TMA372/MAN660 PARTIAL DIFFERENTIAL EQUATIONS
COMPUTER ASSIGNMENT 1

1. Purpose

The purpose of these exercises is

1) to help you better understand the basic differential equation models used in the course. In particular, we study the roles played by the different terms in the differential equations and the signs and sizes of the coefficients;

2) to help you understand how well a finite element solution approximates the solution to a PDE and where and why there can be problems. The idea of adaptivity and the structure of an adaptive algorithm are also important. In particular, we shall see that adaptivity and local mesh refinement can compensate for local non-smoothness of the solution, i. e., by using local mesh refinement it is possible to get the same accuracy as if the solution were smooth;

3) to let you solve a "real world" problem or, at least, taste solving such problems, by constructing or completing adequate mathematical models and then simulate the real world phenomena on the computer using an adaptive finite element program.

Remember that the emphasis should be put on getting a feeling for the equations and their behaviour. As for the "real world" problems, they're not at all fixed, rather, you are encouraged to make up your own application or modify one of the suggested.

You are expected to hand in a *brief* report on these exercises in order to obtain bonus points. What I want is only some notes on what you have learned when working through this assignment. When you need a figure to explain what you mean, include it, but otherwise leave it out. For the "real world" problems, though, a somewhat more complete description of the problem and how you solved it is requested.

2. Model understanding—an excursion

2.1. **A general second order equation model.** We will study the second order differential equation

$$-D(d(x)Du) + c(x)Du + a(x)u = f(x), \quad \text{for } x \in (0, 1),$$

together with the boundary conditions

$$-dDu + ku = g \quad \text{or} \quad u = g \quad \text{at } x = 0,$$

and

$$dDu + ku = g \quad \text{or} \quad u = g \quad \text{at } x = 1.$$

This equation models diffusion type problems. The terminology used here will be that of what we in general think of as diffusion, namely transport of some chemical substance, but you can apply the model to e. g. heat conduction problems as well. The first term in the equation models the diffusive process, and the quantity

$$-dDu(x)$$

is the diffusive flux. The coefficient *d(x)* is the *diffusivity*. The second term is the convective transport, where *c(x)* is the flow velocity. The third term represents absorption of the species and the term on the right hand side is a source term.

The boundary conditions can also be written

$$\pm dDu = k(u - \tilde{g})$$

which might be physically more intuitive. The flux over the boundary is proportional to the difference between the solution at the boundary point and some exterior quantity. Here k (the proportionality constant) is a transfer coefficient.

2.2. **AdFEM.** The tool used to study this equation will be the MATLAB program AdFEM. AdFEM is a finite element program that uses piecewise linear elements and adaptive mesh refinement to find an approximate solution to the equation described above. You can download the program from

http://www.math.chalmers.se/~mohammad/pde1_lab/lab1/adfem.zip

To unzip the files, use 'unzip' (on UNIX machines) or some version of 'pkzip' (on DOS-based machines).

The program is started with the command >> `adfem` in MATLAB. You will be asked to give the coefficients in the model, the boundary conditions etc. The program will then try to solve the problem, refining the mesh if necessary, and present the solution in a plot together with the residual and the meshsize.

If you want to change the problem you can type e. g. >> `convection='1-.5*x'` followed by >> `solve` and AdFEM will try to solve the problem with the new convective velocity. The other coefficients are called `diffusion`, `absorption` and `force`. Check the routine `adfem.m` to find the variable names for changing boundary conditions, domain, tolerance etc. Be sure to get the type correct, i. e. check if you should give a string or a numerical value.

For a more extensive AdFEM manual, see
http://www.math.chalmers.se/~mohammad/pde1_lab/lab1/adfem.ps
(PostScript version), or
http://www.math.chalmers.se/~mohammad/pde1_lab/lab1/adfem.pdf
(PDF version).

2.3. **The excursion.** What will follow are four different sets of equation coefficients which are intended to show how the diffusion and the convection influence the solution. The absorption and source terms will not be considered here. There are also some questions and hints given to help you 'see' the different phenomena. In the next section there are also some questions concerning the numerics, the error and the adaptive routines. Read those questions before doing the simulations.

