Indian Institute of Technology
Mandi

School of Engineering

DP402P : Project Report

PySWE -A Python-based solver for Shallow Water Equations

by
B17130 (Nabeel Khan)

Mentor
Dr. Gaurav Bhutani

Abstract

A Python-based solver was developed for solving 2D shallow water equations (SWE). The accuracy of the solver was computed and the results were further compared with the existing analytical solution for dam break problem and flow on parabolic surface. Finite volume method (FVM) is used to discretizing the equations on a structured mesh due to the ease of handling conservative equations of fluid flow by FVM. A first order Lax-Friedrich spatial scheme and first order explicit transient scheme are implemented in the numerical solver because of the simplicity. To increase the computational speed of this solver, an adaptive time stepping approach is used which automatically modify the size of time step based on the Courant-Friedrichs-Lewy (CFL) condition. Two cases were studied for verifying numerical results – pseudo-2D dam break flow on a flat surface and flow of water on parabolic surface. The free surface height h was compared for the both the cases. Advantages and limitations of the PySWE are discussed in this report. The computational time of our solver was compared with the OpenFoam code – a open source C++ based solver – and the result between the mesh number and time was plotted. The developed solver easily handles source term in the equation without giving oscillatory solution which is a common problem while solving SWE. The solver is available on Github for further open source development. The ease of understanding of the code structure was the one of the main focus of this project, so that anyone new to this field can easily understand the project and help it grow significantly. The importance of Python and open-source contribution is highlighted through this project.

Contents

1 Introduction

Open-channel flows like oceans tides, avalanches, river and channels flows are abundant in nature. Studying these flows help us prepare for potential disasters associated with such flows through prediction of associated flow fields. These flows are also used in a lot of industrial processes and modelling these flow leads to better optimization of these processes. These flows are governed by a set of hyperbolic partial differential equations i.e shallow water equations (SWE) or Saint-Venant Equations, which are simplified version of Navier Stoke Equations. Shallow water equations govern the flow of a thin layer fluid of constant density over a topography with free surface. These equations are also referred as depth-average or depth-integrated NSE. However, these equations are only valid when the ratio of wave amplitude and wavelength is of small value [3]. Solving SWE are much computational economical than solving NSE for the same problem study because it reduces the dependency on vertical dimension and simplifies pressure term, thus reduces one computational dimension from the problem, for instance, a 2-D dam flow can be solved on 1-D computational domain, hence saving both computational time and cost without sacrificing accuracy, as shown by Akhil Akkapelli and Dr. Gaurav Bhutani[1].
These equations are mostly solved to study the evolution of the surface height of the flow. In this project, 2-D SWE without surface friction are solved numerically on both flat surface and curved surface.

1.1 Shallow water equations

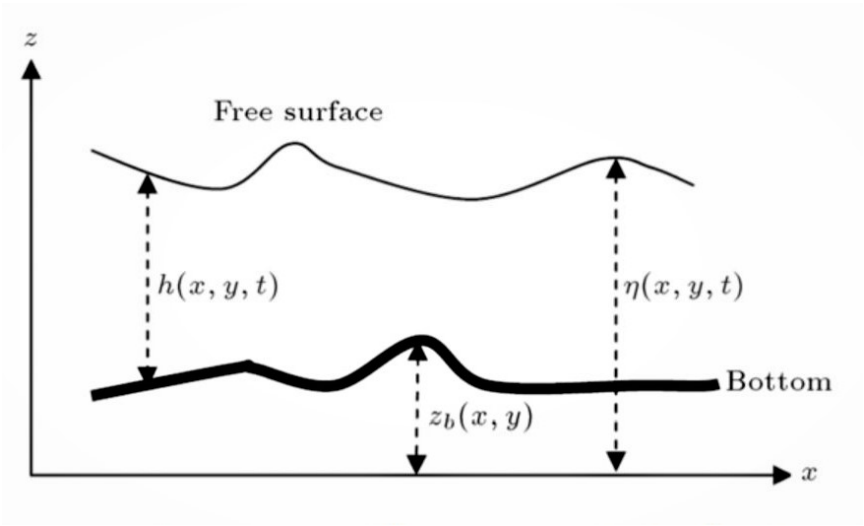


Figure 1: Diagram depicting shallow water equation terms

The 2D shallow water equations for a frictionless bottom surface in conservation form are written as follows:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial y} = \mathbf{S}(\mathbf{U}) \tag{1}$$

