Project 1: Solution of boundary value problems

1 The problem

In this project the aim is to write a program that computes an approximate solution to the boundary value problem

$$-\varepsilon \frac{d^2u}{dx^2}(x) + \beta(x)\frac{du}{dx}(x) + \sigma(x)u(x) = f(x) \text{ in } (0,1) \tag{1}$$
$$u(0) = a \tag{2}$$
$$u(1) = b \tag{3}$$

where $\varepsilon \in \mathbb{R}^+$, $a, b \in \mathbb{R}$ and β and σ are two functions that may be chosen. Observe that we here consider an equation set in only one space dimension, that we could solve analytically in several situations, but the principles are similar in the higher dimensional case, where the equation is written

$$-\varepsilon \Delta u + \overrightarrow{\beta} \cdot \nabla u + \sigma u = f.$$

Depending on the choice of ε, β and σ this equation can model a variety of physical phenomena, such as transport of pollutants in the environment or wave propagation.

Following what we did in class we approximate $u : (0,1) \mapsto \mathbb{R}$ as a vector $U \in \mathbb{R}^{N-1}$ so that for $h = 1/N$ and $i \in \{1, \dots, N-1\}$, $U_i \approx u(x_i)$ where $x_i = i \times h$. Observe that we have not included approximations of $u(0)$ and $u(1)$ in U, since these two values are known through (2)-(3).

We recall the finite difference approximations

$$\frac{d^2u}{dx^2}(x_i) \approx (D^2U)_i = (U_{i+1} - 2U_i + U_{i-1})/h^2 \quad i = 1, \dots, N-1$$

and

$$\frac{du}{dx}(x_i) \approx (DU)_i = (U_{i+1} - U_{i-1})/(2h), \quad i = 1, \dots, N-1.$$

Using these approximations of the derivatives we may write an approximate (finite dimensional) problem:

$$\varepsilon(D^2U)_i + \beta(x_i)(DU)_i + \sigma(x_i)U_i = f(x_i), \quad i = 1, \dots, N-1 \tag{4}$$
$$U_0 = a \tag{5}$$
$$U_N = b. \tag{6}$$

Observe that the system (4) corresponds to $N-1$ linear equations and can therefore be written on the form

$$AU = F \tag{7}$$

where $A \in \mathbb{R}^{N-1\times N-1}$ is the system matrix depending on the equation parameters, $U \in \mathbb{R}^{N-1}$ is the approximate solution and $F \in \mathbb{R}^{N-1}$ is a data term depending on $f(x)$, a and b. Alternatively the two equations for the boundary condition (5) and (6) can be included in (7) as they stand and the dimension of the system is then $N+1$.

2 The project

The aim of the present project is to use the elements above to write a program that computes approximations of the differential equation (1)-(3) using the discrete formulation (4)-(6). Below follows a point by point outline of how the program can be constructed and what it should deliver.

2.1 Program structure

1. Open file on the form `filename.bvp` for read.
2. Read the data ε, β, σ, f, a and b from the file.
3. Also read from the file a string `exact_solution` that either contains the exact solution or the string `'unknown'` and the parameters `N` and `Nlev`
4. An example of the file (`diffusion.bvp`), showing how the data should be organised, reads:

   ```
   1.0
   0.0
   0.0
   1.0
   0.0
   0.0
   x*(1-x)/2.
   10
   4
   ```

5. Construct the linear system (4)-(6) for the given data.

6. Solve the linear system, using the iterative solver that you programmed in week 3, or otherwise. For instance you can use Crout factorization since the system is tridiagonal, see `http://www.physics.arizona.edu/~restrepo/475A/Notes/sourcea-/node69.html`.

2.2 Computation and output

1. if an exact solution is given, i.e. `exact_solution != 'unknown'`:

   - compute the approximation $U \in \mathbb{R}^{2^k N-1}$ with $h = 1/(2^k N)$ for $k = 0, \ldots,$ `Nlev-1`.
   - for each approximate solution compute the approximate error $e_i = u(x_i) - U_i$ for all nodes x_i on the level. Then compute

   $$e_{max} = \max_i |e_i|, \quad e_{l2} = \left(h \sum_i |e_i|^2 \right)^{\frac{1}{2}}.$$

   - Report the results in a table on the form:

   $h \quad e_{l2} \quad \mathrm{eoc}(e_{l2}) \quad e_{max} \quad \mathrm{eoc}(e_{max})$

   followed by the values of the different quantities for the different mesh-sizes h. Here "eoc(e)" stands for '*experimental order of convergence* and is defined by

   $$\mathrm{eoc}(e_k) = \frac{\log(e_{k-1}/e_k)}{\log(2)}$$

   where e_k is the error on level k. Clearly the eoc can not be computed for the first computation, $k = 0$.
   If the error behaves as $e = O(h^\alpha)$, eoc is an approximation of α.
   - Plot the exact solution and the approximate solution in the interval $(0, 1)$

2. if `exact_solution = 'unknown'`:

   - Plot the approximate solution in the interval $(0, 1)$.

3 What to submit for the assessment

You should submit the following items in a zipped folder:

- A .pdf document with details on the members of the group, names and student numbers. Here you may also give some information on the project, such as if some of the extensions have been considered.
- A well commented code that produces the expected output as outlined in the discussion above.
- The output of the program run using the following three input files (available on the moodle page):
  - `diffusion.bvp`
  - `wave.bvp`
  - `convect_diffusion.bvp`
- Include both text (i.e. the tables discussed in the previous section) and graphical output (graphics can be saved using the buttons on the bottom of the Figure window).

3.1 Possible extension

These extensions are not required for the project, but for those whose are curious to try some more aspects of the BVP.

1. If the exact solution is unknown, a computation can be made on a very fine grid first and then the results of this computation is used as exact solution (interpolating between the values if necessary). Modify your code so that this procedure is used if `exact_solution = 'approximate'`
2. (Challenging) If $\beta(x)$ in (1) is replaced by $u(x)$ and $\sigma \equiv 0$ the problem becomes the stationary, viscous Burgers' equation. This is one of the most well known model problems in fluid mechanics modelling fundamental aspects of nonlinear phenomena such as turbulence and shock waves. The discretization of this equation is nontrivial. Can you modify your program to solve this problem? *Hint: You will need to implement a version of your fixed point iteration for the system.*