ENG3165 – Numerical Methods coursework – 2017/18

This coursework is worth 50% of the ENG3165 Numerical Methods & CFD module

Instructions – IMPORTANT: please read carefully

The coursework involves solving two engineering problems by applying numerical methods for partial differential equations and Matlab programming. Coding is only a **minor part** of the assessment (please see the marking scheme for each problem). That means that it is more important to answer the questions correctly by using a Matlab code that works, rather than producing the best possible computer programme.

Some parts of the coursework won't depend on Matlab programming, but on mathematical derivations and written explanations and comments. Matlab must only be **used where specified** to obtain numerical answers or plots.

Several questions will require using data values that are different depending on you URN, and in particular, the last two digits. Make sure you use the right values from your URN. **Double check them before submitting**. If you fail to use the correct URN values, I'm afraid I cannot help.

Any evidence of **plagiarism** or **collusion** with others will be dealt with severely under the University rules. Please ensure that what you submit is your work and only your work; do not "assist" or seek assistance from others by e.g. "sharing" parts of programmes, and do not under any circumstances copy code from books/websites. Both the coursework paper and the code will be submitted to **specialist plagiarism-detection software** to spot potential problems.

As part of this coursework you will be required to write and submit several Matlab codes. As the University has a **site licence for Matlab**, if you do not already have a copy, or have access to a copy, you should be able to get one for personal University use. To obtain this go to https://www.surrey.ac.uk/fepsit/services/ and follow instructions. Please contact IT services if you have a problem: I am afraid I cannot help with this.

The coursework paper must be submitted as an electronic document (either a Word document, **.docx** or **.doc**, an OpenOffice/LibreOffice document, **.odt** or a PDF file **.pdf**). Mathematical formulas can be entered using the Word built-in "Equation" tool (or OpenOffice/LibreOffice equivalent) or by digitalising (picture or scan) a hand-written paper and including it into the main .docx/.doc/.odt/.pdf file. The former method is recommended and preferred. Please **submit only one document** containing the whole coursework paper. If you submit multiple documents I will only assess the latest submission unless otherwise specified.

You must submit **all your Matlab codes as m-files**. Do not copy them onto the main document. A single script per problem that will run all the smaller scripts and functions for me when I'm assessing your code would be preferable but not mandatory. If your numerical answers and plots are not supported by the code you've submitted, then they won't be considered correct. You can submit multiple files on SurreyLearn, even at different points in time.

Submit your coursework paper and Matlab codes (m-files) to your "Numerical Methods Coursework" assignment folder on SurreyLearn. The deadline is Tuesday 9th January 2018 at 16:00. Do allow plenty of time for submission, as the process will likely take a while: do not try to submit at the very last minute. The time of submission of the last file is the one that will be considered for the whole coursework for late submission purposes. Please submit all the individual files separately, do not compress them in a single .zip file.

Data for use in coursework

Please include the following info at the beginning of you coursework document:

Name:

Surname:

URN:

Last two digits from the URN:

6	x	x	x	x	x	x
					$\boldsymbol{a}_{\mathbf{URN}}$	$\boldsymbol{b}_{\mathbf{URN}}$

a_{URN} =

b_{URN} =

Declaration of originality

(see document in SurreyLearn)

Problem 1 [50 marks]

We consider the problem of optimising the cross section of a channel of rectangular shape. The cross section, shown in Figure 1, is assumed to have a perimeter $l = b + 2h$, which is directly proportional to the amount of material required. Therefore, the shape can be described in terms of two parameters/inputs, the size of the base in the rectangle b, and the height of the cross section h.

