WSE-HI- Module 5 - Numerical methods part 2

Free surface flow
(2020)

Ioana Popescu
Juan Carlos Chacon Hurtado

(February 2020)

Numerical methods – part II

1. Introduction

The objective of this exercise is to obtain experience in setting up a mathematical model of a hydraulic phenomenon, to test its performance and to report upon the results. In particular, the case of flow in a channel with a rectangular cross-section and a constant bottom slope will be considered. This model can then form the basis for future extensions to include irregular topographies, variable resistance coefficients and systems of interconnected channels.

Explanation on the work for the exercises

During these sessions of Numerical Methods 2, you will elaborate on the solution of Saint Venant Equations using Preissmann scheme.

The work is carried out in steps, with two main assignments which are described below.

Step 1: The Preismann scheme implementation will be detailed in class, and it is not part of the assignment.
Step 2: The code that implements the solution of Saint Venant equations using Preismann scheme will be developed in class using Python.

> ***Assignment 1***: The developed code will be submitted along with the corresponding pseudocode, as a separate word file, showing which part of your python code corresponds to specific parts of your pseudocode.
> Due date for submission: 19 February 18:00 o'clock (on eCampusXL)

Step 3: The code developed during class will be tested for a number of flow cases (described below). Each one of you will have his own data.

> ***Assignment 2:*** The code will be tested with a number of flow cases (described below). Each one of you will have his own data. The test results and the interpretation you made of them will have to be reported and submitted in the form of a report. Use graphs and plots to present your results in the form of longitudinal profiles and time series. Based on these graphs comment/analyse the results. It can be foreseen that not all tests will be successful. In case of failure you will still have to comment/provide the reasons for this failure (numerical, physical, etc).
> Due date for submission: 1st March 13:00 o'clock (on eCampusXL)

Important remarks

The quality of the assignment report is also part of the assignment mark. A good quality report means that:

- you explain what you do, make short and clear sentences
- you do not put tables with results in report, unless strictly necessary, you only use graphs to present and analyse results. The calculated values will be checked with the help of the submitted code
- you present clean and readable graphs, that you indicate on them *the variables and their units*; it also means that the *x*- and *y*- axes are scaled in such a way that the variations of the lines are visible.

The reports have to be written using a word processor. Submitted version has to be in pdf format. Hand-written reports will not be considered for marking.

2. *Free surface flow Governing Equations and their discretisation using Preissmann scheme*

2.1. Governing equations

For channels of irregular topographies, the flow can be described by the de Saint Venant equations:

$$\frac{\partial Q}{\partial x} + b\ \frac{\partial h}{\partial t} = 0 \qquad \text{continuity (2.1)}$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\beta \frac{Q^2}{A}\right) + gA\frac{\partial h}{\partial x} - gAS_o + gA\frac{|Q|Q}{K^2} = 0 \qquad \text{momentum (2.2)}$$

where:

Q = discharge	(m^3/s)
h= water depth	(m)
b= storage width	(m)
A= cross-sectional area	(m^2)
K= conveyance	(m^3/s)
C= Chezy resistance coefficient	($m^{0.5}/s$)
R= hydraulic radius	(m)
β= Boussinesq coefficient	(-) (we set it to be equal to 1)
g= gravity	(m/s^2)
So=bed slope	(-)

For this exercise, the following simplifications shall be made:

b = **constant** channel width
$R = h$ = **wide channel** approximation
So = bed slope $= -\dfrac{\partial H}{\partial x}$ = **constant**

2.2. Finite Difference Approximation using Preismann scheme

These equations are to be solved using a finite difference approximation of Preissmann. A grid is introduced in time and space where, at every gridpoint, h and Q are defined as the unknown variables. Between two successive gridpoints both the continuity and the momentum equations are applied (Fig. 1).

Together with the boundary conditions, a sufficient number of equations is obtained to solve Q and h at every gridpoint.