Where $\mathbf{U}$ is the vector of conserved flow variables, $\mathbf{F}\mathrm{F}(\mathbf{U})$ and $\mathbf{G}(\mathbf{U})$ is the vector of flux in x and y direction respectively and $\mathbf{S}(\mathbf{U})$ is the source term, and each are defined as follows in vector form:

$$\begin{aligned} \mathbf{U} &= \begin{bmatrix} h \\ h\bar{u} \\ h\bar{v} \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} h\bar{u} \\ h\bar{u}^2 + \frac{1}{2}gh^2 \\ h\bar{u}\bar{v} \end{bmatrix} \\ \mathbf{G}(\mathbf{U}) &= \begin{bmatrix} h\bar{v} \\ h\bar{u}\bar{v} \\ h\bar{v}^2 + \frac{1}{2}gh^2 \end{bmatrix}, \quad \mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ ghS_x \\ ghS_y \end{bmatrix}. \end{aligned} \tag{2}$$

In the above expressions; h is the water depth, $\bar{u}$ is the depth-averaged velocity along the x-direction, $\bar{v}$ is the depth-averaged velocity along y-direction and g is the gravitational acceleration. The basal slope in x and y directions, S_x and S_y, are given by

$$S_x = -\frac{\partial z_b}{\partial x}, \quad S_y = -\frac{\partial z_b}{\partial y} \tag{3}$$

And depth-average velocities are given by integrating the velocity in the vertical direction

$$\bar{u} = \frac{1}{h}\int_{z_b}^{\eta} u dz, \quad \bar{v} = \frac{1}{h}\int_{z_b}^{\eta} v dz \tag{4}$$

You can also include effect of friction in SWE by including its contribution in the source vector S(U) as sink. This is done by taking friction slope which can further be modelled using *Manning resistance law*.

2 Objectives

- The main objective of this project is to develop a Python-based solver for 2-D shallow water equation and host this project on Github. The motivation behind using Python is the simplicity of the language as compared to using fast computing languages like FORTRAN or C++ which has been used in popular open source fluid flow solvers. That is why a lot of focus is given on the code structure, so that anyone can easily understand this solver and start contributing to its new feature. And there is a very vast Python development community, so we can expect the expansion of the present solver at faster rate.
- To implement software development feature in the solver like Object Oriented Programming and creating simulation checkpoints.
- To create separate modules for each steps in numerical methods like Meshing, Discretization, Solver and Boundary condition etc. Having separate module file make it easier to add new features in future.

3 Numerical method for SWE

A finite volume-based approach was implemented in the solver to discretize SWE partial derivates equations because of conservative nature it provides. Lets consider a control volume as show in figure 2. To discretize, first integrate whole equation over volume of given control volume.

$$\iint_V \partial_t \mathbf{U}\, dx\, dy + \iint_V \partial_x \mathbf{F}(\mathbf{U})\, dx\, dy + \iint_V \partial_y \mathbf{G}(\mathbf{U})\, dx\, dy + \iint_V \mathbf{S}(\mathbf{U})\, dx\, dy = 0, \quad (5)$$

$$\partial_t \mathbf{U} + \frac{\mathbf{F}(\mathbf{U}_{(i+1/2,j,t)}) - \mathbf{F}(\mathbf{U}_{(i-1/2,j,t)})}{\Delta x} + \frac{\mathbf{G}(\mathbf{U}_{(i,j+1/2,t)}) - \mathbf{F}(\mathbf{U}_{(i,j-1/2,t)})}{\Delta y} + \mathbf{S}_{(i,j,t)} = 0 \quad (6)$$

Where Δx is the control volume size in x direction, Δy is the control volume size in y direction, subscripts i,j refer to the mesh grid location. Now after integrating this equation against time variable from t to t+Δt and implementing first order explicit scheme, we obtain

$$\frac{\mathbf{U}^{n+1}_{(i,j)} - \mathbf{U}^{n}_{(i,j)}}{\Delta t} = -\frac{\mathbf{F}_{(i+(1/2),j)} - \mathbf{F}_{(i-(1/2),j)}}{\Delta x} - \frac{\mathbf{G}_{(i,j+(1/2))} - \mathbf{G}_{(i,j-(1/2))}}{\Delta y} + \mathbf{S}_{(i,j)} \quad (7)$$

