NUMERICAL SIMULATION OF DAM-BREAK PROBLEM

MAE 442/542 FINAL PROJECT

BY

SIDDHANT S. APHALE

DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

UNIVERSITY AT BUFFALO, THE STATE UNIVERSITY OF NEW YORK

DECEMBER 21, 2015

ABSTRACT:

This report discusses the implementation of numerical schemes to solve a Dam-Break Problem. The Dam Break Problem is defined by Shallow Water Equations which is also known as Reimann Problem mathematically. In particular, this report discusses three methods to solve the problem namely, Godunov Method, Runge-Kutta with Roe-Sweby flux limiter also known as Runge-Kutta with TVD and Weighted ENO which is a higher order scheme. All the methods are studied in detail and compared on various grounds. Numerical results are compared with the results given by S. Vincent et al. (2001) and the accurateness of the schemes is checked. Grid sensitivity is analysed by checking the results for different CFL numbers. The computational time and cost are also discussed in the report.

1.0 INTRODUCTION:

Engineering problems are widely governed by hyperbolic equations. Shallow water equations, euler gas dynamic equations, open channel flow problems are some of the practical examples of Hyperbolic equations. Shallow water equations also known as Saint-Venant equations are a set of hyperbolic partial differential equations that describe the flow below a pressure surface in a fluid. The equations are derived from depth integrating the Navier-Stokes equations, in the case where the horizontal length scale is much greater than the vertical length scale. Shallow water equation is a non-linear equation which means that the use of analytical techniques to solve these equations can be found only in special cases. Numerical methods must be used to obtain the solution for engineering interests. As the shallow water equations are hyperbolic equations, finding numerical solution is a difficult task. Apart from the nonlinear nature, the hardship we run into solving these are the presence of discontinuities. Hyperbolic equations have discontinuous solution, in addition to smooth solutions. Even If the initial data is smooth, the nonlinear character along with the hyperbolic type of equation can lead to discontinuous solution in a finite time. In this particular problem, the discontinuity is associated with shock waves and contact surfaces.

In this report we are concentrating on the numerical methods for observing the behavior of Dam-Break Problem governed by 1-dimensional shallow water equation in which discontinuities are present and are important to model. Three numerical methods are implemented to find the solution. Godunov 1st order accurate with Lax-Freidrich's flux, Modified Runge-Kutta 4th order with TVD and Weighted ENO methods were implemented. All these methods are studied in detail and their implementation along with results is discussed and compared.

1.1 Problem Definition:

Initially, the water on both the sides of the dam is at rest and the dam is located at x=50m with L=100m. Upstream condition is defined for x<50m and downstream when x>50m. The depth of the water upstream is H=10m whereas the water depth downstream is H=1m. At t=0, the dam breaks and the water flows from upstream to downstream. The governing equation was 1-D Shallow water equation. The solution was simulated until t=4s. The initial mesh size taken was Δx=1m and successively grid was made finer. The results with finer grid sizes was compared. The results obtained from these methods were compared and discussed component by component treating it as a scalar equation.

The governing equation is given by

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}(\vec{Q})}{\partial x} = \vec{B} \qquad (1)$$

where

$$\vec{Q} = \begin{pmatrix} H \\ m \end{pmatrix}, \vec{E}(\vec{Q}) = \begin{pmatrix} m \\ mu + \frac{1}{g} gH^2 \end{pmatrix}, \vec{B} = \begin{pmatrix} 0 \\ -gH\frac{\partial b}{\partial x} \end{pmatrix}$$

H is the total depth, b is the elevation of the basin of the water depth if the surface is at rest, m=Hu is the momentum and g is the gravity.

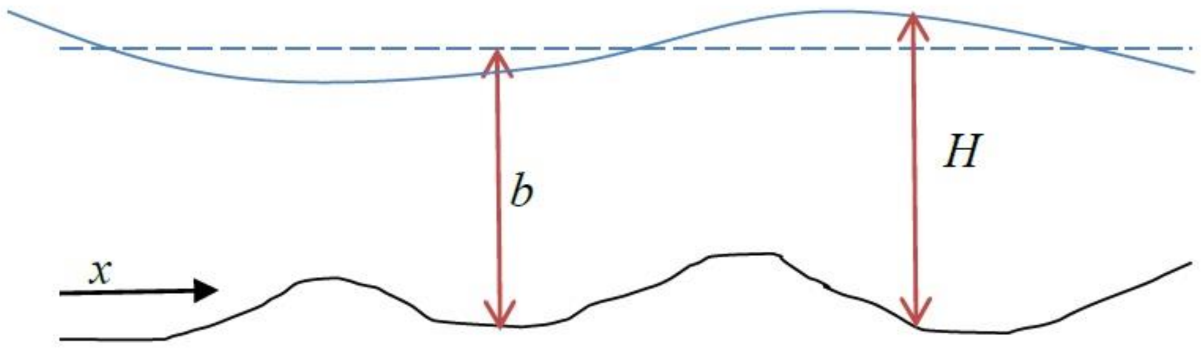


Figure 1: Problem Definition

1.2 Discretization:

The domain is considered of length 0 to 100m The dam is located at 50m. The governing equation was discretized with respect to flux. The fluxes are also discretized at half points.

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}(\vec{Q})}{\partial x} = 0 \qquad (2)$$

$$Q^{n+1} = Q^n - \frac{\Delta t}{\Delta x}(E(Q))^{-}_{i+\frac{1}{2}} - E(Q)^{+}_{i-\frac{1}{2}} \qquad (3)$$

where $E(Q)^{+}_{i-\frac{1}{2}}$ and $E(Q)^{+}_{i+\frac{1}{2}}$ are the fluxes at half points for the i^{th}cell.

