Introduction à la modélisation et à l’analyse numérique des EDP

Exercise sessions
Instructor: Pascal Frey*

The purpose of this exercise session is to give an overview of several aspects of scientific computing through the resolution of a few ‘toy problems’. First, we focus on the solving of an elliptic boundary value problem with the Finite Element Method. In a first part, we focus on the implementation of a very basic PDE model, in order to get the reader to be familiar with FreeFem++ software, which makes it very easy to handle finite element spaces and mesh modifications, within a very few lines of black-box code. Then, we turn to more realistic PDE models from fluid dynamics.

Session 1. Getting started with FreeFem++.

Throughout this section, Ω ⊂ \mathbb{R}^d represents a bounded domain with Lipschitz boundary, \( f \in L^2(\Omega) \) is a source term, and \( u_0 \in H^{\frac{1}{2}}(\partial\Omega) \) is a boundary prescription. The aim is to find the solution \( u \) solving the following so-called Poisson problem:

\[
\begin{aligned}
-\Delta u &= f \quad \text{in } \Omega \\
u &= u_0 \quad \text{on } \partial\Omega
\end{aligned}
\]

(1)

Recall that, from the classical Lax-Milgram theory, this problem admits a unique solution \( u \in H^1_0(\Omega) \), which depends continuously on the data \( f \).

This problem arises, for instance, in electrostatics, as a link between a free charge density \( f \) (up to some constants), and the resulting electric potential field \( u \). Problem (1) enjoys another common physical interpretation, casting \( u \) as the steady state of a heat transfer with source term \( f \).

This first part is devoted to some numerical considerations over problem (1), using the software FreeFem++ [1], a powerful tool for solving boundary value problems with the Finite Element Method. In particular, FreeFem++ allows testing a whole lot of strategies to solve (1) within very few lines of code.

For the sake of numerical issues, we consider the square \( (0,1) \times (0,1) \) as domain \( \Omega \), and we choose \( f = \sin(2\pi x)\sin(2\pi y) \) and \( u_0 = 0 \); respectively.

* UPMC Univ Paris 06, UMR 7598, Laboratoire J.L. Lions, F-75005 Paris, France.
The instructor acknowledge the valuable help of Yannick Privat (LJLL, UPMC) and Charles Dapogny (LJK, UJF, Grenoble) in preparing these class notes.
**Question 1:** Define in FreeFem++ the domain Ω, as well as function $f$, using listing 1

```
/* Border of the computational domain */
border left(t=0.0,1.0){x = 0.0; y = 1.0 -t; label = 1;};
border bottom(t=0.0,1.0){x = t; y = 0.0; label = 1;};
border right(t=0.0,1.0){x = 1.0; y = t; label = 1;};
border top(t=0.0,1.0){x = 1.0-t; y = 1.0; label = 1;};

/* Create, then display initial mesh, with 10 nodes on each part of the boundary */
mesh Th = buildmesh( left(10) + bottom(10) + right(10) + top(10) ) ;
plot(Th, wait=1, ps="initial Mesh");

/* Source term for Poisson equation */
func f = sin(2.0*pi*x)*sin(2.0*pi*y);
```

Listing 1. Definition of the data with FreeFem++

**Question 2:**

- Write the variational formulation associated to problem (1).
- In FreeFem++, introduce a finite element space associated to the mesh defined at question 1. The definition of a finite element space $V_h$, associated to a mesh $T_h$, is triggered by the command

  \[ \text{fespace Vh(Th,typeelt);} \]

  where typeelt is the type of the desired finite element; e.g. typeelt = P1, P2,... for Lagrange-type finite elements, but many other choices are possible.

