PX390 Assignment 4: a chemical reaction in a circular flow reactor

November 2021

1 The description of the problem

An equilibrium chemical reaction characterised by the equation A $\underset{k_-}{\overset{k_+}{\rightleftharpoons}}$ B, where k_+ is the rate constant of the A $\longrightarrow$ B reaction, while k_- is the rate constant of the reverse reaction B $\longrightarrow$ A. In a homogeneous reaction mixture, the concentrations A and B follow the differential equations

$$\frac{\mathrm{d}A}{\mathrm{d}t} = -k_+A + k_-B \quad \text{and} \tag{1}$$

$$\frac{\mathrm{d}B}{\mathrm{d}t} = k_+A - k_-B. \tag{2}$$

This reaction is carried out in a circular reactor as shown in Figure 1, with a circumference of L where a pump is circulating a reaction medium at a constant speed v. It can be assumed that the reactor is thin. Reactant A is added to the vessel at a $S(x)$ source rate, while product B is removed at a rate that depends

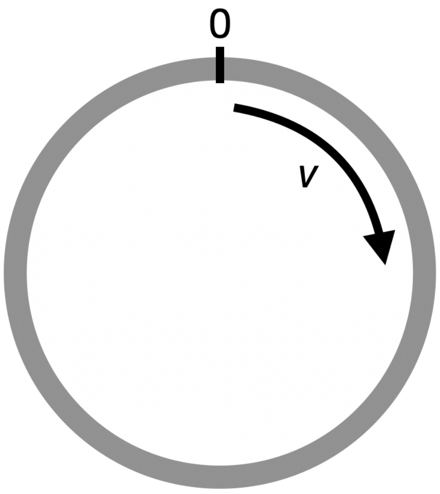

Figure 1: Schematic of the reactor.

on the local concentration $B(x)$ and a spatially varying coefficient $\sigma(x)$. Both compounds undergo diffusion with the same diffusion coefficient D. To summarise, the differential equations describing the variations of concentrations are

$$\frac{\partial A}{\partial t} = D\frac{\partial^2 A}{\partial x^2} - v\frac{\partial A}{\partial x} - k_+A + k_-B + S \tag{3}$$

$$\frac{\partial B}{\partial t} = D\frac{\partial^2 B}{\partial x^2} - v\frac{\partial B}{\partial x} + k_+A - k_-B - \sigma B. \tag{4}$$

Your task is to model the steady-state solutions $A(x)$ and $B(x)$ after a sufficiently long time, where

$$\frac{\partial A}{\partial t} = \frac{\partial B}{\partial t} = 0 \quad \text{resulting in} \tag{5}$$

$$
\begin{aligned}
0 &= D\frac{\partial^2 A}{\partial x^2} - v\frac{\partial A}{\partial x} - k_+ A + k_- B + S \\
0 &= D\frac{\partial^2 B}{\partial x^2} - v\frac{\partial B}{\partial x} + k_+ A - k_- B - \sigma B.
\end{aligned}
\tag{6}
$$

The length of the spatial domain is L and periodic boundary conditions should be used, i.e. $A(L) = A(0)$, $B(L) = B(0)$ etc., as well as assuming $k_+ > 0$, $k_- > 0$, $v > 0$, $D > 0$, $S(x) \geq 0$ and $\sigma(x) \geq 0$ for all x.

You are required to use the `LAPACK` library to invert matrices, so the compiler flags are different this time. This library is installed on the lab computers and on `nenneke`. We will not provide any support for installing these libraries on your own computers, it might be quicker just to log in to `nenneke`. In order to compile your program (here, named `prog4.c`) on `nenneke` you need to execute the following commands:

1. `module purge`
2. `module load intel impi imkl`
3. `gcc prog4.c -lmkl -liomp5 -lm`

You also need to include
`#include <mkl_lapacke.h>`
in your source code. As before, you should test your code with simple test cases. We will, again, be marking it mostly based on how well it reproduces certain tests.

2 Some suggestions

- You do not need to prove that a stable steady state solution exist, you can assume that for sensible input parameters such a solution exist.
- The scheme used to numerically solve the differential equations must be convergent and stable. You can assume that first order methods will satisfy these criteria. To evaluate the spatial derivatives in equation 6 we recommend the central method for the diffusion term, and the upwinding method for the advection term.
- You will be using a grid to represent x and the functions. For N grid points, we recommend to set up the x grid such that $0, \frac{1}{N}L, \frac{2}{N}L, \ldots, \frac{N-1}{N}L$.
- The easiest way to call `LAPACK` is to use the matrix example codes on the PX390 website as a 'skeleton': this provides a framework for setting the values of the matrix at particular locations and solving the inverse problem.
- For reasons of numerical performance, make sure your matrix is a narrow banded matrix.
- One way to check that your answer is right is to write a time-evolution code.
- There are analytical solutions to these equations for simple cases.