For this problem, first order accurate discretization in space at half points by Taylor Series expansion was done.

1.3 Grid Definition:

We have defined a 1-Dimensional grid. The mesh size was defined by $\Delta x = L/100$. Each node is represented with i notation. The grid spacing was given and successively made finer to compare the effects of finer mesh size on the solution. As the grid is made finer, the computational cost increases but the results achieved are finer.

2.0 METHODS:

The governing equation for Dam-Break problem is classified as a Hyperbolic partial differential equation. These types of equations can be solved both by finite difference and finite volume methods. Methods like Lax-Wendroff method, MacCormack method, implicit methods like Beam-Warming method, etc. are examples of Finite difference method. Godunov method, total diminishing variation, essentially non-oscillatory, etc. are examples of Finite volume methods.

These equations can be treated as a single hyperbolic equation and solving one after the other in each time step, component by component as if it were a scalar equation. This approach simplifies the methodology to solve the nonlinear equations. The unknowns are interdependent and can be defined using the vectors Q and E after initial definition. They are updated likewise at each time step.

The initial mesh size given was 100 nodes along X-direction. The equations are solved till time t=4 seconds. The CFL number is adjusted depending on the mesh size and number of time iterations. Results are plotted for initial mesh size and successively for finer mesh size. The following discussed methods were solved and plotted using MATLAB software tool.

2.1 Godunov Method with Lax-Freidrichs Flux

The use of a conservative form of the equations is particularly important when dealing with problems admitting shocks or other discontinuities in the solution. A non-conservative numerical method, i.e. a numerical method in which the equations are not written in a conservative form, might give a numerical solution which looks reasonable but is incorrect. A well-known example is provided by Burger's equation, i.e. the momentum equation of an isothermal gas in which pressure gradients are neglected, and whose non-conservative representation fails dramatically in providing the correct shock speed if the initial conditions contain a discontinuity.

Godunov's method is a conservative Finite-volume method which solves exact or approximate Riemann problems at each inter-cell boundary. The Godunov scheme here is used along with the Lax-Freidrichs flux. Here the constant approximation over cell size Δx of is the average value of the solution at that node. Then the approximate solution to the Riemann problem is obtained and the solution is updated as averaged over the cell defining new approximations for the next step. A conservative scheme for the conservation law $\overrightarrow{Q_t} + E_x = 0$ is a numerical method of the form

$$Q^{n+1} = Q^n - \frac{\Delta t}{\Delta x}(E(Q))^{-}_{i+\frac{1}{2}} - E(Q)^{+}_{i-\frac{1}{2}} \qquad (4)$$

where Q^n is the cell average value of the Q and $E(Q)^{+}_{i-\frac{1}{2}}$ is called the numerical flux, an approximation to the physical flux *E(Q)*. It is necessary to solve the Reimann problem at the (Q^n_{i-1}, Q^n_i) and (Q^n_i, Q^n_{i+1}). The exact solution of the Reimann problem is computationally expensive, hence a flux function which satisfies the condition of monotonicity, consistency and Lipchitz continuity is used.

$$E_{i+\frac{1}{2}} = E\left(Q^{-}_{i+\frac{1}{2}}, Q^{+}_{i-\frac{1}{2}}\right) \qquad (5)$$

For Godunov method,

$Q^{-}_{i+\frac{1}{2}}$=Q_i and $Q^{+}_{i+\frac{1}{2}}$=Q_{i+1} .

Lax-Friedrichs flux is used to compute the fluxes at the half nodes which is given by,

$$E\left(Q^{-}_{i+\frac{1}{2}}, Q^{+}_{i+\frac{1}{2}}\right) = E(Q_i, Q_{i+1})$$

$$E(Q_i, Q_{i+1}) = \frac{1}{2}\left(E(Q_j) + E(Q_{j+1})\right) - |\propto|(Q_{j+1} - Q_j)$$

where $|\propto|$ is the maximum eigenvalue of the given system of equations. For Shallow water equations the eigenvalues are $u_R + c,\ u_R - c,\ u_L + c, u_L - c$ where c is the shock velocity given by $\sqrt{gH}$. The Godunov's method is monotonic, thus the solution is without spurious oscillations. However, as the method is first order, it is highly diffusive.

2.2 Modified Runge-Kutta with TVD using Roe-Sweby Upwind TVD Limiter:

The schemes which utilize central difference approximation of second-order develops oscillations in the vicinity of large gradients. This is due to the dispersion error in these schemes. The Beam and Warming implicit scheme and the Runge-Kutta scheme or the modified Runge-Kutta scheme are examples of such schemes. To eliminate or reduce the oscillations and thereby improve the solution, damping terms are added. TVD essentially eliminates or reduces the oscillations in the solution, hence, it can be added to the finite difference equations to provide a mechanism to reduce the oscillations. We have added a fourth order damping term to the modified Runge-Kutta scheme. The TVD is added at the final stage. The formulation is as given below

$$u_i^{(1)} = u_i^n$$

$$u_i^{(2)} = u_i^n - \frac{\Delta t}{4}\left(\frac{\partial E}{\partial x}\right)_i^{(1)}$$

$$u_i^{(3)} = u_i^n - \frac{\Delta t}{3}\left(\frac{\partial E}{\partial x}\right)_i^{(2)}$$

$$u_i^{(4)} = u_i^n - \frac{\Delta t}{2}\left(\frac{\partial E}{\partial x}\right)_i^{(3)}$$

$$u_i^{n+1} = u_i^n - \Delta t\left(\frac{\partial E}{\partial x}\right)_i^{(4)}$$

is augmented by the following step.