  A typical finite element problem - under variational form - is defined in FreeFem++ through the command

  \[ \text{nameProblem(uh,vh, solver = CG, eps = 1.0e-6)} \]

  \[ \text{solution and unknown } \in V_h \text{ type of solver precision parameter} \]

  \[ \text{int2d(Th)(dx(uh)*dx(vh) ...))} \]

  \[ \text{Left-hand side of variational formula} \]

  \[ \text{int2d(Th)( ...))} \]

  \[ \text{Right-hand side of variational formula} \]

  \[ + \text{on(1,uh=0.0)} \]

  \[ \text{boundary conditions on part labelled 1} \]

  Such a problem is then solved by simply calling its name as a command:

  \[ \text{nameProblem;} \]

  - What do you observe? Does the problem seem properly solved?
**Question 3**: For increased accuracy in the computation, the most intuitive idea would be to refine the computational mesh $T_h$. Try solving the same problem, with the same data function $f$, but with different mesh sizes. Does this work out? Run similar tests with data functions $f$ showing sharper variations, e.g. functions of the form

$$f(x, y) = \sin(m\pi x) \sin(n\pi y), \quad m, n \in \mathbb{N}.$$  

**Question 4**: Actually, there is no realistic way to guess *a priori* what would be the ideal size of a mesh $T_h$ that would enable a proper resolution of a problem such as (1), from the sole knowledge of the data $f$. Hence the need for *mesh adaptation*.

The basic principle of *mesh adaptation* is to start from a coarse mesh, on which the problem at stake is rather poorly solved (but at a very low CPU cost), and to iteratively infer from increasingly refined meshes and accurate solutions (at still low CPU cost) a ‘good’ mesh, adapted to the solution of the particular problem under consideration (that is, the nodes are concentrated only in the areas where enhanced accuracy is required).

![Figure 1](image)

*Figure 1.* (a) Initial mesh (143 points), (b) the associated solution (the amplitude is altogether wrong, although it is not displayed on the figure), (c) adapted mesh, after 5 iterations ($\approx 1000$ points), (d) the associated solution.
In FreeFem++, there exists a (rather sophisticated) command

\[ \text{Th} = \text{adaptmesh}(\text{Th}, \text{uh}, \text{tol}); \]

which proceeds to the adaptation of a mesh \( T_h \), with respect to the variations of a function \( u_h \) defined on it, up to a tolerance \( \text{tol} \).

Write a loop, which proceeds to the adaptation of the coarse initial mesh of question 1 into a mesh amenable to the resolution of problem (1). Usually, the process converges in 4-5 iterations.

**Session 2. The heat equation.**

We now turn to the resolution of the heat equation, which is nothing but the time-dependent version of the previous model. Let \( \Omega \subset \mathbb{R}^2 \) a bounded domain with Lipschitz boundary, filled with a material of thermal conductivity \( \alpha > 0 \), and heated with a source \( f \in L^2(\Omega) \). The temperature distribution at the initial time is described by a function \( u_0 \in H^1(\Omega) \). \( \Omega \) is surrounded by another material with fixed temperature, which is set to 0 for simplicity, so that temperature 0 is imposed at the boundary \( \partial \Omega \) at any time.

The temperature at time \( t \) (\( 0 \leq t \leq T \)) in \( \Omega \) is then solution to the following equation:

\[
\begin{align*}
\frac{\partial u}{\partial t}(t, x) - \alpha \Delta u(t, x) &= f(x) \quad \text{for } (t, x) \in (0, T) \times \Omega \\
u(t = 0, x) &= u_0(x) \quad \text{for } x \in \Omega \\
u(t, x) &= 0 \quad \text{for } t \in (0, T), x \in \partial \Omega.
\end{align*}
\]

**Question 1:** Write down the variational formulation for (2), at the continuous level in space and time (i.e. the time derivative in (2) is not yet discretized). This is achieved by:

1. identifying the functional space \( V \) to which each function \( u(t, .) \) should belong to,
2. multiplying the first equality in (2) by a (time-independent) function \( v \in V \),
3. performing integration by parts where it is needed to end with formulae as close as possible to those of section I..