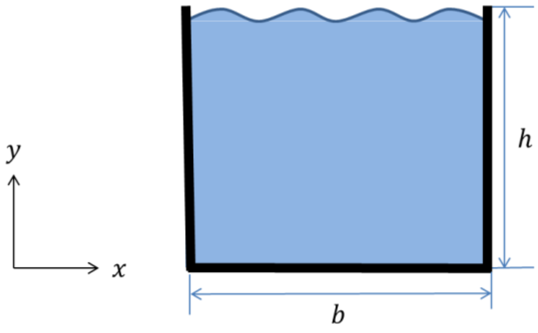

Figure 1: Rectangular cross section

We are interested in the flowrate Q defined as the integral of the velocity u (normal to the cross section) over the cross section Φ

$$Q = \int_{\Phi} u \, dx \, dy \tag{1}$$

The channel is at a slope of angle α, as shown in Figure 2, and the horizontal components of the gravitational force creates the pressure gradient for the downwards flow. The governing equations are the Navier-Stokes equations,

$$\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \boldsymbol{f} + \nu \nabla^2 \boldsymbol{u} \tag{2}$$

Note 1: the above Navier-Stokes equation is written in vector notation. We will develop the full equations component-by-component in 2D below.
Note 2: don't be scared by the Navier-Stokes equation above. We won't solve that, but instead will simplify the equation further as you will see below

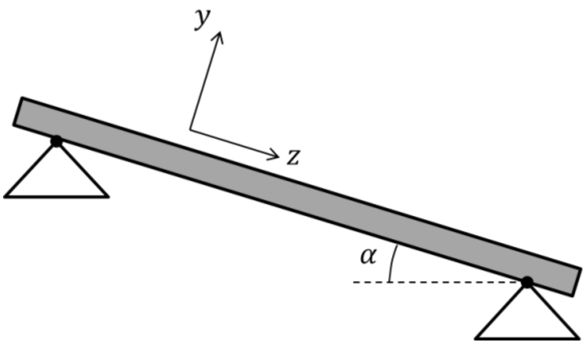

Figure 2: Flow channel

With the assumption of fully developed flow, these can be reduced to Poisson's equation for the velocity component u normal to the channel cross section,

$$-\nabla^2 u = \frac{g \sin \alpha}{\nu} \tag{3}$$

Here, g is the gravitational constant, and ν is the kinematic viscosity of the fluid. For simplicity, we assume that

$$\frac{g \sin \alpha}{\nu} = 1 \tag{4}$$

Along the walls of the channel the velocity is zero, and on the free surface a zero-stress condition $\left(\frac{\partial u}{\partial n} = 0\right)$ is assumed.

We will solve Poisson's equation using a finite difference procedure. Due to symmetry we only consider half the channel section, as shown in Figure 3. The following boundary conditions are applied to the original domain:

$$\begin{cases} \dfrac{\partial u}{\partial n} = 0 & in\ AB\ and\ AD \\ u = 0 & in\ BC\ and\ CD \end{cases} \tag{5}$$

Note 3: the above normal derivative will, of course, be $\frac{\partial u}{\partial x}$ along AB, and $\frac{\partial u}{\partial y}$ along AD.

We want to solve the problem in the rectangular half-domain Ω, using the finite difference method. We use a rectangular grid with $N \times N$ points in the computational domain, giving rectangular cells of size $\Delta x = \frac{b}{2(N-1)}$ and $\Delta y = \frac{h}{N-1}$.

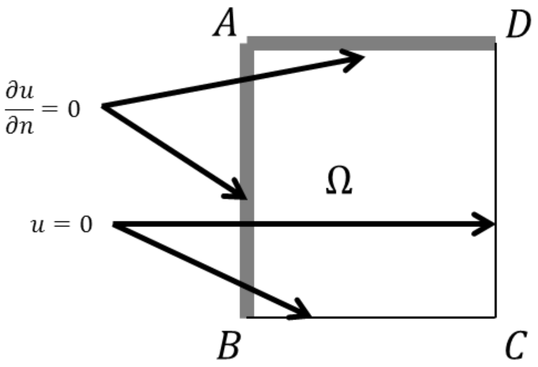

Figure 3: Geometrical representation of the computational domain

Equations (3) and (4) can be written in component notation as

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -1 \tag{6}$$

Questions for Problem 1

1) Using the Taylor series expansion, show that the central difference method for the second derivative is second-order accurate.

[8 marks]

2) Use a second-order central difference method to derive the finite difference schemes for the discretisation of all derivative terms for equation (6) in the interior of the domain. Use a simpler first-order scheme for the Neumann boundary points. You do not have to derive special schemes for the corner points, use the left boundary scheme at corner A, and $u = 0$ at the other three corners.