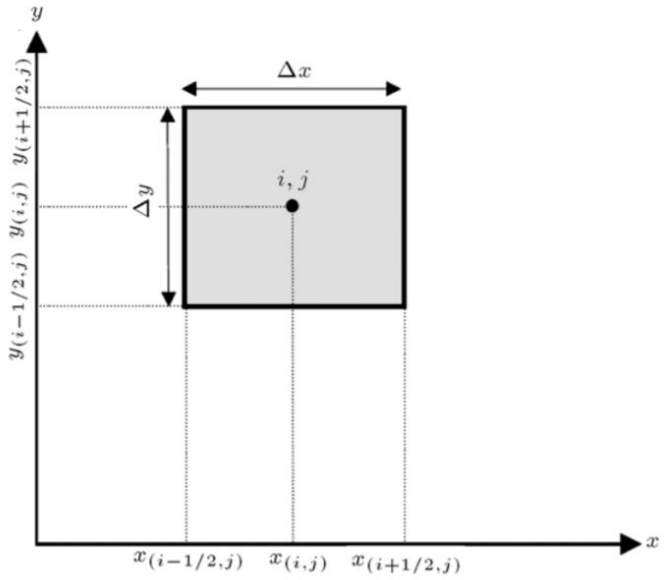


Figure 2: Control volume for 2D shallow water equation in x-y field

Here Δt is time step, superscript n and n + 1 refer to current and next time levels, respectively. And all the values of the variables in right hand side of equation are known.

We could implement various transient scheme like implicit, Runge–Kutta or Crank-Nicolson but we have used first-order explicit scheme because of the simplicity it provide while solving these set of linear algebraic equation.
This equation (7) is sometime referred as semi-discretized equation. And evaluating these interference fluxes are very important because finite volume method is based on equating fluxes over these discrete control volumes. This step strongly influences accuracy, stability, and order of coverage of numerical solution. There are various schemes developed for solving SWE. But most numerical solution of SWE faces problem of instability, oscillation due to hyperbolic nature of the partial differential equation which result in wave characteristic solutions. Often numerical scheme give rise to oscillatory steady solutions and there is difficulty in solving for dry condition where h=0, that is why solving these equations poses several difficulties. Numerical discretization of SWE also give rise to Riemann problem and solving Riemann problem is quite complex and computational expensive.
There are various schemes which are based on Riemann problem, one of them is Gondunov scheme that takes advantage of piecewise construction of all the variable over whole domain. Kurganov and Petrova proposed a well balanced second order central upwind scheme for SWE, which handle the dry condition very vell without giving negative solutions [2] . There is another simplified first order scheme based on Gondunov scheme known as Monotonic Upstream Scheme of Conservation Law (MUSCL), it also based on piecewise construction as proposed by S.M. Amiri [4]. But the only problem with these two schemes is the complexity and computational expensive. Although the 2nd order scheme will converge to exact solution faster with respect to the mesh size, but it will be computational heavy and slow due to increased operations.
That is why we have implement a simple and less computationally expensive scheme in our solver. We used Lax-Friedrich Scheme with modification to make it stable at steady state solution. The interference fluxes in x-direction and y-direction are given by

$$\begin{aligned}\mathbf{F}_{(i+(1/2),j)} &= \frac{1}{2}\left(\mathbf{F}\left(\mathbf{U}_{(i+1,j)}\right)+\mathbf{F}\left(\mathbf{U}_{(i,j)}\right)\right)-\frac{1}{2}\frac{\Delta x}{\Delta t}\left(\mathbf{U}_{(i+1,j)}-\mathbf{U}_{(i,j)}\right),\\ \mathbf{G}_{(i+(1/2),j)} &= \frac{1}{2}\left(\mathbf{F}\left(\mathbf{U}_{(i,j+1)}\right)+\mathbf{F}\left(\mathbf{U}_{(i,j)}\right)\right)-\frac{1}{2}\frac{\Delta x}{\Delta t}\left(\mathbf{U}_{(i,j+1)}-\mathbf{U}_{(i,j)}\right)\end{aligned} \tag{8}$$

The source vector is discretized using Taylors expension.

$$\mathbf{S}_{(i,j)} = \begin{bmatrix} 0 \\ -gh_{(i,j)}\left(\frac{z_{b(i+1,j)}-z_b(i,j)}{\Delta x}\right) \\ -gh_{(i,j)}\left(\frac{z_b(i,j+1))-z_b(i,j)}{\Delta y}\right) \end{bmatrix} \tag{9}$$

Lax Friedrichs also proposed a modification in dealing with $U^n_{i,j}$, to reduce the oscillation occur in numerical solution in equation without compromising the accuracy.

$$U^n_{i,j} = \frac{1}{4}\left[U^n_{i-1,j}+U^n_{i+1,j}+U^n_{i,j+1}+U^n_{i,j-1}\right] \tag{10}$$

The Lax-Friedrichs scheme is first order in nature for both space in time. And due to its

explicit nature, CFL condition must be satisfied every instant. If not, the solution can show instability and unrealistic solutions.