**Question 2:** The sequence \( t \mapsto u(t, .) \) is approximated by a series of functions \( u^n(x) \approx u(t^n, x) \in V \), where \( 0 < t^0 < \ldots < t^N = T \) is a subdivision of the time interval \( (0, T) \), with time step \( \Delta t: t^n = n \Delta t, n = 0, \ldots, N \). Choose a finite difference discretization for \( \frac{\partial u}{\partial t}(t^n, .) \) in terms of \( u^{n-1}, u^n, u^{n+1} \). Note that several choices may be possible.

**Question 3:** Derive the discrete-in-time, space-continuous variational formulation for each function \( u^n \) (involving \( u^{n-1}, \ldots \)).
**Question 4:** In this example (see 2), the domain $\Omega$ is the rectangle $(0, 2) \times (0, 1)$, and the final time is $T = 5$. The thermal conductivity of the considered material is $\alpha = 0.1$, the source $f$ is concentrated in a small ball around point $p_1 = (0.5, 0.75)$ of the domain:

$$f(x) = \begin{cases} 1 & \text{if } ||x - p_1|| < 0.12 \\ 0 & \text{otherwise} \end{cases},$$

and the initial distribution of temperature $u_0$ inside $\Omega$ is given by:

$$u_0(x) = \begin{cases} 100(0.2 - ||x - p_2||) & \text{if } ||x - p_2|| < 0.2 \\ 0 & \text{otherwise} \end{cases},$$

where $p_2 = (1.5, 0.25)$.

![Figure 2. Initial and boundary conditions for the proposed test case for the numerical resolution of the heat equation (2).](image)

Implement the variational formulation of question 3. into **FreeFem++**, and run several tests:
- using different kinds of finite elements for the approximation of functions $u^n$, $n = 0, ..., N$ (e.g. $P^1$, $P^2$ Lagrange finite elements).
- using various finite difference approximations of the time-derivative $\frac{\partial u}{\partial t}$ (see question 2.).
- using different values for the time $\Delta t$.
- acting on the mesh of the domain (for instance, try to use the mesh adaptation technique presented in Section I at some iterations of the process).

What do you observe? What is the influence of these different ways of tackling the problem?
In this section, we are interested in the stationary Stokes equations, posed in a domain \( \Omega \subset \mathbb{R}^2 \) with Lipschitz boundary: a fluid with dynamic viscosity \( \nu \) occupying \( \Omega \) is described by its velocity \( u : \Omega \to \mathbb{R}^2 \), and pressure field \( p : \Omega \to \mathbb{R}^2 \), which satisfy the system:

\[
\begin{align*}
-\nu \Delta u(x) + \nabla p(x) &= f(x) \quad \text{for } x \in \Omega \\
\text{div} \, u(x) &= 0 \quad \text{for } x \in \Omega \\
u(x) &= u_d \quad \text{for } x \in \partial \Omega,
\end{align*}
\]

where \( u_d \) is a prescribed velocity field on \( \partial \Omega \), and \( f \) is a source term.

**Question 1:** Write down the variational formulation associated to equation (3).

Consider, in FreeFem++, the classical *lid driven cavity* for Stokes problem: \( \Omega \) is the unit square \((0,1) \times (0,1)\), filled with a fluid of unit dynamic viscosity: \( \nu = 1 \). The imposed velocity field \( u_d \) on \( \partial \Omega \) is:

\[
\forall x \in \partial \Omega, \quad u_d(x) = \begin{cases} 
(1,0) & \text{if } x \text{ lies on the upper part of } \partial \Omega \\
(0,0) & \text{otherwise}
\end{cases}
\]

and the source term is set to \( f = 0 \).

We are interested in a particular form for the meshes of \( \Omega \), that of a Cartesian grid split into triangles. In FreeFem++, such a mesh \( Th \) is generated using the following command:

\[
Th = \text{square}(50, 50);
\]

Using this command, the labels of the boundary \( \partial \Omega \) are automatically set as depicted in figure 3.

