TDT4200 Fall 2020 - Problem Set 0 (Optional Extra)

Introduction to C

Jacob Odgård Tørring og Anne C. Elster

Deadline: September 6, 2022 by 22:00 in BlackBoard
Evaluation: None

- **This assignment is optional and will not be evaluated.**
- **This assignment has one part.** It consists of writing a serial version of the Finite Difference Method (FDM) for solving the Shallow Water Equations.
- **Solution requirements** are stated in the Evaluation section.
- **Do not deliver any other files than those specified in the Delivery section.** Code should not use external dependencies aside from the ones already included in the handout code.

1 Finite difference approximation of the 1D shallow water equations

In this assignment you will write a serial implementation of the Finite Difference Method (FDM) for solving the 1D shallow water equations. This serial implementation will be the baseline for future problem sets. Information on solving the shallow water equations using FDM is described in the lecture slides.
The shallow-water equations describe a thin layer of fluid where we can consider the density to be constant and in hydrostatic balance. The system is bounded from below by the bottom topography and from above by a free surface. One application of the shallow water equations is that they can describe the propagation of a tsunami accurately until the wave approaches the shore. Near the shore, a more complicated model is required. Similarly, it can also model mud slides and splashes in a tub.

The 1D Finite difference equations. See the slides from our presentation for more details.

$$\frac{\delta}{\delta t}\rho\eta + \frac{\delta}{\delta x}\rho\eta u = 0 \tag{1}$$

$$\frac{\delta}{\delta t}\rho\eta u + \frac{\delta}{\delta x}(\rho\eta u^2 + \frac{1}{2}g\rho\eta^2) = 0 \tag{2}$$

This can be reformulated to the following equations.

$$(\rho\eta)_i^{n+1} = \frac{1}{2}[(\rho\eta)_{i+1}^n + (\rho\eta)_{i-1}^n] - \frac{\Delta t}{2\Delta x}[(\rho\eta u)_{i+1}^n - (\rho\eta u)_{i-1}^n] \tag{3}$$

$$(\rho\eta u)_i^{n+1} = \frac{1}{2}[(\rho\eta u)_{i+1}^n + (\rho\eta u)_{i-1}^n] - \frac{\Delta t}{2\Delta x}[(du)_{i+1}^n - (du)_{i-1}^n] \tag{4}$$

$$(du)_i^n = \rho\eta u^2 + \frac{1}{2}g\rho\eta^2 = (\rho\eta)_i^n(u_i^n)^2 + \frac{1}{2}g\frac{((\rho\eta)_i^n)^2)}{\rho} \tag{5}$$

$$(u)_i^n = \frac{(\rho\eta u)_i^n}{(\rho\eta)_i^n} \tag{6}$$

Which together with these boundary conditions, provides the information necessary to implement the Shallow Water Equations using the Finite Difference Method.

$$(\rho\eta)_{max+1}^n = (\rho\eta)_{max-1}^n \tag{7}$$

$$(\rho\eta)_0^n = (\rho\eta)_2^n \tag{8}$$

$$(\rho\eta u)_0^n = -(\rho\eta u)_2^n \tag{9}$$

$$(\rho\eta u)_{max+1}^n = -(\rho\eta u)_{max-1}^n \tag{10}$$

Please see slides on this topic for further information. The initial values for the problem have already been implemented in the handout code. The skeleton for your serial implementation can be found in `shallow_water_serial.c`.

Program description

Information on how to install the dependencies and run the program is provided in the `README.md` of the handout code.

Although we provide a Makefile in the handout code, we encourage you to inspect the Makefile to see how to compile and run the program.

Tasks

The following tasks all have corresponding TODO-comments in `shallow_water_serial.c`. We recommend that you ensure the program compiles after completing a task even though the program might not run correctly between tasks.

1. Get the command-line arguments from the OPTIONS struct and store the values in the corresponding global variables: N, max_iterations and snapshot_frequency.
2. Allocate memory on the heap for the mass, mass-velocity, velocity and acceleration arrays with enough space for N+2 real_t elements.
3. Update the boundary conditions and the domain variables for each time step
   a. Update the edges of the domain for $\rho\eta$ and $\rho\eta u$ based on the boundary conditions.
   b. Update the acceleration over the entire domain and the borders.
   c. Update the $\rho\eta$-velocity of the next time step over the entire domain.
   d. Update the $\rho\eta$ of the next time step over the entire domain.
   e. Update the velocity of the next time step over the entire domain.
4. Implement the swap function to swap two arrays.
5. Free the heap-allocated memory for the program.

Evaluation

The code should run error-free and produce correct output. Your solution can be checked for correctness with the correctness script in the handout code.

Delivery

Deliver the file `shallow_water_serial.c` on BlackBoard. Do **NOT** upload a ZIP file.