$$u_i^{n+1} = u_i^{n+1} - \frac{\Delta t}{2\Delta x}\left(\emptyset_{i+\frac{1}{2}}^n - \emptyset_{i-\frac{1}{2}}^n\right) \qquad (6)$$

where $\emptyset$ represents the flux limiter at half points. We have used Roe-Sweby Upwind TVD Limiter in this particular solution. The Roe-Sweby Upwind TVD Limiter is defined as follows

$$\emptyset_{i+\frac{1}{2}} = \left[\frac{G_i}{2}\left(\left|\propto_{i+\frac{1}{2}}\right| + \frac{\Delta t}{\Delta x}\propto_{i+\frac{1}{2}}^2\right) - \left|\propto_{i+\frac{1}{2}}\right|\right]\Delta u_{i+\frac{1}{2}}$$

$$\emptyset_{i-\frac{1}{2}} = \left[\frac{G_{i-1}}{2}\left(\left|\propto_{i-\frac{1}{2}}\right| + \frac{\Delta t}{\Delta x}\propto_{i-\frac{1}{2}}^2\right) - \left|\propto_{i-\frac{1}{2}}\right|\right]\Delta u_{i-\frac{1}{2}}$$

there are several choices for *G*. The *G* defined for our problem is

$$G_i = \frac{r + |r|}{1 + r}$$

where

$$r = \frac{u_{i+1+\sigma} - u_{i+\sigma}}{\Delta u_{i+\frac{1}{2}}}$$

$$\sigma = Sgn\left(\propto_{i+\frac{1}{2}}\right)$$

If at a point $\Delta u_{i+\frac{1}{2}}$ is zero, then *r* is set equal to zero in order to prevent a division by zero.

2.3 Weighted ENO Method:

Weak solutions of nonlinear conservation laws are piecewise smooth with jump discontinuities. High order accurate numerical approximations to these functions are such that they achieve high accuracy on smooth regions and sharpen profiles of discontinuities, without spurious oscillations. Essentially non oscillatory (ENO), polynomial reconstruction procedures were designed to accomplish this purpose. ENO methods are high-order accurate on smooth regions and appear to be very robust on shocks. A Weighted ENO (WENO) method that takes a convex combination of the three ENO approximations. Of course, if any of the three approximations interpolate across a discontinuity, it is given minimal weight in the convex combination in order to minimize its contribution and the resulting errors. Otherwise, in smooth regions of the flow, all three approximations are allowed to make a significant contribution in a way that improves the local
accuracy from third order to fourth order. Jiang and Shu improved the WENO method by choosing the convex combination weights in order to obtain the optimal fifth order accuracy in smooth regions of the flow. The WENO approximation for $\emptyset_x^{\pm}$ is a convex combination of the three possible ENO approximations. The given stencil is for $x_{i+\frac{1}{2}}^{-}$. The calculation of the stencil for $x_{i-\frac{1}{2}}^{+}$ is trivial and can be obtained from the equation below by trivial replacement of *i*.

$$s0 = \frac{1}{3}E_{i-2} - \frac{7}{6}E_{i-1} + \frac{11}{6}E_i$$
$$s1 = -\frac{1}{6}E_{i-1} + \frac{5}{6}E_i + \frac{1}{3}E_{i+1}$$

$$s2 = \frac{1}{3}E_i + \frac{5}{6}E_{i+1} - \frac{1}{6}E_{i+2}$$

the corresponding linear weights are $\gamma_0 = \frac{1}{16}$, $\gamma_1 = \frac{5}{8}$ and $\gamma_2 = \frac{5}{16}$. We calculate nonlinear weights by following way

$$w_0 = \frac{\alpha_0}{\alpha_0 + \alpha_1 + \alpha_2}$$
$$w_1 = \frac{\alpha_1}{\alpha_0 + \alpha_1 + \alpha_2}$$
$$w_2 = \frac{\alpha_2}{\alpha_0 + \alpha_1 + \alpha_2}$$

where

$$\alpha_0 = \frac{\gamma_0}{(\epsilon + b0)^2 + (\epsilon + b1)^2 + (\epsilon + b2)^2}$$
$$\alpha_1 = \frac{\gamma_1}{(\epsilon + b0)^2 + (\epsilon + b1)^2 + (\epsilon + b2)^2}$$
$$\alpha_2 = \frac{\gamma_2}{(\epsilon + b0)^2 + (\epsilon + b1)^2 + (\epsilon + b2)^2}$$

and

$$b0 = 13/12(E_{i-2} - 2E_{i-1} + E_i)^2 + 1/4(3E_{i-2} - 4E_{i-1} + E_i)^2$$
$$b1 = 13/12(E_{i-1} - 2E_i + E_{i+1})^2 + 1/4(3E_{i-1} - E_{i+1})^2$$
$$b3 = 13/12(E_{i-2} - 2E_{i-1} + E_i)^2 + 1/4(3E_{i-2} - 4E_{i-1} + E_i)^2$$

$\epsilon = 10^{-6}$. The values of fluxes is the weighted sum and is given as

$$E^{-} = w_0 s_0 + w_1 s_1 + w_2 s_2$$

Once we have the values of fluxes at $x^{-}_{i+\frac{1}{2}}$ and $x^{+}_{i-\frac{1}{2}}$. We can use Lax-Friedrichs flux or Roe flux to calculate the new values and advance in time.