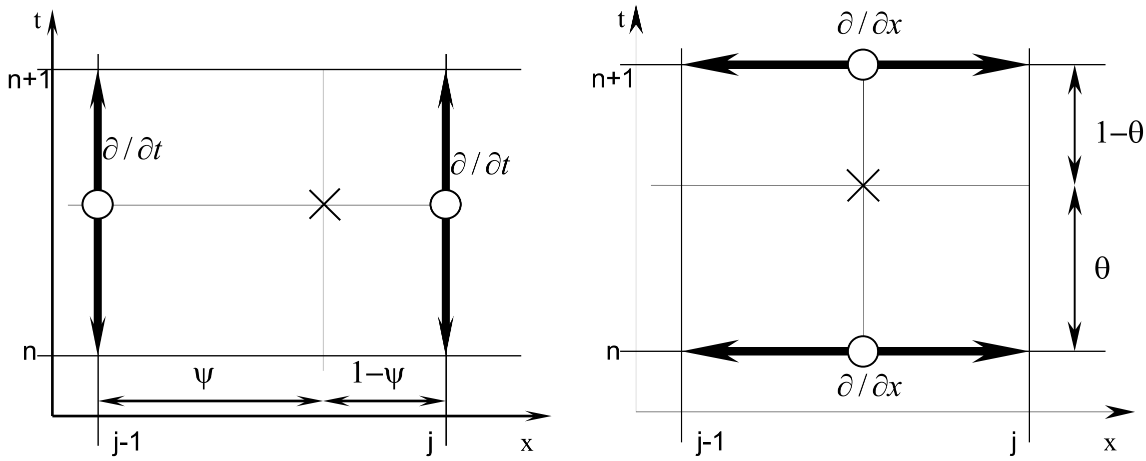


Figure 1. Operator for finite difference scheme of the Preissmann type

Discretisations for equation (2.1), using the Preissmann scheme is:

$$\frac{\partial Q}{\partial x} = (1-\theta)\left[\frac{Q_{j+1} - Q_j}{\Delta x}\right]^n + \theta\left[\frac{Q_{j+1} - Q_j}{\Delta x}\right]^{n+1}$$

$$b\ \frac{\partial h}{\partial t} = b_{j+\frac{1}{2}}^{n+\frac{1}{2}}\left((1-\psi)\left[\frac{h^{n+1} - h^n}{\Delta t}\right]_j + \psi\left[\frac{h^{n+1} - h^n}{\Delta t}\right]_{j+1}\right)$$

$$\frac{\partial Q}{\partial t} = (1-\psi)\left[\frac{Q^{n+1} - Q^n}{\Delta t}\right]_j + \psi\left[\frac{Q^{n+1} - Q^n}{\Delta t}\right]_{j+1}$$

After introducing these terms into equation (3.1) the general form of the equations is:

$$A1_j\, Q_j^{n+1} + B1_j\, h_j^{n+1} + C1_j\, Q_{j+1}^{n+1} + D1_j\, h_{j+1}^{n+1} = E1_j \qquad (2.3)$$

A similar equation can be obtained for equation (2.2)

The operator can be applied twice at each of the internal gridpoints. Together with an upstream and a downstream boundary condition this forms a complete set of simultaneous equations of the form:

$$\mathbf{A}\ z = \mathbf{B} \qquad (2.4)$$

There are two possible ways of solving this system:

1. use the double-sweep algorithm when a standard programming language is used (e.g. Python, MATLAB, PASCAL, FORTRAN, C++, etc.)
2. use a built-in matrix inversion function when using a more elaborate programming language (e.g. Python, MATLAB), where the matrix **A** can be inverted directly.

Remark: you will be asked to code the second solution (use of built-in matrix inversion solution). Double sweep algorithm is considered ***as bonus, if chosen***.