4 Results

There are no general solutions for 2D SWE due to non-linearity exist in the hyperbolic partial derivative equations. However, William Carlisle Thacker solved analytically SWE on a parabolic 3D profile with frictionless base with certain initial condition. And this solution was also used to verify result of our solver. Due to unavailability of solutions and literature, we also used psedo-2D dam break problem to verify results. We have compared the free surface height with the analytical solutions.

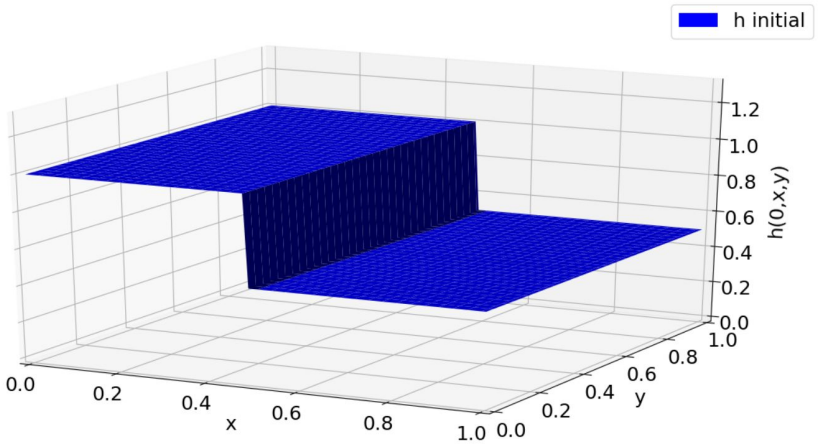


Figure 3: Initial condition for pseudo-2D dam break problem

4.1 Case 1: Dam break problem

Nature of pseudo-2D problems are between 2D and 1D. We have numerically solved a 2D dam break study which can be modelled with 1D analytical solution.
No source is present and friction at base is also neglected for this case. The initial conditions for the test problem as follows:

$$u(x,y,0) = 0, v(x,y,0) = 0 \text{ and } h(x,y,0) = \begin{cases} 1, & x < 0.5, \quad \forall y \\ 0.5, & x \geq 0.5, \quad \forall y \end{cases} \tag{11}$$

The barrier at x=0.5 is removed at t=0, slip boundary condition is implemented at both the boundaries.
Water bed height and velocity field are obtained by the solver and then compared with the analytical solution for this condition as given by J.J. Stokers [5].

$$u(x,t) = \begin{cases} 0 & \text{if} \quad x < \frac{1}{2} - t\sqrt{g} \\ \frac{1}{3t}(2x - 1 + 2t\sqrt{g}) & \text{if} \quad \frac{1}{2} - t\sqrt{g} \leq x \leq (u_2 - c_2)\, t + \frac{1}{2} \\ u_2 & \text{if} \quad (u_2 - c_2)\, t + \frac{1}{2} < x \leq \mathrm{S}t + \frac{1}{2} \\ 0 & \text{if} \quad x > \mathrm{S}t + \frac{1}{2} \end{cases}$$

$$h(x,t) = \begin{cases} 1 & \text{if } x < \frac{1}{2} - t\sqrt{g} \\ \frac{1}{9g}\left(2\sqrt{g} - \frac{2x-1}{2t}\right)^2 & \text{if} \quad \frac{1}{2} - t\sqrt{g} \leq x \leq (u_2 - c_2)\, t + \frac{1}{2} \\ \frac{1}{4}\left(\sqrt{1 + \frac{16S^2}{g}} - 1\right) & \text{if} \quad (u_2 - c_2)\, t + \frac{1}{2} < x \leq \mathrm{S}t + \frac{1}{2} \\ \frac{1}{2} & \text{if } x > \mathrm{S}t + \frac{1}{2} \end{cases} \tag{12}$$

Where, S, u_2 and c_2 are given as:

$$\begin{aligned} u_2 &= S - \frac{g}{8S}\left(1 + \sqrt{1 + \frac{16S^2}{g}}\right) \\ c_2 &= \sqrt{\frac{g}{4}\left(\sqrt{1 + \frac{16S^2}{g}} - 1\right)} \\ S &= 2.957918120187525 \end{aligned} \tag{13}$$