3.0 RESULTS AND DISCUSSIONS:

3.1 Godunov Method

In implementation of Godunov method we have used Lax-Friedrichs flux and made first order approximations for the piecewise solution for Reimann problem. Due to the first order method, the accuracy of the solution is lower and it does not capture the discontinuity hence it is very dissipative. On the other end there are no oscillations or ringing at the sharp edges due to its first order accuracy. The results are plotted for $\Delta x = L/100$ and $\Delta t = 0.001$.

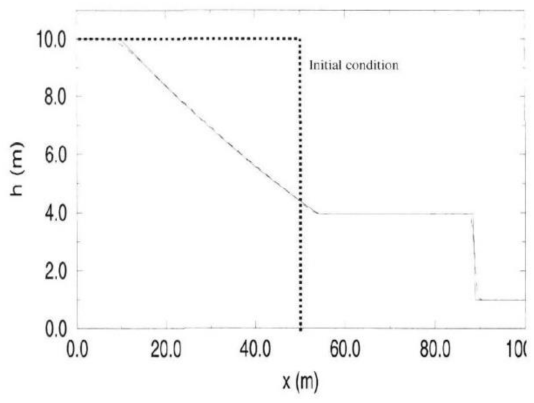
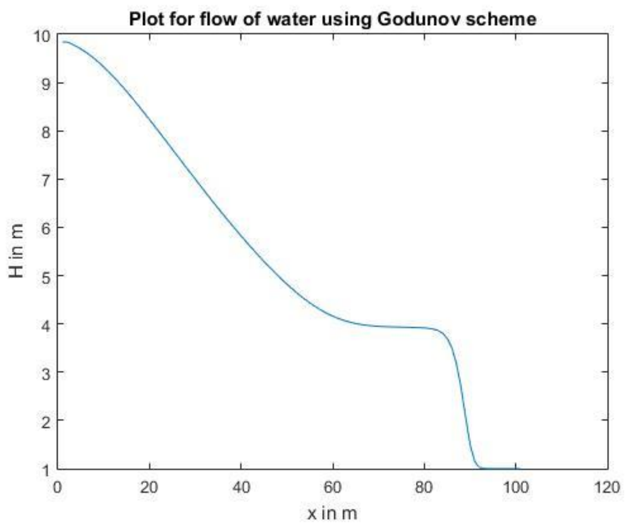


Fig. 3. Water surface profile at t = 4 s - Comparison between the analytical solution (______) and the simulation (- - -) with the MacCormack TVD scheme. 200 grid points are computed with $\Delta t = 0.01$.



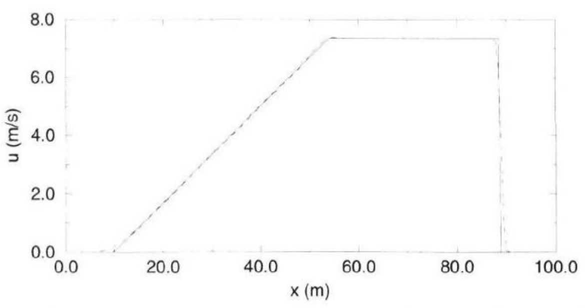


Fig. 4. Velocity profile at t = 4 s - Comparison between the analytical solution (______) and the simulation (- - -) with the MacCormack TVD scheme. 200 grid points are computed with $\Delta t = 0.01$.

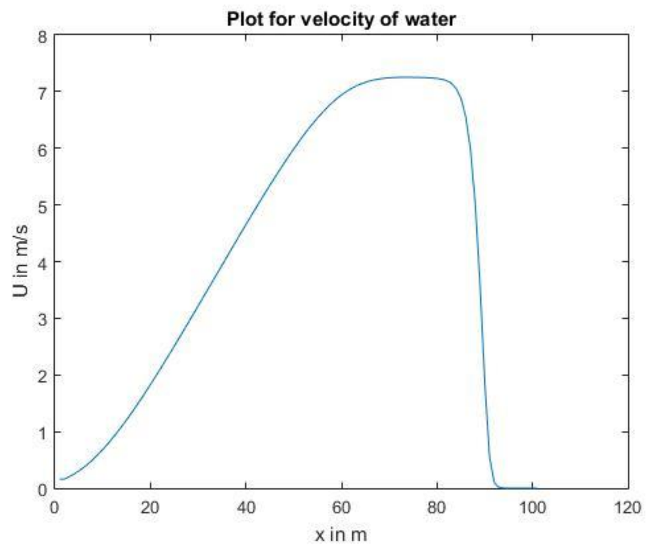


Figure 2: Godunov Method Water Surface and Velocity profile comparison with Analytical Solution

3.2 Modified Runge-Kutta with TVD using Roe-Sweby Upwind TVD Limiter:

In implementation of Modified Runge-Kutta TVD scheme we have made use of Roe-Sweby flux limiter. We have made first order approximation for the piecewise solution for the Riemann problem. As the method is second-order accurate and we have used a flux limiter there is high resolution shock capturing at the discontinuity. This method is second order accurate in space and fourth order accurate in time. Figure 3 shows the comparison of the water surface plot for the analytical solution and modified Runge-Kutta with TVD. It also compares the velocity plots for the same,

