3.2 Heat Exchanger

Summary *Here we shall learn more about geometry input and triangulation files, as well as read and write operations.*

The problem Let $\{C_i\}_{1,2}$, be 2 thermal conductors within an enclosure C_0. The first one is held at a constant temperature u_1 the other one has a given thermal conductivity κ_2 5 times larger than the one of C_0. We assume that the border of enclosure C_0 is held at temperature $20°C$ and that we have waited long enough for thermal equilibrium.
In order to know $u(x)$ at any point x of the domain Ω, we must solve

$$\nabla \cdot (\kappa \nabla u) = 0 \quad \text{in} \quad \Omega, \quad u_{|\Gamma} = g$$

where Ω is the interior of C_0 minus the conductors C_1 and Γ is the boundary of Ω, that is $C_0 \cup C_1$ Here g is any function of x equal to u_i on C_i. The second equation is a reduced form for:

$$u = u_i \text{ on } \quad C_i, \quad i = 0, 1.$$

The variational formulation for this problem is in the subspace $H_0^1(\Omega) \subset H^1(\Omega)$ of functions which have zero traces on Γ.

$$u - g \in H_0^1(\Omega) \ : \ \int_\Omega \nabla u \nabla v = 0 \quad \forall v \in H_0^1(\Omega)$$

Let us assume that C_0 is a circle of radius 5 centered at the origin, C_i are rectangles, C_1 being at the constant temperature $u_1 = 60°C$.

Example 3.3 (heatex.edp)

```
//    file heatex.edp
int C1=99, C2=98;              //    could be anything such that ≠0 and C1≠C2
border C0(t=0,2*pi){x=5*cos(t); y=5*sin(t);}

border C11(t=0,1){ x=1+t;  y=3;      label=C1;}
border C12(t=0,1){ x=2;    y=3-6*t;  label=C1;}
border C13(t=0,1){ x=2-t;  y=-3;     label=C1;}
border C14(t=0,1){ x=1;    y=-3+6*t; label=C1;}

border C21(t=0,1){ x=-2+t; y=3;      label=C2;}
border C22(t=0,1){ x=-1;   y=3-6*t;  label=C2;}
border C23(t=0,1){ x=-1-t; y=-3;     label=C2;}
border C24(t=0,1){ x=-2;   y=-3+6*t; label=C2;}

plot(    C0(50)                          //    to see the border of the domain
        + C11(5)+C12(20)+C13(5)+C14(20)
        + C21(-5)+C22(-20)+C23(-5)+C24(-20),
        wait=true, ps="heatexb.eps");

mesh Th=buildmesh(    C0(50)
                    + C11(5)+C12(20)+C13(5)+C14(20)
                    + C21(-5)+C22(-20)+C23(-5)+C24(-20));
plot(Th,wait=1);
```

```
fespace Vh(Th,P1); Vh u,v;
Vh kappa=1+2*(x<-1)*(x>-2)*(y<3)*(y>-3);
solve a(u,v)= int2d(Th)(kappa*(dx(u)*dx(v)+dy(u)*dy(v)))
                +on(C0,u=20)+on(C1,u=60);
plot(u,wait=true, value=true, fill=true, ps="heatex.eps");
```

Note the following:

- `C0` is oriented counterclockwise by t, while `C1` is oriented clockwise and `C2` is oriented counterclockwise. This is why `C1` is viewed as a hole by `buildmesh`.
- `C1` and `C2` are built by joining pieces of straight lines. To group them in the same logical unit to input the boundary conditions in a readable way we assigned a label on the boundaries. As said earlier, borders have an internal number corresponding to their order in the program (check it by adding a `cout<<C22;` above). This is essential to understand how a mesh can be output to a file and re-read (see below).
- As usual the mesh density is controlled by the number of vertices assigned to each boundary. It is not possible to change the (uniform) distribution of vertices but a piece of boundary can always be cut in two or more parts, for instance `C12` could be replaced by `C121+C122`:

```
                                //    border C12(t=0,1) x=2; y=3-6*t; label=C1;
border C121(t=0,0.7){ x=2;    y=3-6*t;  label=C1;}
border C122(t=0.7,1){ x=2;    y=3-6*t;  label=C1;}
... buildmesh(.../* C12(20) */ + C121(12)+C122(8)+...);
```