First, we will study the diffusivity. Solve the problem with $d(x) = 1$, $c(x) = a(x) = 0$, $f(x) = 1$ and take the boundary conditions to be of Dirichlet type and homogeneous ($= 0$), i. e., take $u = 0$ at both $x = 0$ and $x = 1$. Why does the solution look like this? How can we interpret the boundary conditions? If we change to homogeneous Neumann conditions at both ends, what happens? How do we now interpret the boundary conditions? Finally, change to the inhomogeneous Neumann condition $u'(1) = -1$ (keep $u'(0) = 0$). Is this solution unique?

Next, change the boundary conditions back to homogeneous Dirichlet type, change the diffusivity to

$$d(x) = 0.01 + x^4$$

(type >> `diffusion='0.01+x^4'`) and put

$$f(x) = \sin(40 * x).$$

Now, near x=0 the diffusivity is small and near x=1 it is larger. How will this influence the solution? What can we say about the role of the diffusivity?

Change the diffusivity back to 1 and set >> `force='1/x'`. Here we have a source that is singular at the left end point of the interval. Why does this not cause any problems in the solution when we have such a large diffusivity?

We will now add some convection to the equation. Let d=0.02, set >> `convection='1'` and change the source back to 1 (>> `force='1'`). Why does the solution look like this? (*Hint:* What happens if you set d=0, is the solution compatible with the boundary conditions? Compare with Problem 18.5 on page 459 in the book.)

2.4. **Error and Adaptivity.** There are two main questions that you need to think about here. The first is how close to the exact solution it is possible to come with this numerical method, and the second is how do I choose my mesh in order to get as close to the exact solution as I want to in an efficient way.

To answer the first one, you should study the relation between the solution and the residual. Where is the residual large and what does the solution look like there? (What *is* the residual?) If you haven't done the preparatory exercises, you may need to take a look at them. The theory behind this is given in Chapter 5 in the book.

The answer to the second one can be found in the relation between the residual and the meshsize, e. g. how large is $hR(U)$? If you don't want to figure it out for yourself, you can read Section 8.2.3 in the book.

3. Applications

Choose one of the following, or make up your own.

3.1. **Heat Conduction Through a Wall.** A wall consists of three layers: the core of the wall is 25 cm of isolation material with heat conductivity 0.024 W/(m K) and on each side of this core, there is a 3 cm layer of wood, heat conductivity 0.14 W/(m K). What is the heat flow through the wall if the inner temperature is $T_{in} = 21°$ C and the outer temperature is $T_{out} = -10°$ C.

The solution procedure can be split in the following way: a) Derive the boundary value problem

$$-(d(x)u'(x))' = 0, \quad -1 < x < 1; \quad u(-1) = g_0, \ u(1) = g_1,$$

where all quantities are dimensionless.

b) Solve the equation by hand when

$$d(x) = \epsilon(1 + (\beta x)^2)$$

and use this to check that the program is correct and gives the right answer. It is always a good thing to check a program on an easy but non-trivial model problem.

c) Solve the given problem using AdFEM. (*Hint:* The routine `diff` in MATLAB gives the difference between consecutive components in a vector. The meshsize used is stored in the vector `h` and the solution is stored in `u`. Use this to compute a difference approximation to the flux.)

3.2. **Time-dependent Heat Equation.** Write your own finite element code that solves the one dimensional heat equation:

$$\dot{u} - (d(x)u'(x))' = 0, \quad -1 < x < 1; \quad u(-1) = g_0, \; u(1) = g_1,$$

subject to some initial condition

$$u(x, 0) = u_0(x)$$

Use the cG1 method in space and compare the cG1 and dG0 methods for discretizing in time. Can you see any differences?

3.3. **Time-dependent Convection-diffusion Equation.** Simulate how a pollution spreads in a flowing liquid using the MATLAB program *ibvp*. It can be downloaded from

http://www.math.chalmers.se/~mohammad/pde1_lab/lab1/ibvp.zip

This program is similar to AdFEM, you start it with >> `ibvpI`. Choose the coefficients and boundary conditions (motivate your choice) and set the initial value to e. g. a narrow Gauss curve. Study the influence of the diffusivity and the convective velocity. How does the adaptivity work? You may want to go ahead in the book and read a little in Chapter 18.