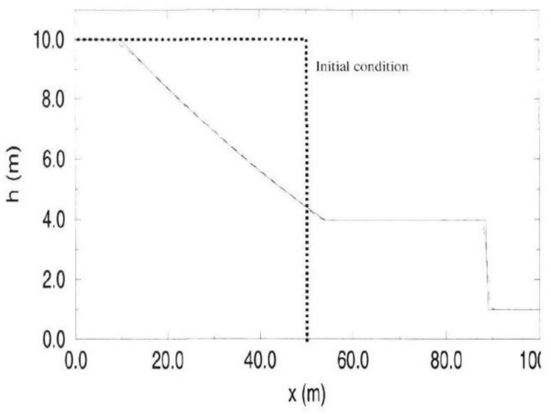
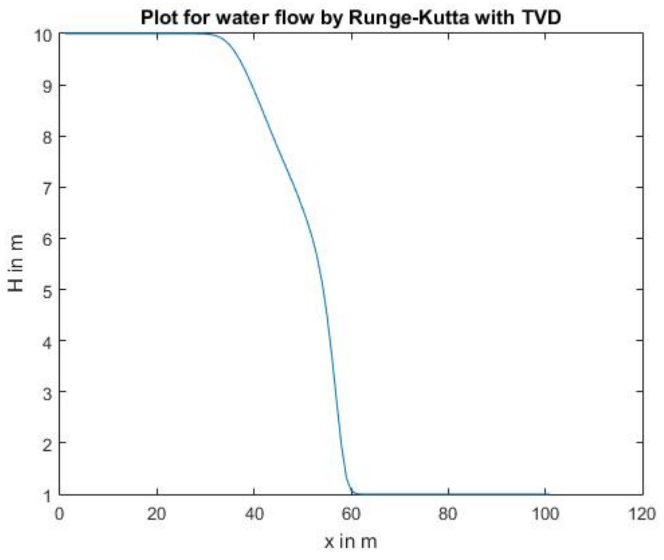


Fig. 3. Water surface profile at t = 4 s - Comparison between the analytical solution (______) and the simulation (- - -) with the MacCormack TVD scheme. 200 grid points are computed with $\Delta t = 0.01$.



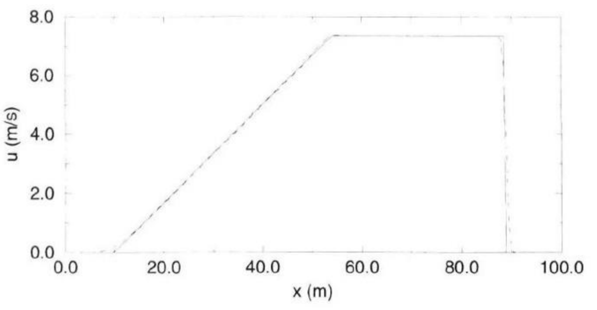


Fig. 4. Velocity profile at t = 4 s - Comparison between the analytical solution (______) and the simulation (- - -) with the MacCormack TVD scheme. 200 grid points are computed with $\Delta t = 0.01$.

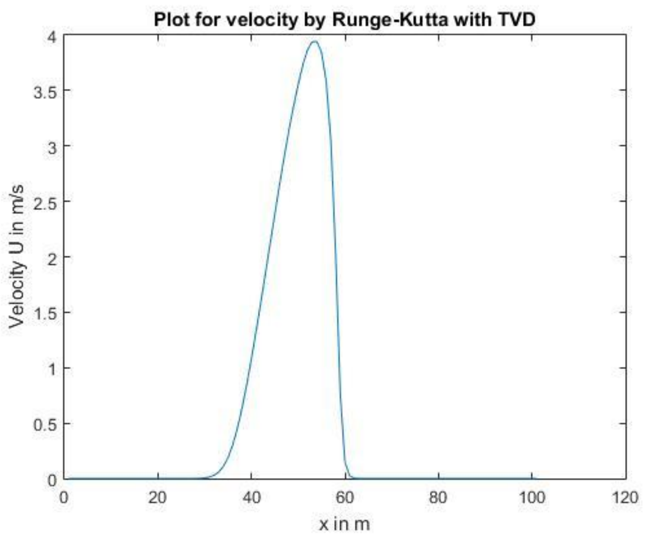


Figure 3: Modified Runge-Kutta with TVD Water Surface and Velocity profile comparison with Analytical Solution

3.3 Weighted ENO Method:

The results below are for weighted ENO method. It has $O(\Delta t^4, \Delta x^3)$ accuracy. It is the best method among all the methods discussed here. There are no oscillations and neither the method is diffusive. It has good shock capturing capacity. The results are good match to the results given from MacCormack TVD scheme by S. Vincent et al. (2001).

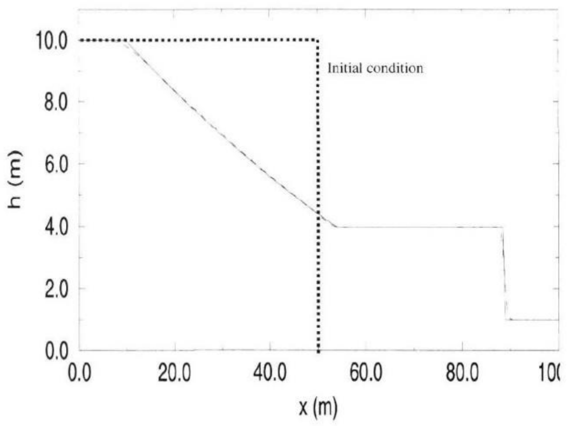
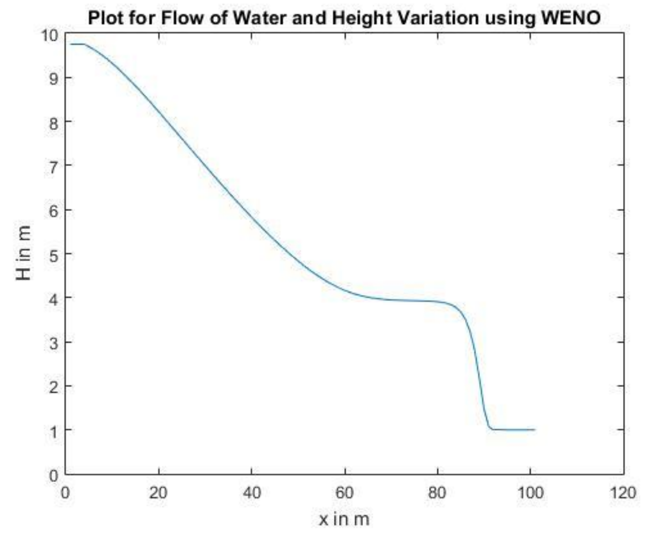


Fig. 3. Water surface profile at t = 4 s - Comparison between the analytical solution (______) and the simulation (- - -) with the MacCormack TVD scheme. 200 grid points are computed with $\Delta t = 0.01$.