**Figure 3.** A particular mesh of the unit square, and the boundary labels generated by the FreeFem++ command square.
**Question 2:** Implement the variational formulation of question 1. in **FreeFem++**, using $\mathbb{P}^2$ Lagrange finite elements for discretizing the velocity $u$, and $\mathbb{P}^1$ elements for the pressure.

*Hint: use the penalization method (seen in the theoretical part of the course) to bring back the variational formulation of question 1. to a ‘classical’ variational formulation over the couple $(u,p)$.*

**Question 3:** Try out other pairs of finite elements for discretizing $(u,p)$, and notably the pairs $\mathbb{P}^1 \times \mathbb{P}^0$, and bubble-$\mathbb{P}^1 \times \mathbb{P}^0$. Does the method always produce the ‘correct’ solution to the system? Why is that so, according to you?

---

**Session 4. The unsteady Stokes equations.**

We are now getting closer and closer to realistic models; let us now consider the transient model associated to Stokes equations (3). A fluid, whose density is set to $\rho=1$ for simplicity, with dynamic viscosity $\nu$, is occupying a domain $\Omega$, and is described by means of its (time-dependent) velocity field $u(t,x)$ and pressure field $p(t,x)$ all along a period of time $0,T$.

The initial state of the fluid is known, that is, $u(t=0,.) = u_0$, and $p(t=0,.) = p_0$ are given. Mixed boundary conditions are considered, that is, $\partial \Omega$ is decomposed into two complimentary parts $\Gamma_D$ and $\Gamma_N$. The velocity $u(t,.) = u_d$ is prescribed at every time on $\Gamma_D$, and an external stress $g$ is exerted on $\Gamma_N$, so that the system is driven by the system of equations:

\[
\begin{align*}
\frac{\partial u}{\partial t}(t,x) - \nu \Delta u(t,x) + \nabla p(t,x) &= f(t,x) \quad \text{for} \quad (t,x) \in (0,T) \times \Omega \\
\text{div } u(t,x) &= 0 \quad \text{for} \quad (t,x) \in (0,T) \times \Omega \\
u \frac{\partial u}{\partial n}(t,x) - p(t,x) \cdot n(x) &= g(t,x) \quad \text{for} \quad t \in (0,T) \text{ and } x \in \Gamma_D \\
u u(0,x) &= u_0(x) \quad \text{for} \quad x \in \Omega
\end{align*}
\]

**Question 1:** Write down the variational formulation associated to system (4).

We now consider the benchmark **obstacle test case**, described in figure 4: on the left (blue) part of the boundary, a parabolic profile is imposed for the velocity: $u_d(x) = 4y(1-y)$, whereas on the remainder of the Dirichlet boundary (in black), homogeneous Dirichlet boundary conditions are imposed: $u_d(x) = 0$. Eventually, the right, red part corresponds to the Neumann part $\Gamma_N$ of the boundary, and no stress is applied: $g = 0$. Similarly, no body forces are applied: $f = 0$.

The other parameters for the computation are: $\nu = 0.005$, $T = 40$ and

\[
\forall x \in \Omega, \quad u_0(x) = \begin{cases}
(4y(1-y),0) & \text{if } x \text{ lies on blue part of } \partial \Omega \\
(0,0) & \text{otherwise}
\end{cases}
\]

**Question 2:** Implement the previous variational formulation in **FreeFem++**. Once again, try different combinations of finite element spaces for the velocity and pressure fields, as well as different discretizations of the time derivative $\frac{\partial u}{\partial t}$. You may also try using the mesh adaptation function described in section 1 to reach a higher quality result.
SESSION 5. THE NAVIER-STOKES EQUATIONS.