[10 marks]

3) Write a Matlab programme to solve the above problem. Then, using your code, calculate the solution $u(x, y)$ using a grid size $N = 41$, for $l = 3.0 + \frac{a_{URN}}{20}$ and $b = 1.0 - \frac{b_{URN}}{50}$, where a_{URN} and b_{URN} are the last two digits from your URN. Plot the solution (contour plot).

[12 marks]

4) Write a Matlab programme to calculate the flow rate from the final $u(x, y)$ values resulting from question 3. If you have not solved question 3, you can still answer this question by using a random-value $u(x, y)$ matrix with the correct number of rows and column, instead of the actual solution from question 3. In order to calculate the flow rate using equation (1) you will need to apply a numerical integration scheme as explained in the box below.

[5 marks]

Numerical integration schemes were introduced in Year 2 Numerical Methods. Even if you don't recall them, you should be able to follow the procedure below that will allow you to calculate the double integral in equation (1). We are going to use the simplest numerical integration scheme, that is the trapezoidal rule. Assuming we have a matrix with values $u(x_i, y_j)$, with $i = 1, .., N$ and $j = 1, .., N$, you can calculate a single integral along *x* for each y_j value as follows:

$$I(y_j) = \int_x u(x, y_j)dx \cong \frac{\Delta x}{2}\left[u(x_1, y_j) + 2\sum_{i=2}^{N-1} u(x_i, y_j) + u(x_N, y_j)\right]$$

Once you have calculated the above integral for all y_j values, you can then calculate the double integral in equation (1) using the same technique:

$$Q = \int_\Phi u\,dx\,dy = \int_y I(y)dy \cong \frac{\Delta y}{2}\left[I(y_1) + 2\sum_{j=2}^{N-1} I(y_j) + I(y_N)\right]$$

Hint: remember that you are only computing *half* of your physical domain. In order to obtain the actual flow rate over the *whole* section, you'll have to adjust your results accordingly.

5) Write a Matlab programme to calculate the convergence rate for the L_2 norm of the solution. Use the grid sizes $N = 11, 21, 41, 81$. Since we don't know the exact solution, use the solution for $N = 81$ as a reference. To calculate the L_2 norm, only use the reference grid points that correspond to the coarsest grid. This will avoid interpolation problems, keeping the complexity of your code at a minimum. Comment on your results. If you have not solved questions (3) and/or (4), you can still comment on what is your expected convergence rate of this numerical scheme, from theoretical considerations, to get at least partial marks.

[10 marks]

Error-free, clean, well commented and well indented Matlab code will get you additional 5 marks. An efficient, vectorised code with sensible and justified use of user-defined functions can get you a bonus of up to 5 marks (provided that the total marks for Problem 1 do not exceed 50).

[5 (+5) Marks]

Important: in order for me to assess whether your answers are supported by your Matlab code when required, please **write down detailed instructions on how to run your code** to obtain the above answers.

Problem 2 [50 marks]

In lectures and tutorials we have studied diffusion problems in 1D and 2D in depth. In this problem we'll be looking at some more interesting physics: a model represented by *reaction-diffusion* equations. It is a system that has the physics of diffusion but also has some kind of reaction that adds different behaviours to the solution. In particular we are going to look at the *Gray-Scott* model, which simulates the interaction of two generic chemical species reacting and, of course, diffusing. Some amazing patterns can emerge with simple reaction models, eerily reminiscent of patterns formed in nature. See for example the following video by Karl Sims on YouTube; it almost looks like a growing coral reef…

https://youtu.be/8dTmUr5qKvI

The Gray-Scott model represents the reaction and diffusion of two generic chemical species, U and V, whose concentration at a point in space is represented by variables u and v. The model follows some simple rules:

- each chemical *diffuses* through space at its own rate;
- species U is added at a constant feed rate into the system;
- two units of species V can "turn" a unit of species U into V: $2V + U \rightarrow 3V$;
- there is a constant kill rate removing species V.