To compare the results, we have taken a xz plane at the middle of domain and plotted numerical result and the analytical solution on graph. As you can see the numerical solution

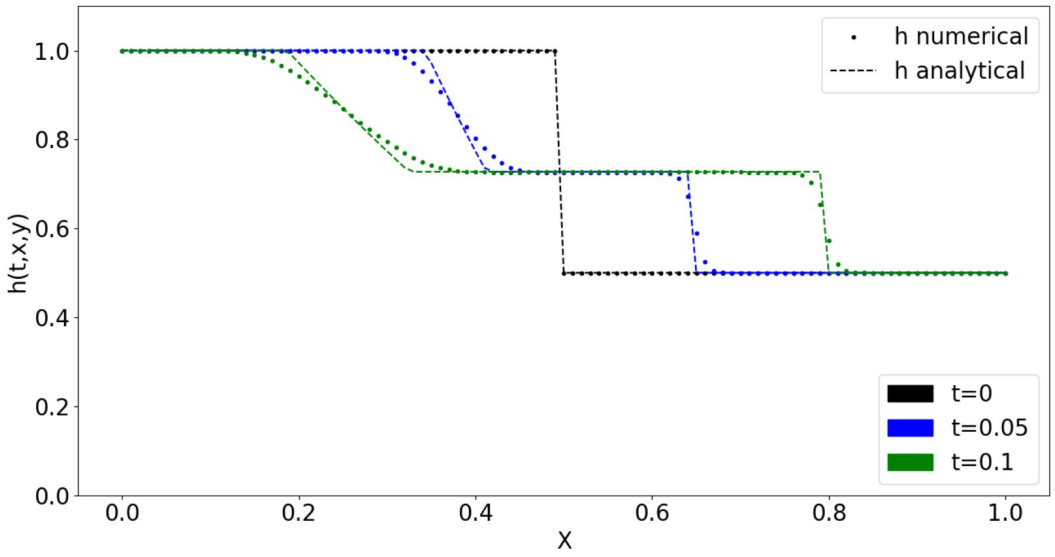


Figure 4: Comparison of pseudo-2D numerical solution with 1D analytical solution

exactly follows the analytical solution but there is a slight false diffusion present at the at the edges of numerical solution that we will discuss later.
The study was done for 300×300 mesh grid.

4.2 Case 2: Flow on 3D parabolic profile

Water flow simulation is carried out on a parabolic profile, which involve the significance of the source term in the governing equation. Again frictionless bottom was considered while solving SWE.
The equation of 3D parabola and the initial condition are given as follows:

$$\begin{aligned} z(r) &= -h_0\left(1-\tfrac{r^2}{a^2}\right) \\ r &= \sqrt{(x-L/2)^2+(y-L/2)^2} \end{aligned} \tag{14}$$

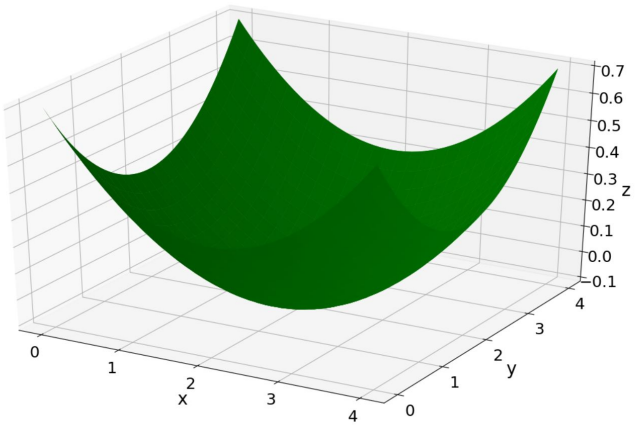


Figure 5: 3D parabolic profile

The exact analytical solution was given by W.C. Thacker for 2D SWE but it was only applicable on frictionless bottom [6].