3 Specifications

The input and output x grid has N ($N \geq 3$) grid points, starting at 0, but not including the "right" end of the domain; this grid will be taken to run from $x_0 = 0$ to $x_{N-1} = \frac{N-1}{N}L$. The grid points you use in setting up the matrix problem are up to you.

3.1 Input

The scalar input parameters will be read from a file called `input.txt`. The content of the file should be

1. L: right x boundary of domain,
2. N: number of grid points,

3. D: the diffusion coefficient of A and B,
4. v: advection velocity,
5. k_+: the forward rate constant of the reaction,
6. k_-: the backward rate constant of the reaction.

We will leave you to determine whether these should be read as integers or floating point numbers based on their meaning (note that the example input file does not necessarily have decimal points on all the values that should be floating point). It is sufficient to use `scanf` to read in these parameters and simply test that the reads were successful: we are not specifically forbidding the use of other ways of reading input but we found roughly 50% of students attempting a more complicated input scheme got it wrong. The functions $S(x)$ and $\sigma(x)$ (in that order) will be given at the N grid points as two columns of double precision numbers in a file called `coefficients.txt`. There is an example input parameters file and coefficients file on the assignment page.

3.2 Output

The code should write the solution to a file `output.txt`, in the following order: $x, A(x), B(x)$ in three column format for the grid points $x_0, x_1, \ldots, x_{N-1}$.

4 Marking

I graded the assignments using the following marking scheme.

4.1 Compilation, completion: 4 marks

1 mark was deduced for compiler warnings, which mostly included unused variables.

4.2 Memory management: 3 marks

You have lost one mark if the pointers (all pointers) to allocated memory were not validated against the `NULL` value. Additional 1 mark was lost if the code was incorrectly accessing non existent array elements. `valgrind` was used to check for arrays that were not freed after usage, and 1 mark was deducted in such cases. Note that no marks were lost due to any memory errors related to memory leaks within LAPACK library.

4.3 Input and output: 2 marks

Files and format. If the input and output files were not implemented according to the specification, I deducted 1 mark. In some cases the solution was dumped on the standard output. I have captured these and used this output for the rest of the marking.

4.4 Readability, commenting, structure, conciseness: 1 mark

I only deducted points if the code was very difficult to read, either because of the indentation style, very long lines or/and lack of comments.

4.5 Test cases

Test cases were designed to test the implementation for a combination of input parameters. These and model solutions are described below. The main marking criterion was accuracy: I compared the relative root mean square error (RMSE) of the solution vectors to the model solution, using the following formula:

$$\text{RMSE} = \frac{\sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(c_i - c_i^{\text{ref}})^2}}{\frac{1}{N}\sum_{i=0}^{N-1} c_i^{\text{ref}}} \tag{7}$$

A timeout of 120 s was applied to all runs. I have also tested if the run completes within 2 s, as a well-designed solution should comfortably run within a few milliseconds for each test case. One mark was deducted if the run time was longer than 2 s, although this mostly affected test 5, as the others used significantly smaller grid sizes.

Below, the low values of S and σ were set to 1×10^{-6}, not 0 as it would otherwise appear from the figures.

4.5.1 Analytical solution: 2 marks

Equations 6 on the $x \in [0, 1]$ domain are satisfied by the following solution functions

$$A(x) = 3 + \frac{1}{6}\cos 2\pi x + \frac{2}{3}\sin 2\pi x \tag{8}$$

$$B(x) = 1 + \frac{1}{6}\sin 2\pi x \tag{9}$$

if the following parameters are used:

$$\begin{aligned} D &= \frac{1}{4\pi^2} \\ v &= \frac{1}{2\pi} \\ k_+ &= 1 \\ k_- &= 1 \\ S(x) &= 2 + \cos 2\pi x + \sin 2\pi x \\ \sigma(x) &= 2 \end{aligned}$$

A small grid of $N = 100$ is sufficient to achieve accurate numerical results due to the smooth nature of the solutions. In my model solution, the RMSE between the numerical and analytical values were less than 0.3% - I used a tolerance of 1% to award the maximum 2 marks for this exercise.