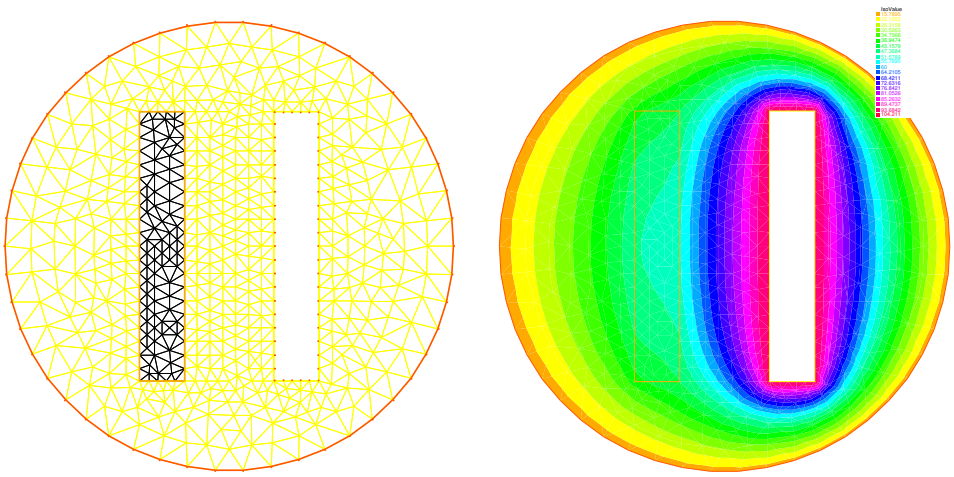

Figure 3.2: The heat exchanger

Exercise Use the symmetry of the problem with respect to the axes; triangulate only one half of the domain, and set Dirichlet conditions on the vertical axis, and Neumann conditions on the horizontal axis.

Writing and reading triangulation files Suppose that at the end of the previous program we added the line

```
savemesh(Th,"condensor.msh");
```

and then later on we write a similar program but we wish to read the mesh from that file. Then this is how the condenser should be computed:

```
mesh Sh=readmesh("condensor.msh");
fespace Wh(Sh,P1); Wh us,vs;
solve b(us,vs)= int2d(Sh)(dx(us)*dx(vs)+dy(us)*dy(vs))
                +on(1,us=0)+on(99,us=1)+on(98,us=-1);
plot(us);
```

Note that the names of the boundaries are lost but either their internal number (in the case of `C0`) or their label number (for `C1` and `C2`) are kept.

3.3 Acoustics

Summary *Here we go to grip with ill posed problems and eigenvalue problems*
Pressure variations in air at rest are governed by the wave equation:

$$\frac{\partial^2 u}{\partial t^2} - c^2 \Delta u = 0.$$

When the solution wave is monochromatic (and that depend on the boundary and initial conditions), u is of the form $u(x,t) = Re(v(x)e^{ikt})$ where v is a solution of Helmholtz's equation:

$$\begin{aligned} &k^2 v + c^2 \Delta v = 0 \quad \text{in} \quad \Omega, \\ &\frac{\partial v}{\partial n}|_\Gamma = g. \end{aligned} \tag{3.1}$$

where g is the source. Note the "+" sign in front of the Laplace operator and that $k > 0$ is real. This sign may make the problem ill posed for some values of $\frac{c}{k}$, a phenomenon called "resonance".
At resonance there are non-zero solutions even when $g = 0$. So the following program may or may not work:

Example 3.4 (sound.edp)

```
//    file sound.edp
real kc2=1;
func g=y*(1-y);

border a0(t=0,1) { x= 5; y= 1+2*t ;}
border a1(t=0,1) { x=5-2*t; y= 3 ;}
border a2(t=0,1) { x= 3-2*t; y=3-2*t ;}
border a3(t=0,1) { x= 1-t; y= 1 ;}
border a4(t=0,1) { x= 0; y= 1-t ;}
border a5(t=0,1) { x= t; y= 0  ;}
border a6(t=0,1) { x= 1+4*t; y= t ;}
```

```
mesh Th=buildmesh( a0(20) + a1(20) + a2(20)
        + a3(20) + a4(20) + a5(20) + a6(20));
fespace Vh(Th,P1);
Vh u,v;

solve sound(u,v)=int2d(Th)(u*v * kc2 - dx(u)*dx(v) - dy(u)*dy(v))
                  - int1d(Th,a4)(g*v);
plot(u, wait=1, ps="sound.eps");
```

Results are on Figure 3.3. But when $kc2$ is an eigenvalue of the problem, then the solution is not unique: if $u_e \neq 0$ is an eigen state, then for any given solution $u + u_e$ is another a solution. To find all the u_e one can do the following

```
real sigma = 20;                                    //    value of the shift
                                  //    OP = A - sigma B ; // the shifted matrix
varf  op(u1,u2)= int2d(Th)(  dx(u1)*dx(u2) + dy(u1)*dy(u2) - sigma* u1*u2 );
varf b([u1],[u2]) = int2d(Th)( u1*u2 ) ;    //   no Boundary condition see note
9.1

matrix OP= op(Vh,Vh,solver=Crout,factorize=1);
matrix B= b(Vh,Vh,solver=CG,eps=1e-20);

int nev=2;                        //    number of requested eigenvalues near sigma

real[int] ev(nev);                                 //    to store the nev eigenvalue
Vh[int] eV(nev);                                  //    to store the nev eigenvector

int k=EigenValue(OP,B,sym=true,sigma=sigma,value=ev,vector=eV,
                    tol=1e-10,maxit=0,ncv=0);
cout<<ev(0)<<" 2 eigen values "<<ev(1)<<endl;
v=eV[0];
plot(v,wait=1,ps="eigen.eps");
```

3.4 Thermal Conduction

Summary *Here we shall learn how to deal with a time dependent parabolic problem. We shall also show how to treat an axisymmetric problem and show also how to deal with a nonlinear problem.*

How air cools a plate We seek the temperature distribution in a plate $(0, Lx) \times (0, Ly) \times (0, Lz)$ of rectangular cross section $\Omega = (0, 6) \times (0, 1)$; the plate is surrounded by air at temperature u_e and initially at temperature $u = u_0 + \frac{x}{L}u_1$. In the plane perpendicular to the plate at $z = Lz/2$, the temperature varies little with the coordinate z; as a first approximation the problem is 2D.

We must solve the temperature equation in Ω in a time interval (0,T).

$$\begin{aligned} &\partial_t u - \nabla \cdot (\kappa \nabla u) = 0 \text{ in } \Omega \times (0, T), \\ &u(x, y, 0) = u_0 + x u_1 \\ &\kappa \frac{\partial u}{\partial n} + \alpha(u - u_e) = 0 \text{ on } \Gamma \times (0, T). \end{aligned} \tag{3.2}$$

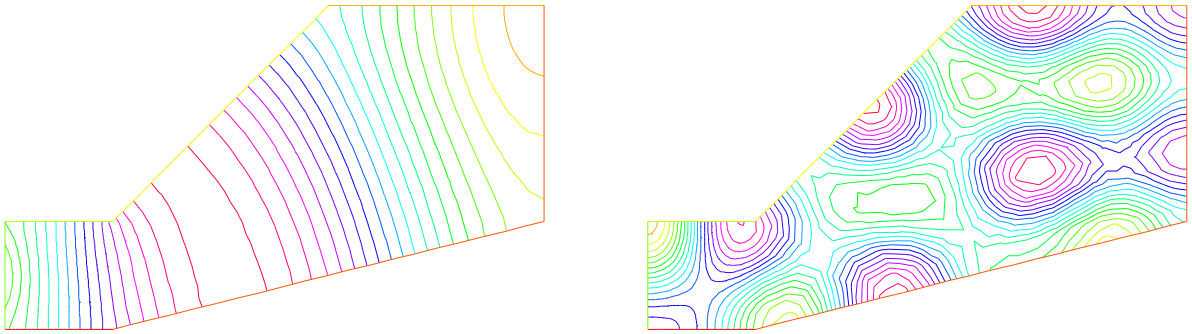

Figure 3.3: Left:Amplitude of an acoustic signal coming from the left vertical wall. Right: first eigen state ($\lambda = (k/c)^2 = 19.4256$) close to 20 of eigenvalue problem $:-\Delta\varphi = \lambda\varphi$ and $\frac{\partial\varphi}{\partial n} = 0$ on Γ