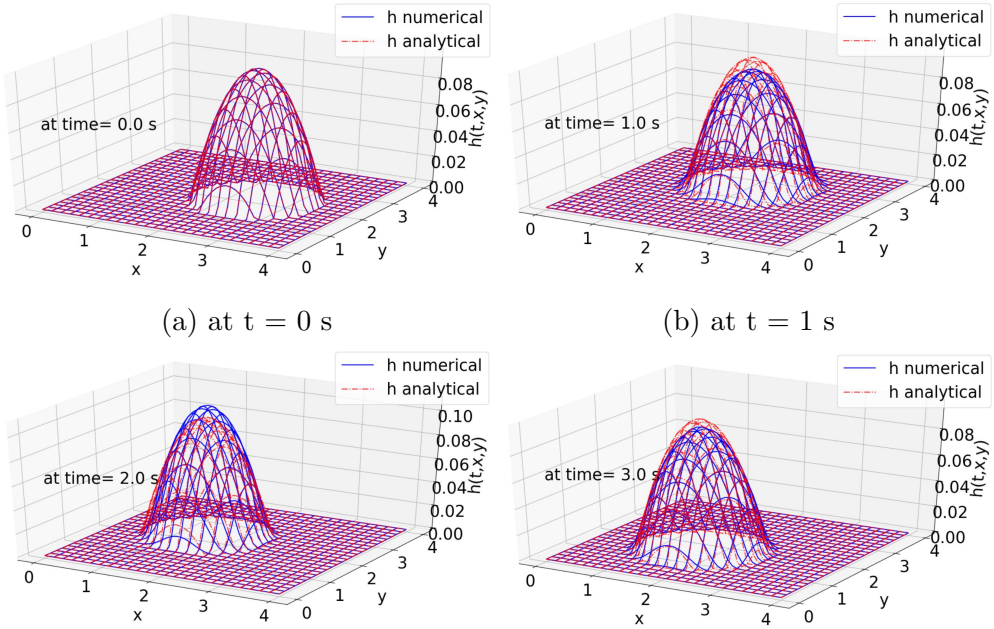
$$\begin{aligned} h(x,y,t) &= \frac{\eta h_0}{a^2}\left(2\left(x-\frac{L}{2}\right)\cos(\omega t)+2\left(y-\frac{L}{2}\right)\sin(\omega t)-\eta\right)-z(x,y) \\ u(x,y,t) &= -\eta\omega\sin(\omega t) \\ v(x,y,t) &= \eta\omega\cos(\omega t) \end{aligned} \tag{15}$$

Where ω is the frequency and given by $\omega = \frac{\sqrt{2gh_0}}{a}$ and η is a parameter. For this study we considered, L = 4 m, a= 1 m, $h_0 = 0.1$ and $\eta = 0.5$ for 250×250 mesh grid.

It is hard to visualize total height of the free surface from the reference, so we have only plotted the surface height from the surface.

As you can see in figure 6, there is a slight deviation of the numerical solution with the analytical solution due to presence of false diffusion in the scheme which can be reduced using very fine grid.

A mesh convergence study is perform and the root mean square error of the numerical solution is computed and a log-log graph is plotted with the RMSE and mesh size. The



(a) at t = 0 s (b) at t = 1 s

(c) at t = 2 s (d) at t = 3 s

Figure 6: Comparison of numerical and analytical solution for parabolic flow

slope of this graph gives the order of accuracy of the scheme which comes out to be 0.7 which is close to 1.0 that is theoretical value. As you can see the RMSE is in range of 10^5 for 70×70 mesh size which is reasonably accurate. The RMSE decreases due to decrease in the numerical diffusion which decrease with the mesh size.