This model results in the following system of partial differential equations for the concentrations $u(x, y, t)$ and $v(x, y, t)$ of both chemical species:

$$\begin{aligned} \frac{\partial u}{\partial t} &= D_u \nabla^2 u - uv^2 + F(1-u) \\ \frac{\partial v}{\partial t} &= D_v \nabla^2 v + uv^2 - (F+k)v \end{aligned} \tag{7}$$

You should see some familiar terms, and some unfamiliar ones. On the left-hand side of each equation, we have the time rate of change of the concentrations. The first term on the right of each equation corresponds to the spatial diffusion of each concentration, with D_u and D_v the respective rates of diffusion. In case you forgot from Problem 1, the ∇^2 operator is the Laplacian:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{8}$$

The second term on the right-hand side of each equation corresponds to the reaction. You see that this term decreases u while it increases v by the same amount: uv^2. The reaction requires one unit of U and two units of V, resulting in a reaction rate proportional to the concentration u and to the square of the concentration v. This result derives from the *law of mass action*. We assume here a reaction rate constant equal to 1, which just means that the model is non-dimensionalised in some way.

The final terms in the two equations (7) are the "feed" and "kill" rates, respectively: $F(1-u)$ replenishes the species U (which would otherwise run out, as it is being turned into V by the reaction); $-(F+k)v$ is diminishing the species V (otherwise the concentration v would simply increase without bound).

The values of F and k are given parameters that can be modified, together with the diffusion constants, to see what happens in the system.

The system will be represented by two matrices, u and v, holding the discrete values of the concentrations u and v, respectively. In order to initialise the u and v variables correctly you'll need to download the `uvinitial.mat` file from the "Numerical Methods > Coursework" folder on SurreyLearn and save it into your working Matlab folder. In order to start from the same initial values you'll need to load these data into your code in the initialisation section of your Matlab programme by adding the following statement:

```
load('uvinitial.mat')
```

The above statement will add two variables, u and v, to your workspace with the correct initial values. You can subsequently use those two variables in your code without the need of initialising them. In order for the above Matlab statement to work, the `uvinitial.mat` file must be in the current working Matlab folder, i.e. the one where you are running your code from. The initial data look like the ones in Figure 4.

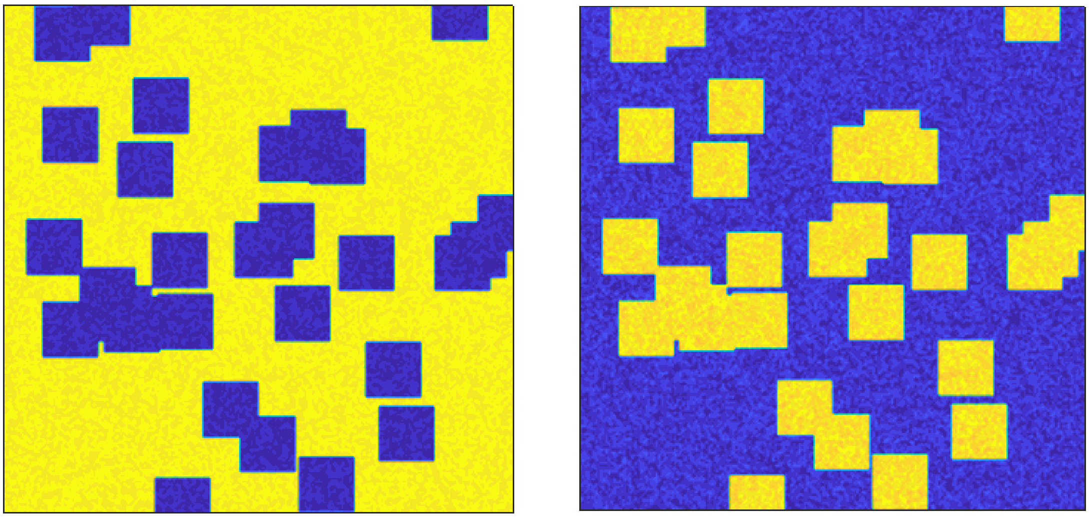

Figure 4: Representation (contour plot) of the initial u (left) and v (right) values