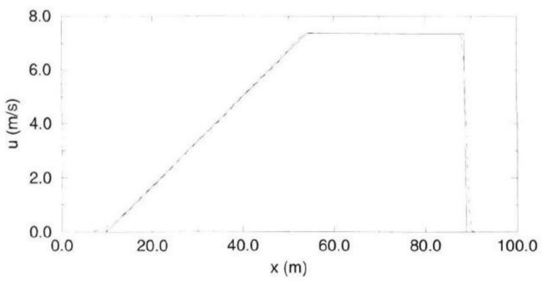


Fig. 4. Velocity profile at t = 4 s - Comparison between the analytical solution (______) and the simulation (- - -) with the MacCormack TVD scheme. 200 grid points are computed with $\Delta t = 0.01$.

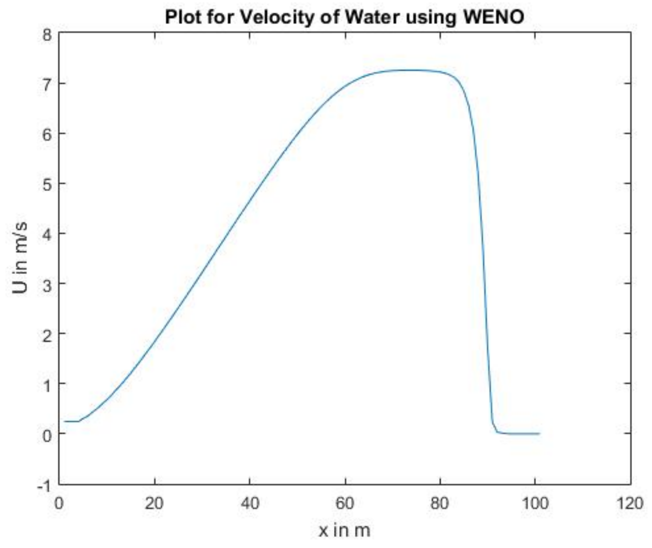


Figure 4: WENO Water Surface and Velocity profile comparison with Analytical Solution

4.0 GRID SENSITIVITY:

4.1 Godunov Method

Grid sensitivity analysis was done for all the methods discussed. Initially the mesh size used was $\Delta x = L/100$. Further the mesh was made finer successively with $\Delta x = L/200$ and $\Delta x = L/500$. It can be observed that with increase in mesh size we are tuning the CFL number which ultimately improves the solution. The comparison can be seen in the plots below in fig. 5.