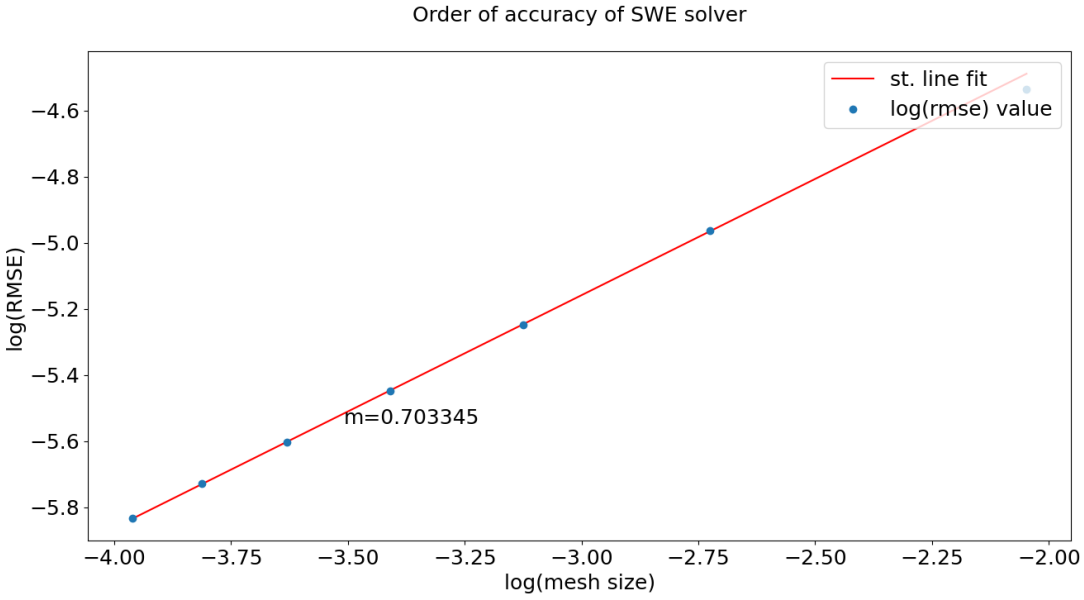


Figure 7: Mesh convergence study

5 Features of our Solver

5.1 Advantages

- The main advantage of our solver is the ease of understanding the source code as this take advantage of the simplicity and features of python. Any new contributor can easily and quickly understand this solver and can further add new features. There are also solver like FORTRAN based fluidity and C++ based OpenFOAM, but if you want to contribute or add new feature you may take weeks or months to understand the code structure.
- This solver is available on GitHub for open source contribution. Anyone from IIT Mandi or outside can become its next contributor and make this project grow. As python is new and there is a big python development community, so it is expected that this project may grow at a fast pace.
- Although python is considered a slow computing language, but we have used its very useful module i.e numpy which helps to perform vectorized operation, reducing dependency on loops. Implementing this vectorized operation feature increased the computational speed many folds. The computation speed of our solver is compared with OpenFOAM, an open source C++ based fluid flow solver. As per figure 8, our solver is very fast compared to OpenFOAM.
- We have implemented adaptive time stepping which automatically determine the best time step size Δt according to stability condition i.e. CFL condition. This had also increased the computational speed many folds, saving both memory and time.

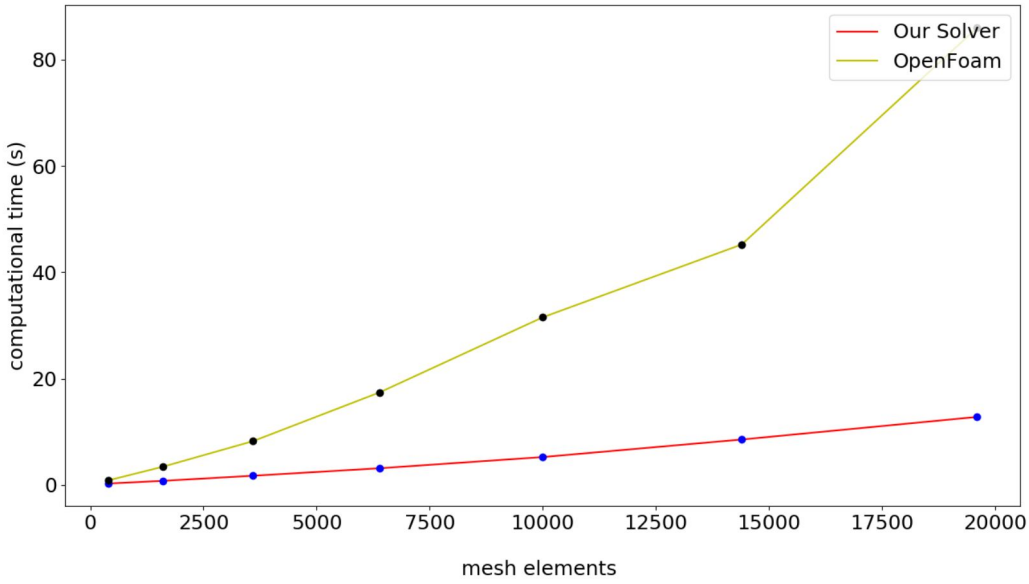


Figure 8: Computational speed comparision with OpenFOAM

- There already exist a python-based solver called mattflow, ready to use and it is also first order in nature and most of feature same as ours. But this mattflow cannot solve SWE with source term or on a 3D profile and its support has been discontinued since Dec 2020.

