This non linear BVP is solved by the Newton-Raphson method.
Derive the equations to be solved at each iteration.

c) Detail the algorithm to implement.
Implement your home-developed non-linear solver based on this Newton-Raphson algorithm.
Recall. Start from the Python-Fenics code(s) available on the course Moodle page.

d) Implement the "black-box" non-linear solver proposed in FEniCS.

d) Propose a strategy to validate your computational code.

e) Compare the obtained solutions, including with the linear BVP one (solved by using your algorithm).
Compare the CPU-time between the different cases.

Part 2. Unsteady models

We consider the same non-linear BVP as before but in its unsteady version. The equation reads :

$$\partial_t u(x,t) - \mathrm{d}iv(\lambda(u(x,t))\ \nabla u(x,t)) + w(x) \cdot \nabla u(x,t) \ = \ f(x,t) \text{ in } \Omega \times [0,T]. \tag{4}$$

The equation is closed with the same BC as before, see (2), and an Initial Condition $u(x,0) = u_0(x)\ \forall x$.

3 Build a higher-order solver

3.1 Task 1 : Set up a meaningful physical BVP

Based on this model, imagine a physical problem you aim at simulating. You will detail all the data of your problem.

3.2 Task 2 : build a higher-order solver

a) Implement the model using a higher-order FE scheme : RKq in time and P_k-Lagrange in space.
You will synthetically write the equations which are coded and the global algorithm(s).
Explain briefly the pros and cons of the mass condensation technique in your context.

b) Explain synthetically how you can show the actual order of your computational code.

b) Starting from your I.C., show that you can recover the steady-state solution computed by the non-linear solver developed in the previous part. Detail your procedure (including criteria of convergence etc).