INSA Toulouse, Applied Mathematics department
Finite Element Methods

Programming Practicals with FEniCS

By Jerome.Monnier@insa-toulouse.fr
Other information and all files are available on the INSA Moodle page.

1 Introduction to FEniCS Library

a) Consult the short introduction and examples presented on-line, see URLs provided on the course Moodle page.
Upload a few basic FEniCS demo files

b) Perform different simulations based on :
— the Laplace - Poisson equation,
— the time-dependent diffusion equation (heat equation).

2 Non-linear model

In this section, we consider a steady-state advection-diffusion equation with the diffusivity parameter μ depending on the field u.

2.1 Model equations

The geometry is supposed to be bidimensional as $\Omega = [0, L_x] \times [0, L_y]$.
The function $u(x)$ denotes the unknown (scalar) field.
The considered equation reads :

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

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

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

with c a given parameter, $c \geq 0$.

The diffusivity parameter $\mu(u)$ is given. It is supposed to be C^1. Moreover : $\mu : u \in V \mapsto \mu(u) \in L^\infty(\Omega), \quad \mu(u) > 0$.

For example, the diffusivity expression may be set as :

$$\mu(u) = \mu_0 u^m \text{ with } m \geq 1$$

μ_0 given, strictly positive.

2.2 Build-up 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).

3 The advection-diffusion equation

3.1 Equations

The considered steady-state linear advection-diffusion equation reads :

$$-\mathrm{d}iv(\mu\ \nabla u(x)) + w(x) \cdot \nabla u(x) \ = \ f(x) \text{ in } \Omega \tag{3}$$

The geometry is the same as previously.
$u(x)$ is the unknown (scalar) field, $w(x)$ is a given velocity field.

The diffusivity coefficient μ is either given or depending the unknown field $u(x)$.
In the first case, the model is linear, and we assume that $\mu \in L^\infty(\Omega)$, $\mu > 0$ a.e.
In the second case, the model is non linear, and we assume like previously that : $\mu(u)$ is differentiable, $\mu : u \in V \mapsto \mu(u) \in L^\infty(\Omega), \quad \mu(u) > 0$.

The elliptic PDE above is closed with the following B.C. :

$$u \text{ given on } \Gamma_{in};\ \partial_n u = 0 \text{ on } \Gamma_{wall};\ -\mu\partial_n u = c\ u \text{ on } \Gamma_{out} \tag{4}$$

with c a given parameter, $c \geq 0$.

3.2 Mathematical analysis

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

3.3 Numerical results

a) Implement the numerical model in FeniCS by using P_k-Lagrange FE.
To do so, you will 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 $Pe = (\frac{L^* W^*}{\mu})$.
Comment the obtained Peclet map.
c) Comment the simulated phenomena.

3.4 To go further

This is a "to go further section". This section may be addressed after all other tasks have been finished.

3.4.1 FE solution with vs 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.

3.4.2 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.

3.4.3 Goal-oriented mesh refinement

This is a "to go further section".
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).

4 Time-dependent model

In this section, you will solve the same advection-diffusion model as previously but in its unsteady version.
Moreover you are invited to consider a meaningful phenomena.
A few ideas : chemical specie or a virus in atmosphere or in a biological fluid, a temperature field, etc

4.1 Model equations

The equation reads :

$$\partial_t u(x,t) - \mathrm{d}iv(\mu\ \nabla u(x,t)) + w(x) \cdot \nabla u(x,t) \;=\; f(x,t) \text{ in } \Omega \times [0,T]. \quad (5)$$

The equation is closed with the same BC as before, see (4), plus an Initial Condition (IC) : $u(x,0) = u_0(x)\ \forall x \in \Omega$.

As previously the diffusivity coefficient μ is either given or depending on the unknown field $u(x)$.
In the first case, the model is linear. We assume that $\mu \in L^\infty(\Omega)$, $\mu > 0$ a.e.
In the second case, the model is non linear. We assume like previously that :
$\mu(u)$ is differentiable, $\mu : u \in V \mapsto \mu(u) \in L^\infty(\Omega), \quad \mu(u) > 0$.

In the sequel, the source term $f(x,t)$ may be defined time-independent.

4.2 Build-up a "higher-order" solver

a) Implement the model using a 2nd order scheme : Crank-Nicholson scheme in time and P_2-Lagrange in space.
You will first detail the equations and the algorithm(s) to be coded.

b) Starting from your I.C., show that you recover a steady-state solution computed at the previous stage (steady-state section).
Explain briefly the choice of your time-step Δt, your criteria of convergence (to reach a stationnary solution) etc.

c) Briefly explain how you could, (should !), validate your computational code and demonstrate its actual order.
Bonus : Detail the expression of an exact solution enabling to demonstrate

the actual order of your computational code.

d) To go further : same questions as above but by considering the RK-2 time scheme.
RK2 scheme is also called the mid-point scheme or Heun's method.