Assignment 2: Variational forms in Femlab

Femlab is a general-purpose finite-element package developed by Comsol in Sweden. The software can be used in several ways. There is a nice Graphical User Interface to interactively build the application, and there are also predefined templates and interfaces for specific applications. But equations and boundary conditions can also be specified at a lower level: in terms of the coefficients in a very general system of partial differential equations (*coefficient form*), in terms of flux functions in a conservation-law formulation (*general form*), or directly as the variational form used for the finite-element discretization (*weak form*).

Femlab can be run on its own, but here we will use the Matlab interface to Femlab, and write the application in terms of a m-file.

Femlab is installed at the Unix system at the IT-department, and you have access to the Femlab commands as soon as you start Matlab. The Matlab desktop tend to be slow on Unix systems, which is why it may be a good idea to start Matlab from the Unix prompt with

```
triangeln> matlab -nodesktop
```

The Femlab commands are used similarly as Matlab commands; for instance, you can as usual type `help whatever` to get help with the command `whatever`.

Some commands gave problem when running Femlab under Matlab 7.0; use the default Matlab version which should be Matlab 6.5.

Initialization

All data used for setting up a "Femlab model" is collected in the *Femlab structure*, which is the main data type used as in- and output of the Femlab commands. When setting up a new application, it is good practice always to start by clearing the variable name you have in mind for the Femlab structure, for instance,

```
clear fem;
```

Below, we will define fields of the structure `fem` that will contain all the information that Femlab needs. For instance, `fem.geom`, `fem.mesh`, and `fem.equ` will contain information on the geometry, the mesh, and the equation.

Geometry definitions

Femlab can operate in one, two, or three space dimensions. You can create two-dimensional *solid objects* enclosed by circles, rectangles, squares, and ellipses. These can be inserted as *geometry objects* into the Femlab structure. For example,

```
fem.geom = circ2(0.5, 1, 2);
```

creates a disc object enclosed by a circle with radius 2 centered at $x = 0.5$, $y = 1$ and inserts it as the geometry object in the Femlab structure. To plot the geometry, do

```
geomplot(fem);
```

The circle will look like an ellipse. To avoid this, type

```
axis('equal');
```

which produces equally-spaced tick marks in the x and y directions. To see how solid objects enclosed by rectangles, squares, and ellipses are created, type `help rect2`, `help square2`, and `help ellip2`.

New solid objects can be created by point-set operations such as as union, intersection, and difference through the operators `+`, `*`, and `-`. For example, the commands

```
c2 = circ2(0, 0, 2);
c1 = circ2(0, 0, 0.6);
r1 = rect2(-2.5, 2.5, -2.5, 0);
r2 = rect2(0.75,2.5,-1,0);
g = r1*(c2 - c1) + r2;
fem.geom = geomdel(g);
geomplot(fem);
axis('equal');
```

creates the discs C_2 and C_1, the rectangular objects R_1 and R_2 and the machine-part-like object $G = R_1 \cap (C_2 \setminus C_1) \cup R_2$. The command `geomdel(g)` deletes the internal borders between the parts of G. This is important when meshing and defining boundary conditions. (To see the difference, plot `g` before and after applying `geomdel`.)

The boundary of a geometry object is subdivided into a number of *boundary segments*. The command

```
geomplot(fem,'EdgeLabels', 'on');
```

makes visible the numbering of these segments. This numbering is used to set different boundary conditions on different parts of the boundary.

Meshing

If the Femlab structure `fem` contains a geometry object `fem.geom`, a triangular mesh is generated simply by typing

```
fem.mesh = meshinit(fem);
```

To plot the mesh, type

```
meshplot(fem);
axis('equal');
```

To control the size of the triangles, you can set an optional property `hmax`,

```
fem.mesh = meshinit(fem, 'hmax', 0.2);
```

which causes each triangle-side not to exceed, in this case, 0.2.

The variational form

We consider setting up the definitions needed to solve the Poisson problem

$$\begin{aligned} -\Delta u &= 1 \qquad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned}$$

Corresponding variational problem is

$$\begin{aligned} &\text{Find } u \in H_0^1(\Omega) \text{ such that} \\ &\int_\Omega \nabla v \cdot \nabla u \, d\Omega = \int_\Omega v \, d\Omega \qquad \forall v \in H_0^1(\Omega). \end{aligned} \tag{1}$$

The commands

```
fem.sdim = {'x' 'y'};
```

and

```
fem.dim = 'u';
```

specify the names of the spatial coordinates and the unknown. The above names are the default ones for scalar equations in two space dimensions, so if you are happy with these, you do not have to specify `fem.sdim` and `fem.dim` at all.