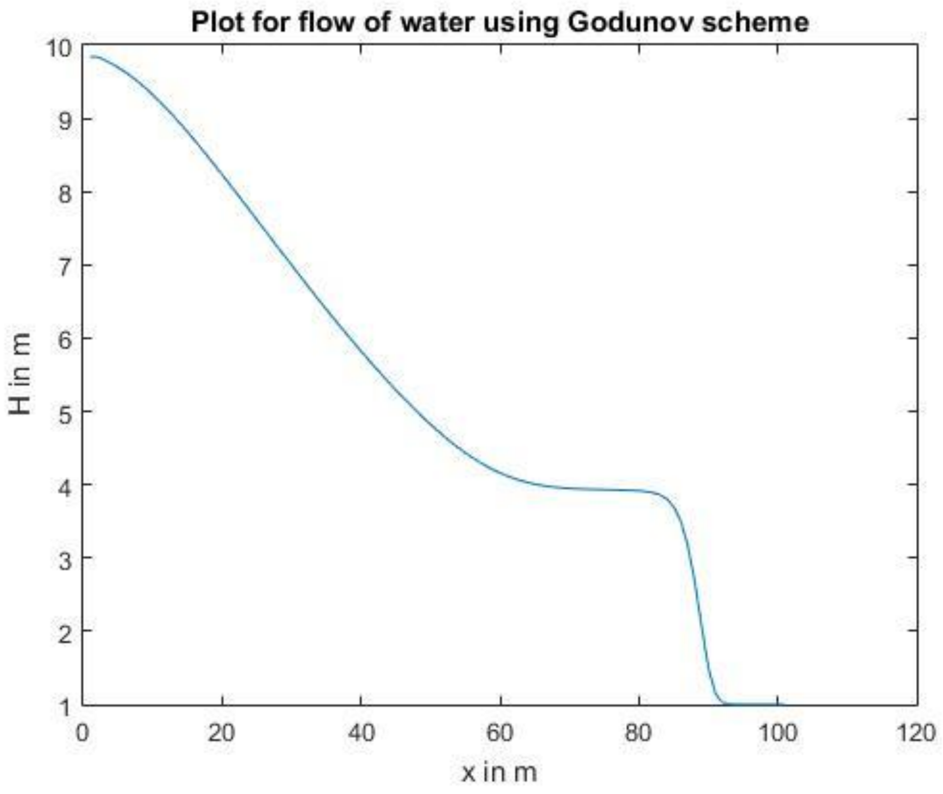


Figure 5a: Godunov Method with mesh size dx=L/100 Water Surface

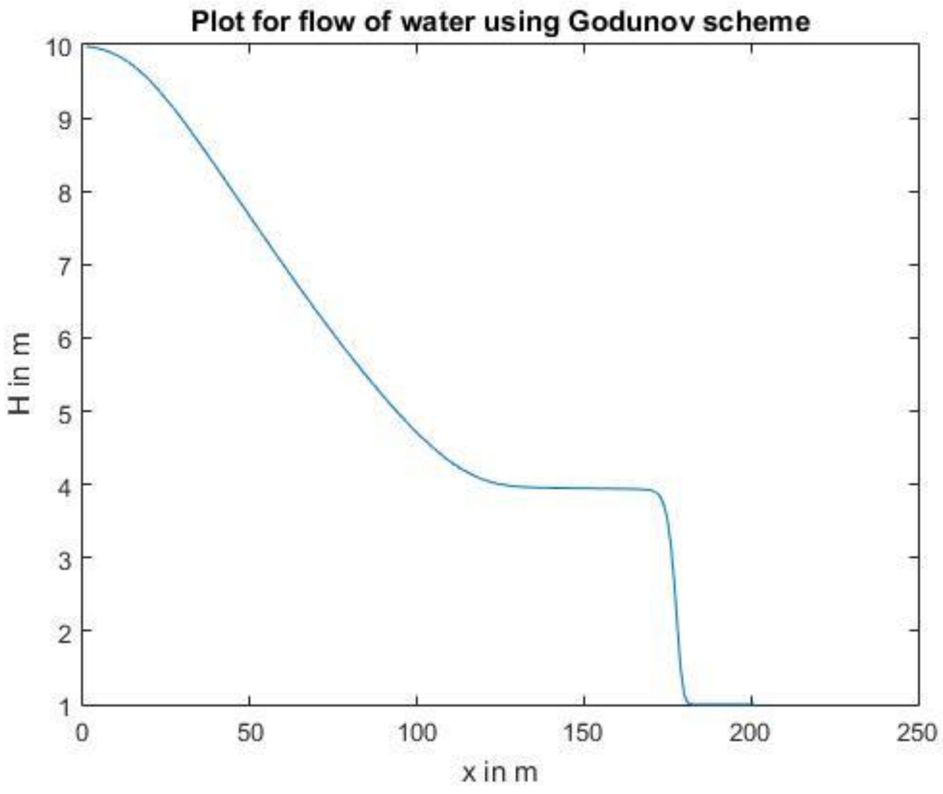


Figure 5b: Godunov Method with mesh size dx=L/200 Water Surface

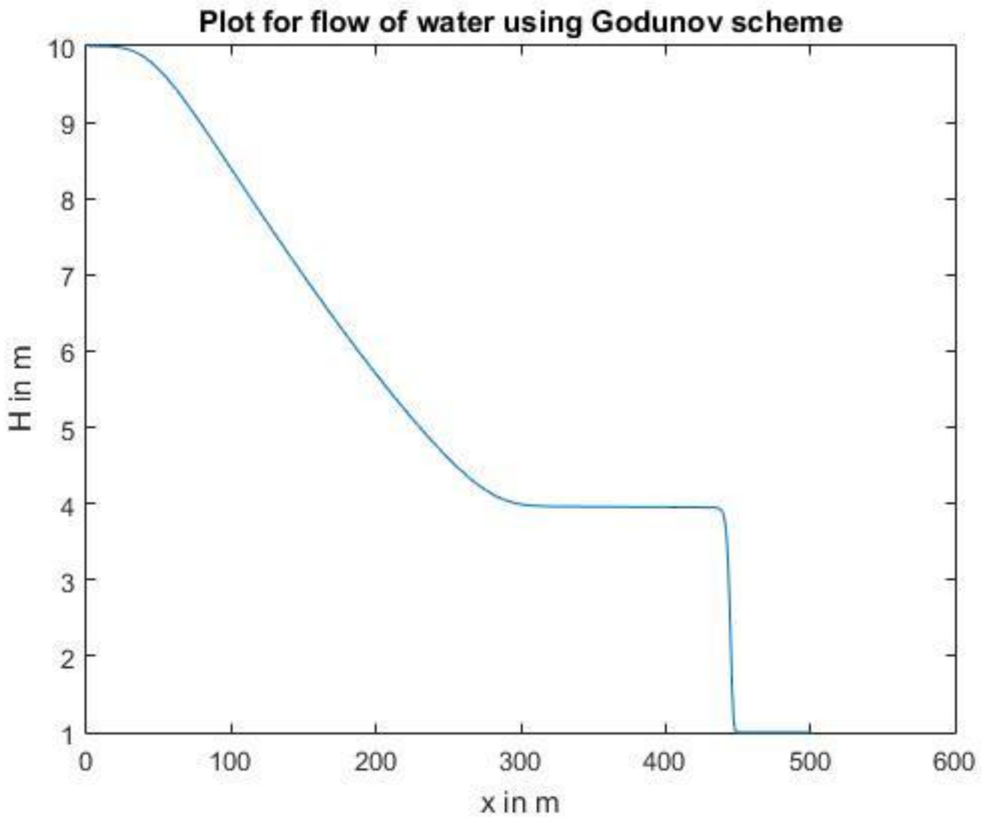


Figure 5c: Godunov Method with mesh size dx=L/500 Water Surface

4.2 Modified Runge-Kutta with Roe-Sweby TVD Flux Limiter:

We have analyzed the effects of grid sensitivity for R-K with TVD scheme. The comparison plots are shown in fig. 6a to 6c with mesh sizes $\Delta x = L/100, \Delta x = L/200$ and $\Delta x = L/500$. We can see that fig. 5c is more diffusive than fig. 5a and fig. 5b. Thus it is clear that as we make finer grid, the solution improves.

