Project 5, deadline December 9, 2016

Partial Differential equations

Fall semester 2016

Theoretical background and description of the system

This project is more numerically directed, with tests of algorithms for solving partial differential equations using finite difference schemes. Chapter 10 of the lecture notes provides the necessary theoretical background.

The physical problem can be that of the temperature gradient in a rod of length $L = 1$ or that of channel flow between two flat and infinite plates at $x = 0$ and $x = 1$, where the fluid is initially at rest and the plate $x = 1$ is given a sudden initial movement. We are looking at a one-dimensional problem

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, t > 0, x \in [0, L]$$

or

$$u_{xx} = u_t,$$

with initial conditions, i.e., the conditions at $t = 0$,

$$u(x, 0) = 0 \quad 0 < x < L$$

with $L = 1$ the length of the x-region of interest. The boundary conditions are

$$u(0, t) = 0 \quad t > 0,$$

and

$$u(L, t) = 1 \quad t > 0.$$

The function $u(x, t)$ can be the temperature gradient of a rod or represent the fluid velocity in a direction parallel to the plates, that is normal to the x-axis. In the latter case, for small t, only the part of the fluid close to the moving plate is set in significant motion, resulting in a thin boundary layer at $x = 1$. As time increases, the velocity approaches a linear variation with x. In this case, which can be derived from the incompressible Navier-Stokes, the above equations

constitute a model for studying friction between moving surfaces separated by a thin fluid film.

In this project we want to study the numerical stability of three methods for partial differential equations (PDEs). These methods we will discuss are the explicit forward Euler algorithm with discretized versions of time given by a forward formula and a centered difference in space resulting in

$$u_t \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t}$$

and

$$u_{xx} \approx \frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{\Delta x^2},$$

or

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2}.$$

We will also discuss the implicit Backward Euler with

$$u_t \approx \frac{u(x,t)-u(x,t-\Delta t)}{\Delta t} = \frac{u(x_i,t_j)-u(x_i,t_j-\Delta t)}{\Delta t}$$

and

$$u_{xx} \approx \frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{\Delta x^2},$$

or

$$u_{xx} \approx \frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2},$$

Finally we use the implicit Crank-Nicolson scheme with a time-centered scheme at $(x,t+\Delta t/2)$

$$u_t \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} = \frac{u(x_i,t_j+\Delta t)-u(x_i,t_j)}{\Delta t}.$$

The corresponding spatial second-order derivative reads

$$u_{xx} \approx \frac{1}{2}\left(\frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2} + \frac{u(x_i+\Delta x,t_j+\Delta t)-2u(x_i,t_j+\Delta t)+u(x_i-\Delta x,t_j+\Delta t)}{\Delta x^2}\right).$$

Note well that we are using a time-centered scheme wih $t+\Delta t/2$ as center.

For this project you can collaborate with fellow students and you can hand in a common report. This project (together with projects 3 and 4) counts 1/3 of the final mark.

Project 5a): Setting up the algorithms. Write down the algorithms for these three methods and the equations you need to implement. For the implicit schemes show that the equations lead to a tridiagonal matrix system for the new values.

Project 5b): Truncation errors and analytic solutions. Find the truncation errors of these three schemes and investigate their stability properties. Find also the analytic solution to the continuous problem.

Project 5c): Implementation. Implement the three algorithms in the same code and perform tests of the solution for these three approaches for $\Delta x = 1/10$, $h = 1/100$ using Δt as dictated by the stability limit of the explicit scheme. Study the solutions at two time points t_1 and t_2 where $u(x, t_1)$ is smooth but still significantly curved and $u(x, t_2)$ is almost linear, close to the stationary state. Remember that forsolving the tridiagonal equations you can use your code from project 1.

Project 5d): Compare the solutions at t_1 and t_2 with the analytic result for the continuous problem. Which of the schemes would you classify as the best?

Project 5e): Moving to two dimensions. Extend the code you have developed here to two dimensions. It means that we deal with a $2 + 1$ dimensional problem. Our differential equation becomes

$$\frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2} = \frac{\partial u(x,y,t)}{\partial t}, t > 0, x, y \in [0,1],$$

where we now have made a model with a square lattice for x and y. How would you extend the boundary conditions from one dimension to two dimensions? And can you find a closed form solution here as well? It is left to you to decide upon what kind of boundary conditions you deem appropriate.

Project 5f): Solving the two-dimensional equations numerically. Here you can choose between an explicit or an implicit scheme. For the implicit scheme discussed in chapter 10 of the lecture notes, upi need to use for example an iterative method like Jacobi's.

Outline the algorithm for solving the two-dimensional diffusion equation and implement the explicit or the implicit scheme as function of Δx (assuming $\Delta x = \Delta y$) and Δt. Solve the equations numerically and give a critical discussion of your results. Compare your results with the closed-form answer. Discuss the stability of the solution as function of different values of Δx and Δt.

You should, if you have time, try to parallelize the equations using MPI or OpenMP.

References. A very good reference is the textbook by Winther and Tveito on partial differential equations. It is available online from the University library.

Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.
- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.
- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.
- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats or as an ipython notebook file. As programming language we prefer that you choose between C/C++, Fortran2008 or Python. The following prescription should be followed when preparing the report:

- Use Devilry to hand in your projects, log in at `http://devilry.ifi.uio.no` with your normal UiO username and password and choose either 'fys3150' or 'fys4150'. There you can load up the files within the deadline.

- Upload **only** the report file! For the source code file(s) you have developed please provide us with your link to your github domain. The report file should include all of your discussions and a list of the codes you have developed. Do not include library files which are available at the course homepage, unless you have made specific changes to them.
- In your git repository, please include a folder which contains selected results. These can be in the form of output from your code for a selected set of runs and input parametxers.
- In this and all later projects, you should include tests (for example unit tests) of your code(s).
- Comments from us on your projects, approval or not, corrections to be made etc can be found under your Devilry domain and are only visible to you and the teachers of the course.

Finally, we encourage you to work two and two together. Optimal working groups consist of 2-3 students. For this specific report you can hand in a common report.