Here the diffusion κ will take two values, one below the middle horizontal line and ten times less above, so as to simulate a thermostat. The term $\alpha(u - u_e)$ accounts for the loss of temperature by convection in air. Mathematically this boundary condition is of Fourier (or Robin, or mixed) type.
The variational formulation is in $L^2(0,T;H^1(\Omega))$; in loose terms and after applying an implicit Euler finite difference approximation in time; we shall seek $u^n(x,y)$ satisfying for all $w \in H^1(\Omega)$:

$$\int_\Omega (\frac{u^n - u^{n-1}}{\delta t} w + \kappa \nabla u^n \nabla w) + \int_\Gamma \alpha(u^n - u_u e) w = 0$$

```
func u0 =10+90*x/6;
func k = 1.8*(y<0.5)+0.2;
real ue = 25, alpha=0.25, T=5, dt=0.1 ;

mesh Th=square(30,5,[6*x,y]);
fespace Vh(Th,P1);
Vh u=u0,v,uold;

problem thermic(u,v)= int2d(Th)(u*v/dt + k*(dx(u) * dx(v) + dy(u) * dy(v)))
                + int1d(Th,1,3)(alpha*u*v)
                - int1d(Th,1,3)(alpha*ue*v)
                - int2d(Th)(uold*v/dt) + on(2,4,u=u0);
ofstream ff("thermic.dat");
for(real t=0;t<T;t+=dt){
    uold=u;     //    uold ≡ u^{n-1} = u^n ≡u
    thermic;  //    here solve the thermic problem
    ff<<u(3,0.5)<<endl;
```

```
    plot(u);
}
```

Notice that we must separate by hand the bilinear part from the linear one.
Notice also that the way we store the temperature at point (3,0.5) for all times in file `thermic.dat`. Should a one dimensional plot be required, the same procedure can be used. For instance to print $x \mapsto \frac{\partial u}{\partial y}(x, 0.9)$ one would do

```
for(int i=0;i<20;i++) cout<<dy(u)(6.0*i/20.0,0.9)<<endl;
```

Results are shown on Figure 3.4.

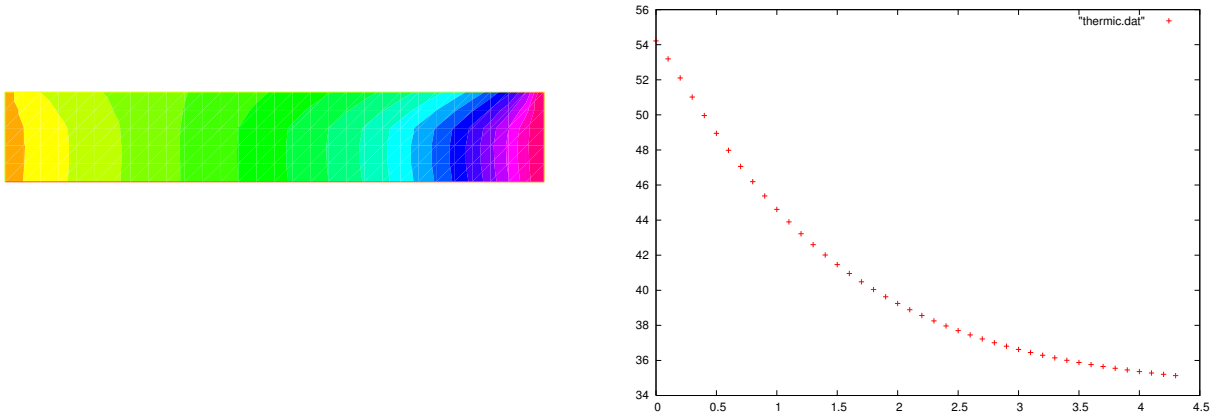

Figure 3.4: Temperature at T=4.9. Right: decay of temperature versus time at x=3, y=0.5