We now come to the central model in this course, namely that of Navier-Stokes equations, which is a more realistic description of the motion of an incompressible fluid than that of the mere Stokes equations. Using the exact same notations as in the previous section, they characterize the couple velocity-pressure \((u, p)\) of an incompressible fluid as the solution to:

\[
\begin{cases}
\frac{\partial u}{\partial t}(t,x) + u \cdot \nabla u(t,x) - \nu \Delta u(t,x) + \nabla p(t,x) = f(t,x) & \text{for } (t,x) \in (0,T) \times \Omega \\
\text{div } u(t,x) = 0 & \text{for } (t,x) \in (0,T) \times \Omega \\
\nu \frac{\partial u}{\partial n}(t,x) - p(t,x) \cdot n(x) = g(t,x) & \text{for } t \in (0,T) \text{ and } x \in \Gamma_N \\
u_0(x) = u_0(x) & \text{for } x \in \Omega
\end{cases}
\]

The main difference between Stokes and Navier-Stokes equations lies in the nonlinear convective term \(u \cdot \nabla u\), induced by the acceleration of the fluid. This term can be neglected when the velocity of the fluid is low when compared to the viscous forces, but may become dominant in the high-velocity regime, in which the Stokes approximation becomes dramatically rough. This term is also the main cause of the difficulties in the theoretical and numerical treatments of Navier-Stokes equations.

In this section, we propose to solve equation (5) using the method of characteristics for dealing with the (difficult) convective term.

If \(x_0 \in \Omega\), and \(t_0 \in T\), denote as \(t \mapsto X(t,t_0,x_0)\) the characteristic curve passing at \(x_0\) at time \(t = t_0\), solution to the ordinary differential equation:

\[
\begin{cases}
\forall t \in (0,T), \quad \dot{X}(t,t_0,x_0) = u(t,X(t,t_0,x_0)) \\
X(t_0,t_0,x_0) = x_0
\end{cases}
\]

which is nothing but the trajectory of a fluid particle located at \(x_0\) at time \(t_0\). Then, one can see that:

\[
\frac{d}{dt} (u(t,X(t,t_0,x_0))) = \left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right)(t,X(t,t_0,x_0)),
\]

meaning that the convective term is the time derivative of \(u\) along the trajectories of the fluid particles, in other terms the dynamic acceleration of the fluid. According to this remark, the following finite difference approximation can be proposed for the convective term (with
obvious notations):
\[
\left( \frac{\partial u}{\partial t} + u \cdot \nabla u \right)(t^n, x) \approx \frac{u^n(x) - u^{n-1}(X(t^{n-1}, t^n, x))}{\Delta t}
\]

Now, in the course of an iterative computation, if \( u^{n-1} \) is available, one only has to evaluate the term \( u^{n-1}(X(t^{n-1}, t^n, x)) \). This can be achieved in FreeFem++ using the \texttt{convect} command (see listing 2).

```
Vh ux,uy; // components of the velocity field
real dt = 0.05; // time step
Vh f,g; // f= scalar quantity to be convected, g = resulting quantity
g = convect ([ux,uy], -dt, f); // sign – indicates backtracking of the
// characteristic curves
```

**Listing 2.** Backtracking a characteristic curve in FreeFem++

Consider the \textit{step} test case, depicted in figure 5.

```
0
1
0.5
20
```

**Figure 5.** The step test case for the Navier-Stokes equations.

Here, \( T = 40, \nu = 0.005 \), and the boundary conditions are exactly those of the previous example, and the initial velocity field is given by:

\[
\forall x \in \Omega, \quad u_0(x) = \begin{cases} 
(4y(1-y), 0) & \text{if } x < 2 \\
(0,0) & \text{otherwise}
\end{cases}
\]

**Question:** Implement the Navier-Stokes model in FreeFem++, relying on the method of characteristics for the discretization of the convective term.

**Remark** Numerous different techniques exist for the solution of the Navier-Stokes equations, among which Chorin-Temam’s projection method, and Newton’s method (seen during the theoretical part of the course). At this point, you should have all the required knowledge about FreeFem++ for programming them!

**References**