If you're curious about how the above initial conditions were built, here is a short description. We started by setting `u = 1` everywhere and `v = 0` everywhere, then introduce areas of difference (10 x 10-element patches) as initial conditions. We then added a little noise to the whole system to help the u and v reactions along. A short Matlab code was used for this purpose (see below):

```
% generate initial conditions for Problem 2 in ENG3165 coursework 2017-18
%
num_patches = 30; n = 192;
rand_x = randi(n,[1,num_patches]);
rand_y = randi(n,[1,num_patches]);
u = ones(n);
v = zeros(n);

r = 10;

for k = 1:length(rand_x)
    i = rand_x(k);
    j = rand_y(k);
    i_min = max(i-r,1); i_max = min(i+r,n);
    j_min = max(j-r,1); j_max = min(j+r,n);
    u(i_min:i_max,j_min:j_max) = 0.50;
    v(i_min:i_max,j_min:j_max) = 0.25;
end

u = u + 0.05 * rand(n);
v = v + 0.05 * rand(n);

save('uvinitial.mat', 'u', 'v')
```

NOTE: DO NOT USE THIS CODE IN YOUR COURSEWORK. We are showing it here to help you understand how the system is constructed. However, you *must use the data we've supplied above* as your starting condition.

Questions for Problem 2

1) Consider the simpler 1D diffusion equation (not the more complex reaction-diffusion equations set up above):

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} \tag{9}$$

where ν is the viscosity. Determine whether equation (9) is elliptic, parabolic or hyperbolic. Show your reasoning and calculations.

[5 marks]

2) Discretise the reaction-diffusion equations (7) in both the interior of the domain and the boundaries. Use a forward-time, central-space (FTCS) scheme for the interior points. Use zero Neumann boundary conditions on all sides of the domain and discretise using a first order special scheme; at the corners, use the zero Neumann boundary condition along x (in other terms: consider the corner as part of either the left or the right boundaries, not the bottom or top ones).

[15 marks]

3) Write a Matlab programme to solve the above problem. In particular, calculate the solutions $u(x, y, t)$ and v(x, y, t) using the following assumptions:
   - $\Delta x = \Delta y = \delta$
   - For your time step (in seconds), set

$$\Delta t = \frac{9}{40} \frac{\delta^2}{\max(D_u, D_v)}$$

   - Grid of points with dimension $N \times N$ points, where $N = 192$
   - Domain is 5 m x 5 m
   - Final time (in seconds) is $8000 - [(a_{URN} + 1) * b_{URN}]$
   - $D_u = 0.00016; D_v = 0.00008$
   - $F = 0.035; k = 0.065$

   Plot the final solution for both chemical species, using the `contourf` function (look it up in the Matlab help). [*Matlab tip: for a better visualisation of the results, you may want to remove the contour lines – by adding the option* `(..., 'LineColor', 'none')` *to the* `contourf()` *function – and use at least 20 contour levels*]

[15 marks]

4) Write a Matlab programme to calculate the evolution in time of the average concentration of the two species over the whole domain:

$$\bar{u}(t) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} u(x_i, y_j, t)$$
$$\bar{v}(t) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} v(x_i, y_j, t)$$

   Plot $\bar{u}(t)$ and $\bar{v}(t)$ vs. time on the same graph.

[10 marks]

Error-free, clean, well commented and well indented Matlab code will get you additional 5 marks. An efficient, vectorised code with sensible and justified use of user-defined functions can get you a bonus of up to 5 marks (provided that the total marks for Problem 2 do not exceed 50).

[5 (+5) Marks]

Important: in order for me to assess whether your answers are supported by your Matlab code when required, please **write down detailed instructions on how to run your code** to obtain the above answers.

Once you have completed the coursework, you might want to explore a few more of the interesting patterns that can be obtained with the Gray-Scott model. The conditions below will result in a variety of patterns and should work without any other changes to your existing code.

D_u	D_v	F	k	Description
0.00016	0.00008	0.035	0.065	Bacteria 1
0.00014	0.00006	0.035	0.065	Bacteria 2
0.00016	0.00008	0.060	0.062	Coral
0.00019	0.00005	0.060	0.062	Fingerprint
0.00016	0.00008	0.020	0.055	Unstable
0.00016	0.00008	0.050	0.065	Worms 1
0.00016	0.00008	0.054	0.063	Worms 2
0.00016	0.00008	0.035	0.060	Zebrafish

References:

http://www.karlsims.com/rd.html

http://science.sciencemag.org/content/261/5118/189

http://www.aliensaint.com/uo/java/rd/