5.2 Limitation of our solver

- The greatest setback of our solver is the numerical/false diffusion arises with the first order scheme that we have implemented. However, this can be reduced using fine mesh grid but it will increase the computational cost. As you refine you mesh, the numerical solution converege to analytical solution as shown in figure 9.
- Due to false diffusion, our solver may not be able to capture very small amplitude wave solution of the SWE. But these small amplitudes are of little significance.
- This is an explicit scheme based solver. We are limited by the CFL condition to decide our time step size.

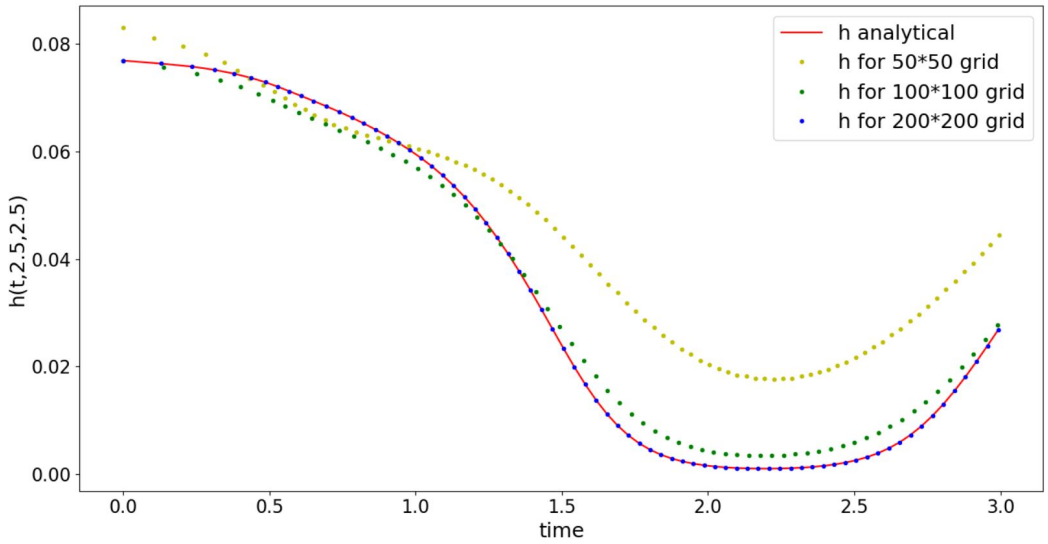


Figure 9: Mesh refinement

6 Overall Progress and Future Scope:

We have successfully developed 3D Advection-Diffusion Equation solver in previous semester which is also based on python. This semester we successfully developed both 1D and 2D Shallow Water Equation solver and implemented all the features as described in previous section.
This project has laid a foundation for solving Avalanche problem. We can do so by implementing plastic rheology in the existing SWE solver and can be used for model shallow

granular flow. However it may not as accurate as discrete element method but it will be reasonable accurate and computationally economical. With the ADE solver we can implement the heat transfer model of snow which have coupled effect on granular flow and we find many interesting quantities like temperature profile and pressure on wall by avalanche.

7 Conclusion

In this project, python based 2D shallow water equation solver was developed which uses finite volume method to discretize partial derivative equations. The results were computed against analytical solution and order of accuracy computed which is coming out to be 0.7. The RMSE of the solutions are computed which is under acceptance limit. Hence, the solver is working quite accurate but only problem is to use very fine grid to reduce false diffuse. Still the computational time for a 200*200 grid is around 25s. The code has been uploaded on the GitHub repository and further progress of the project will be pushed on repository.

Link to repo: `https://github.com/strawhat04/PySWE`

References

[1] Akhil Akkapelli and Gaurav Bhutani. Numerical modelling of free-surface flows using shallow water equations in a finite element framework. *Fluid Mechanics and Fluid Power*, December 2020.

[2] Alexander Kurganov and Guergana Petrova. A second-order well-balanced positivity preserving central-upwind scheme for the saint-venant system. *COMMUN. MATH. SCI.*, 2007.

[3] R. Setiyowati and Sumardi. A simulation of shallow water wave equation using finite volume method lax-friedrichs scheme. *IOP Conf. Series: Journal of Physics: Conf. Series 1306 012022*, 2019.

[4] A. Baghlani. S.M. Amiri, N. Talebbeydokhti. A two-dimensional well-balanced numerical model for shallow water equations. *Scientia Iranica*, 2012.

[5] J.J. Stoker. *Water Waves: The Mathematical Theory with Applications*. Wiley, 1957.

[6] William Carlisle Thacker. Some exact solutions to the nonlinear shallowwater wave equations. *Journal of Fluid Mechanics*, 1981.