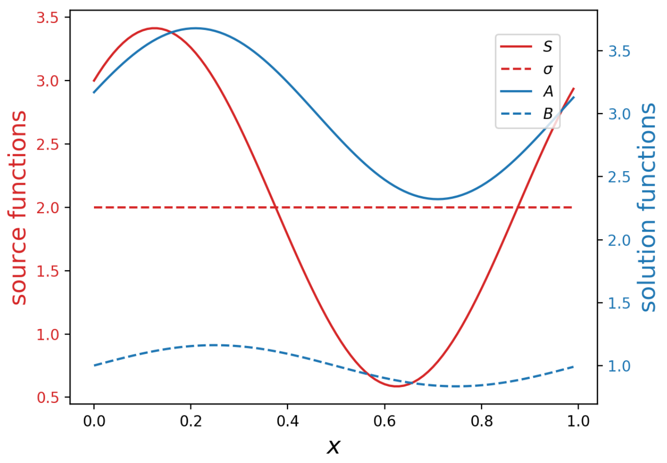

Figure 2: Source and solution functions in test 1.

4.5.2 Advection dominated case: 2 marks

In this test case the diffusion coefficient is very low and transport is dominated by advection. The source terms are rectangular functions.

4.5.3 Diffusion dominated case: 2 marks

In this test case the velocity of advection is very low and transport is dominated by diffusion.

4.5.4 Irreversible reaction case: 2 marks

In this test case the k_- is low, representing an almost irreversible reaction between A and B.

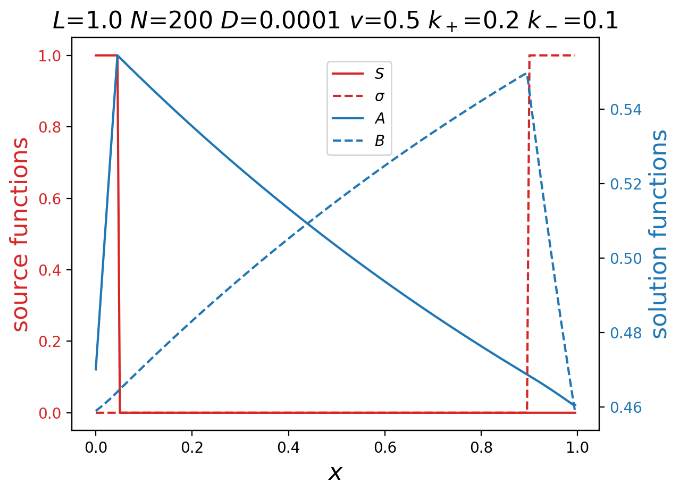

Figure 3: Source and solution functions in test 2.

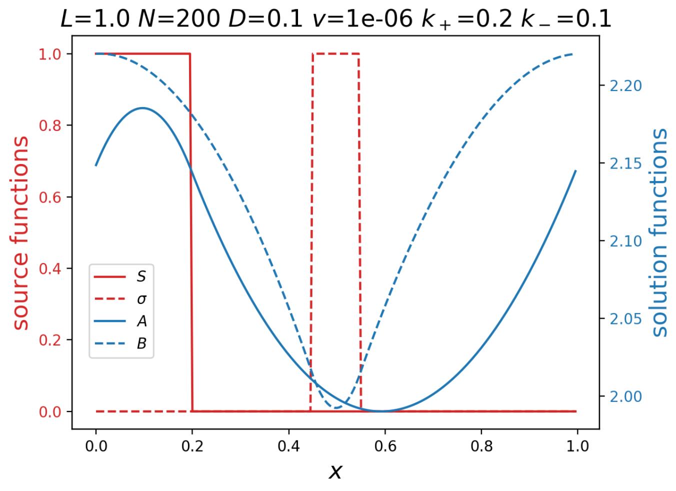

Figure 4: Source and solution functions in test 3.

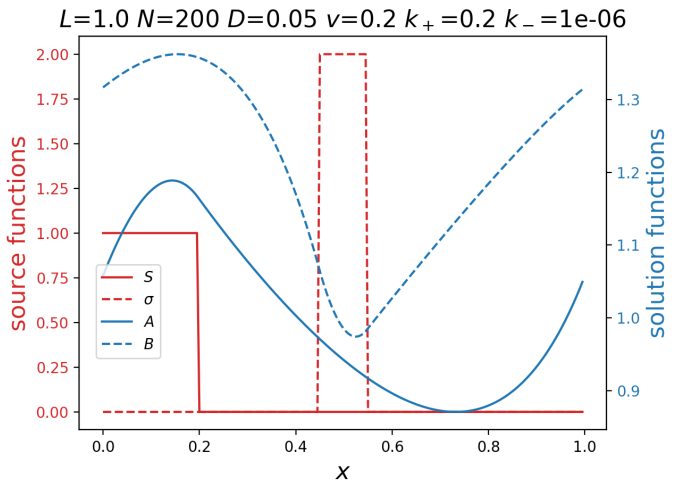

Figure 5: Source and solution functions in test 4.