DOUBLE SWEEP ALGORITHM:

The linear system of equations to be solved is of the form:

$$A1_j Q_j^{n+1} + B1_j h_j^{n+1} + C1_j Q_{j+1}^{n+1} + D1_j h_{j+1}^{n+1} = E1_j \tag{2.5}$$

$$A2_j Q_j^{n+1} + B2_j h_j^{n+1} + C2_j Q_{j+1}^{n+1} + D2_j h_{j+1}^{n+1} = E2_j \tag{2.6}$$

In case a double sweep algorithm is used for the final solution of this system the following recurrence relation are introduced:

$$Q_j^{n+1} = F_j h_j^{n+1} + G_j \tag{2.7}$$

$$h_j^{n+1} = H_j Q_{j+1}^{n+1} + I_j h_{j+1}^{n+1} + J_j \tag{2.8}$$

With the value for the coefficients as:

$$H_j = \frac{-C1_j}{A1_j F_j + B1_j}$$

$$I_j = \frac{-D1_j}{A1_j F_j + B1_j}$$

$$J_j = \frac{E1_j - A1_j G_j}{A1_j F_j + B1_j}$$

$$F_{j+1} = \frac{D2_j\left(A1_j F_j + B1_j\right) - D1_j\left(A2_j F_j + B2_j\right)}{C1_j\left(A2_j F_j + B2_j\right) - C2_j\left(A1_j F_j + B1_j\right)}$$

$$= \frac{-I_j\left(A2_j F_j + B2_j\right) - D2_j}{H_j\left(A2_j F_j + B2_j\right) + C2_j}$$

$$G_{j+1} = \frac{\left(E1_j - A1_j G_j\right)\left(A2_j F_j + B2_j\right) - \left(E2_j - A2_j G_j\right)\left(A1_j F_j + B1_j\right)}{C1_j\left(A2_j F_j + B2_j\right) - C2_j\left(A1_j F_j + B1_j\right)}$$

$$= \frac{E2_j - A2_j F_j J_j - A2_j G_j - B2_j J_j}{H_j\left(A2_j F_j + B2_j\right) + C2_j}$$

Boundary conditions are analysed as follows:

-Upstream BC

If $\boldsymbol{Q_0}$ given

$F_0 = 0$

$G_0 = \boldsymbol{Q_0}$

If $\boldsymbol{h_0}$ given:

$F_0 = \infty$ (say $10^5 - 10^6$)

$G_0 = -F_0 \boldsymbol{h_0}$

or:

$$F_1 = \frac{A1_0 D2_0 - A2_0 D1_0}{A2_0 C1_0 - A1_0 C2_0}$$

$$G_1 = \frac{A2_0 E1_0 - A1_0 E2_0 - \boldsymbol{h_0}\left(A2_0 B1_0 - A1_0 B2_0\right)}{A2_0 C1_0 - A1_0 C2_0}$$

- Downstream BC

  If Q prescribed then

$$Q_N^{n+1} = Q_{boundary} \qquad h_N^{n+1} = (Q_N^{n+1} - G_N)/F_N$$

  If h prescribed then

$$h_N^{n+1} = h_{boundary} \qquad Q_N^{n+1} = F_j h_j^{n+1} + G_j$$

3. Model testing

Model performance will be tested for the following 3 cases:

static test:	where the fluid is initially at rest and must remain at rest for a sufficiently long period of simulation (e.g. > 10000 timesteps)
steady-state test:	where the fluid is initially flowing in a steady state and the boundary conditions remain unchanged.
transient:	where the fluid is initially flowing in a steady state and the boundary conditions are subjected to a sudden change

Optionally the performance can be as well tested for the following case:

seiche test	where the fluid is in an initial, non-steady state variation of water levels and/or flow rates and the boundary are kept zero in time.

For each of these tests the solution is known. Run your code and compare the obtained results with your expectations.

The data for tests will be handed in class.

4. Code specifications

The designed computer program should solve the free surface flow problem represented by equations (2.1) and (2.2).
The code will be made such that any user can input data, hence code will read the following data from input files:

- time-varying water levels and discharges at any of the boundaries (h or Q, upstream or downstream)
- initial water levels and discharges at all points of the model (since you may start from non-constant conditions in space)
- geometry of the canal
- numerical parameters

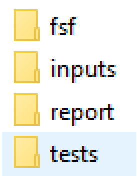
Designing of these files it is part of the Assignment 1, i.e your choice on how to do that.
Avoid the use of constants for input data within the main program.