To specify that we want to give the equation in variational form, set

```
fem.form = 'weak';
```

Then, the command

```
fem.equ.weak = 'u_test-ux_test*ux-uy_test*uy';
```

specifies the variational form (1).

- A text string in `fem.equ.weak` thus contains the argument for *domain* integrals.
- Note that all terms in the variational form (1) are moved to the right-hand side when specifying `fem.equ.weak`.
- `ux` and `uy` means derivative of `u` with respect to `x` and `y`. (Here the name of the unknown variable and the spatial coordinates should be according to the definitions of `fem.dim` and `fem.sdim`)
- The addition `_test` after an unknown variable indicates test function.

If needed, you can add a *boundary integral* to the variational form by giving the integral argument as a string in the field `fem.bnd.weak`. (However, there are no boundary integrals in variational form (1).) In such boundary expressions, the symbols `nx` and `ny` are used to access the x- and y-components of the outward-directed unit normal. Setting `fem.bnd.weak` to a single string expressions specifies a integral over the whole boundary. If you want to have different expressions at different portions of the boundary, you need to use the numbering of the boundary segments discussed above. Assume for instance that the boundary consists of 3 boundary segments. To specify different integral expressions for these three boundary segments, set

```
fem.bnd.weak = {{string1},{string2},{string3}};
```

where `string1`, `string2`, `string3` are strings containing the symbolic expressions for the integrals over boundary segments 1, 2, and 3, respectively. The string `'0'` specifies that the boundary integral should not include corresponding segment.

The above sets up the variational form in a finite-element subspace to $H^1(\Omega)$, which would be correct if a "natural" boundary condition would be specified on the domain boundary. However, problem (1) is defined in $H_0^1(\Omega)$, so we need to specify the "essential" boundary condition $u = 0$ on $\partial\Omega$. This condition is regarded by Femlab as a *boundary constraint* on u and is specified by the command

```
fem.bnd.constr = 'u';
```

which specifies that u should vanish on the boundary. More complicated essential boundary conditions are specified again by referring to the numbering of the boundary segments. If 3 segments comprise the boundary, the command

```
fem.bnd.constr = {{'u-x'},{'u-1'},{'0'}};
```

specifies that $u = x$ on boundary segment 1, $u = 1$ on boundary segment 2, and that there are no constraints on boundary segment 3 (that is, the natural boundary condition applies on segment 3)

Finally, do the following commands:

```
fem.xmesh = meshextend(fem);
fem.sol = femlin(fem);
postsurf(fem,'u','triz','u')
```

The function `meshextend` performs various preprocessing, `femlin` solves the linear system, and `postsurf` visualizes the result.

To do

In the domain depicted in figure 1, fluid with zero temperature is coming in on the top and escaping through boundary Γ_1. The boundary Γ_0 is held at zero temperature, whereas constant heating is applied to boundary Γ_h. Denoting the temperature field by u and the given velocity field by $\boldsymbol{U}$, the system

$$\begin{aligned} -\nu\Delta u + (\boldsymbol{U}\cdot\nabla)u &= 0 \qquad \text{in } \Omega, \\ u &= 0 \qquad \text{on } \Gamma_0, \\ \frac{\partial u}{\partial n} &= 0 \qquad \text{on } \Gamma_1, \\ \frac{\partial u}{\partial n} &= 1 \qquad \text{on } \Gamma_h. \end{aligned} \tag{2}$$

models the situation.

Set up and solve problem (2) in Femlab by specifying the variational form as described above. The divergence-free velocity field $\boldsymbol{U}$ should correspond to a constant

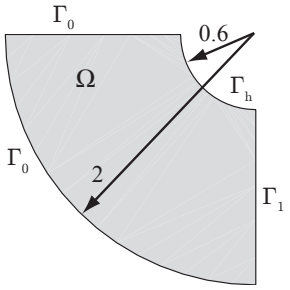

Figure 1: The domain form the advection–diffusion problem

rotation around the origin in the counter-clockwise direction such that at each point in Ω, $|\boldsymbol{U}| = r$, where r is the distance to the origin.

Solve the problem for different values of the thermal diffusivity ν. Study particularly what happens for small values of ν and discuss both mathematically and physically what happens.