4.5.5 Large grid case: 2 marks

In this test case the number of grid points N is high, as required by the narrow nature of the source terms. It tests the implementation's scalability. Solutions utilising a narrow banded matrix complete within a few milliseconds, whereas programs using what is essentially a full matrix will take around 4-6 s to run on `vonneumann` and over 30 s to run on `nenneke`.

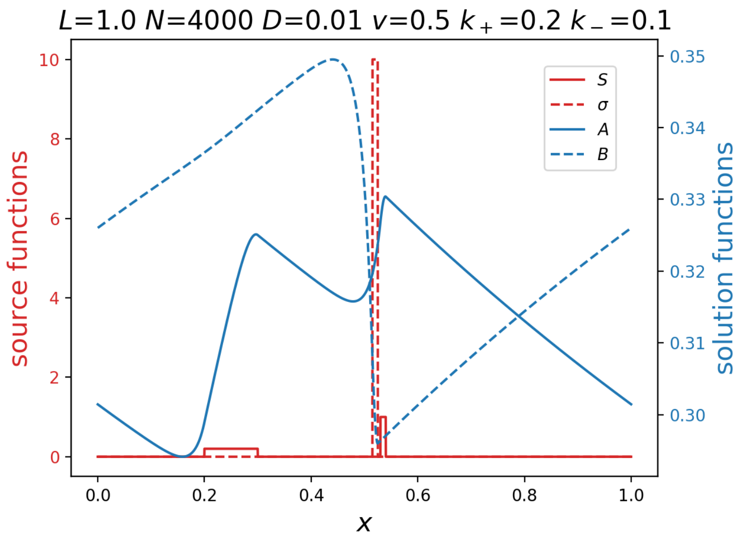

Figure 6: Source and solution functions in test 5.

5 Feedback

The problem can be solved by defining a banded $2N \times 2N$ system with the upper and lower band width of 4. After writing equations 6 in a discrete form, the trick was to solve for A and B simultaneously. A computationally efficient solution is to group the equations by grid points, and fold the grid points using the technique described in the lecture. I did not penalise less than optimal solutions, provided they ran in the allotted time.

Individual feedback was given as marks for each subsection, as described above, as well as comments. Most programs compiled, and most of you carefully eliminated compiler warnings. Common mistakes were:

- No validation of `malloc` calls.
- Output sent to the standard output rather than the specified `output.txt`.
- Some of you incorrectly defined L as an integer, causing the program to fail when reading the input parameters. I corrected this issue after deducting a mark.
- In some solutions the grid spacing was defined as $\Delta x = \frac{L}{N-1}$,which should have been simply $\Delta x = \frac{L}{N}$. The point $x = L$ is equivalent to $x = 0$, so there is no need to represent it. Accuracy, especially in the smaller grids, suffered.
- Some solutions did not change the band width definition of the matrix from 1, which unfortunately resulted in incorrect results, and in some test cases segmentation faults.
- In a few cases the band width was set to N (or a constant fraction of N), resulting in a correct solution that does not scale linearly with the grid size.
- If a sign was incorrectly implemented, that generally resulted in an inaccurate and incorrect solution. However, some test cases, notably 2, 3 and 4, were designed to "ignore" the signs of D, v and k_- respectively. This allowed recovering some marks for an almost, but not entirely, correct program.

- Depending on the approximation used for the first derivative:

$$\frac{f_i - f_{i-1}}{\Delta x} \quad \text{vs.} \quad \frac{f_{i+1} - f_i}{\Delta x} \quad \text{vs.} \quad \frac{f_{i+1} - f_{i-1}}{2\Delta x}$$

  results may differ slightly, but the difference is less than the 1% tolerance I used.

The most important piece of feedback is the need for *testing*. As demonstrated in test 1, it is not difficult to come up with simple, analytical source (and sink) terms that result in exact solutions for A and B. Solutions should be tested against such cases, highlighting most errors in the implementation. Even without an analytic solution, some basic tests should indicate if the solution makes sense.