Results of the running of any case using your code should be saved in output file(s). Think carefully about the presentation of your results for report when designing the out file(s). You should be able to produce graphs for your report of time series in Q and/or h for any chosen grid point, and longitudinal profiles for any chosen time level.

It is required that alongside the code, the following functions are submitted to test the assignment:

- a function named **test_h0**. This function takes as an argument only one integer value (i.e.1,2,3,…) indicating one of the case that you run the model for. The function returns a vector containing the water depth at the left boundary of the channel, for all time steps of computation.
- a function named **test_q0**. This function takes as an argument only one integer value (i.e.1,2,3,…) indicating one of the case that you run the model for. The function returns a vector containing the discharge at the left boundary of the channel, for all time steps of computation.
- a function named **test_h_end**. This function takes as an argument only one integer value (i.e.1,2,3,…) indicating one of the case that you run the model for. The function returns a vector containing the water depth at the right boundary of the channel, for all time steps of computation.
- a function named **test_q_end.** This function takes as an argument only one integer value (i.e.1,2,3,…) indicating one of the case that you run the model for. The function returns a vector containing the discharge at the right boundary of the channel, for all time steps of computation.

In addition, the code should be organised as follows:



The folder "**fsf**" should contain the implementation of the engine to solve the free surface flow equations. The folder "**inputs"** should contain all the input files used to run your cases and tests. Pay attention on the name you give to these files. The "**report"** folder should contain all the model output files that contain data based on which the graph results were build for your report. The "**tests"** folder should contain the testing functions described above. These functions are compulsory tasks of your coding assignment.
All 4 folders described are compulsory. Apart of these 4 folders you can however, add any additional folders (or sub-folders) that you consider necessary.

<u>Recommendations</u>:

- Store the computational results at the end of every time step — do not wait for the simulation to be terminated to dump the results into a file: in case of a bug that causes floating-point errors, you will lose all the results of the computation;
- Do not program the name of the input and output files directly into the code, but give the user the possibility to read the data from and to store the results into files that have different names.
- Make sure to use relative references when calling a file from within your code. Relative references will ensure that the code will be correctly executed in other PC's.

- Programming debug information (catching errors within the code execution), will help you understand better the reasons why the program may fail.
- Breaking down your code into functional (and testable) pieces will help you implement and debug your implementation.

References

Abbott, M.B. and Minns A.W., 1997, *Computational Hydraulics*,2nd ed., Ashgate, U.K.

Abbott, M.B. and Basco, D.R., 1989, *Computational Fluid Dynamics - An Introduction for Engineers*, Longman Scientific & Technical, U.K.

Abbott, M.B. and Verwey, A., 1970, Four point method of characteristics, *Proc. ASCE*, 96, No. HY12.

Cunge, J.A., Holly, F.M. and Verwey, A., 1980, *Practical Aspects of Computational River Hydraulics*, reprinted by Institute of Hydraulic Research, Univ. of Iowa, Iowa City, U.S.A., 1986.

Evans, E.P., 1977, The behaviour of a mathematical model in open channel flow, *Proc. 17th Congress IAHR,* Paper A97, Baden-Baden, pp 173-180.

Mahmood, K. and Yevyevich, V. (Editors), 1975, *Unsteady Flow in Open Channels*, 3 Volumes, Water Resources Publications, Fort Collins.

Popescu, I, 2014, Computational Hydraulics, IWA Publishing

Verwey, A., 1975, On external and internal boundary conditions and their influence on net flow and mean water levels in systems of channels, *Proc. 16th Congress IAHR*, Sao-Paulo.

Annex1: Student set of data

Student no	Student name	Data set number
1008784		1
1041829		16
1042957		3
1052068		19
1057438		5
1057883		6
1060031		14
1061032		8
1062906		9
1064735		13
1067143		11
1067911		12
1068076		10
1068292		7
1068411		15
1069406		2
1069431		17
1069728		18
1070284		4
1071842		20