3.4.1 Axisymmetry: 3D Rod with circular section

Let us now deal with a cylindrical rod instead of a flat plate. For simplicity we take $\kappa = 1$. In cylindrical coordinates, the Laplace operator becomes (r is the distance to the axis, z is the distance along the axis, θ polar angle in a fixed plane perpendicular to the axis):

$$\Delta u = \frac{1}{r}\partial_r(r\partial_r u) + \frac{1}{r^2}\partial^2_{\theta\theta} u + \partial^2_{zz}.$$

Symmetry implies that we loose the dependence with respect to θ; so the domain Ω is again a rectangle $]0, R[\times]0, |[$. We take the convention of numbering of the edges as in `square()` (1 for the bottom horizontal ...); the problem is now:

$$\begin{aligned} &r\partial_t u - \partial_r(r\partial_r u) - \partial_z(r\partial_z u) = 0 \text{ in } \Omega, \\ &u(t=0) = u_0 + \frac{z}{L_z}(u_1 - u) \\ &u|_{\Gamma_4} = u_0, \quad u|_{\Gamma_2} = u_1, \quad \alpha(u - u_e) + \frac{\partial u}{\partial n}|_{\Gamma_1 \cup \Gamma_3} = 0. \end{aligned} \tag{3.3}$$

Note that the PDE has been multiplied by r.
After discretization in time with an implicit scheme, with time steps `dt`, in the `FreeFem++` syntax r becomes x and z becomes y and the problem is:

```
problem thermaxi(u,v)=int2d(Th)((u*v/dt + dx(u)*dx(v) + dy(u)*dy(v))*x)
                + int1d(Th,3)(alpha*x*u*v) - int1d(Th,3)(alpha*x*ue*v)
                - int2d(Th)(uold*v*x/dt) + on(2,4,u=u0);
```

Notice that the bilinear form degenerates at $x = 0$. Still one can prove existence and uniqueness for u and because of this degeneracy no boundary conditions need to be imposed on Γ_1.

3.4.2 A Nonlinear Problem : Radiation

Heat loss through radiation is a loss proportional to the absolute temperature to the fourth power (Stefan's Law). This adds to the loss by convection and gives the following boundary condition:

$$\kappa \frac{\partial u}{\partial n} + \alpha(u - u_e) + c[(u + 273)^4 - (u_e + 273)^4] = 0$$

The problem is nonlinear, and must be solved iteratively. If m denotes the iteration index, a semi-linearization of the radiation condition gives

$$\frac{\partial u^{m+1}}{\partial n} + \alpha(u^{m+1} - u_e) + c(u^{m+1} - u_e)(u^m + u_e + 546)((u^m + 273)^2 + (u_e + 273)^2) = 0,$$

because we have the identity $a^4 - b^4 = (a - b)(a + b)(a^2 + b^2)$. The iterative process will work with $v = u - u_e$.

```
...
fespace Vh(Th,P1);                                    //    finite element space
real rad=1e-8, uek=ue+273;                        //    def of the physical constants
Vh vold,w,v=u0-ue,b;
problem thermradia(v,w)
    = int2d(Th)(v*w/dt + k*(dx(v) * dx(w) + dy(v) * dy(w)))
                + int1d(Th,1,3)(b*v*w)
                - int2d(Th)(vold*w/dt) + on(2,4,v=u0-ue);

for(real t=0;t<T;t+=dt){
    vold=v;
    for(int m=0;m<5;m++){
       b= alpha + rad * (v + 2*uek) * ((v+uek)^2 + uek^2);
       thermradia;
    }
}
vold=v+ue; plot(vold);
```

3.5 Irrotational Fan Blade Flow and Thermal effects

Summary *Here we will learn how to deal with a multi-physics system of PDEs on a Complex geometry, with multiple meshes within one problem. We also learn how to manipulate the region indicator and see how smooth is the projection operator from one mesh to another.*