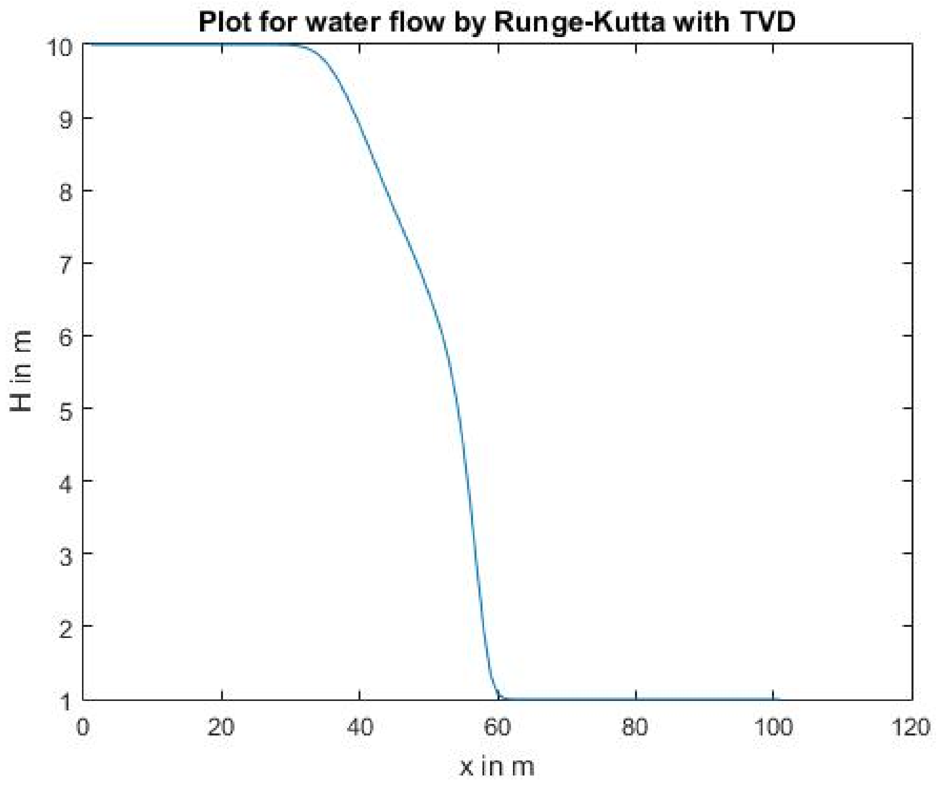


Figure 6a: R-K with TVD with mesh size dx=L/100 Water Surface

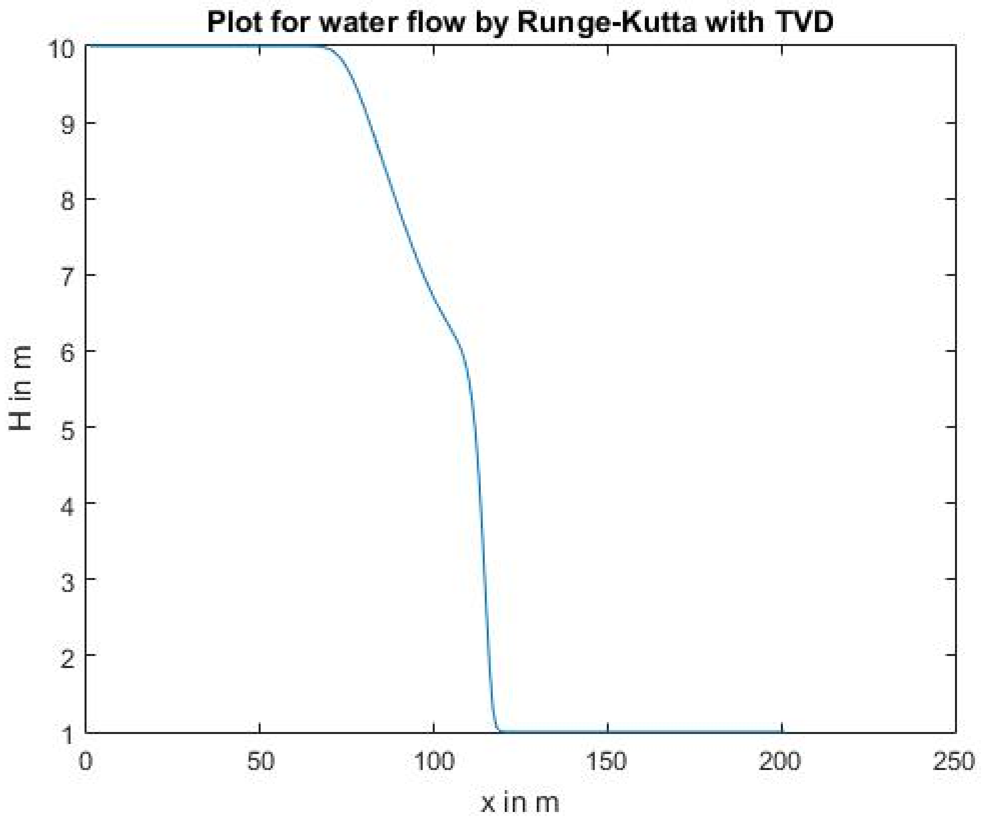


Figure 6b: R-K with TVD with mesh size dx=L/200 Water Surface

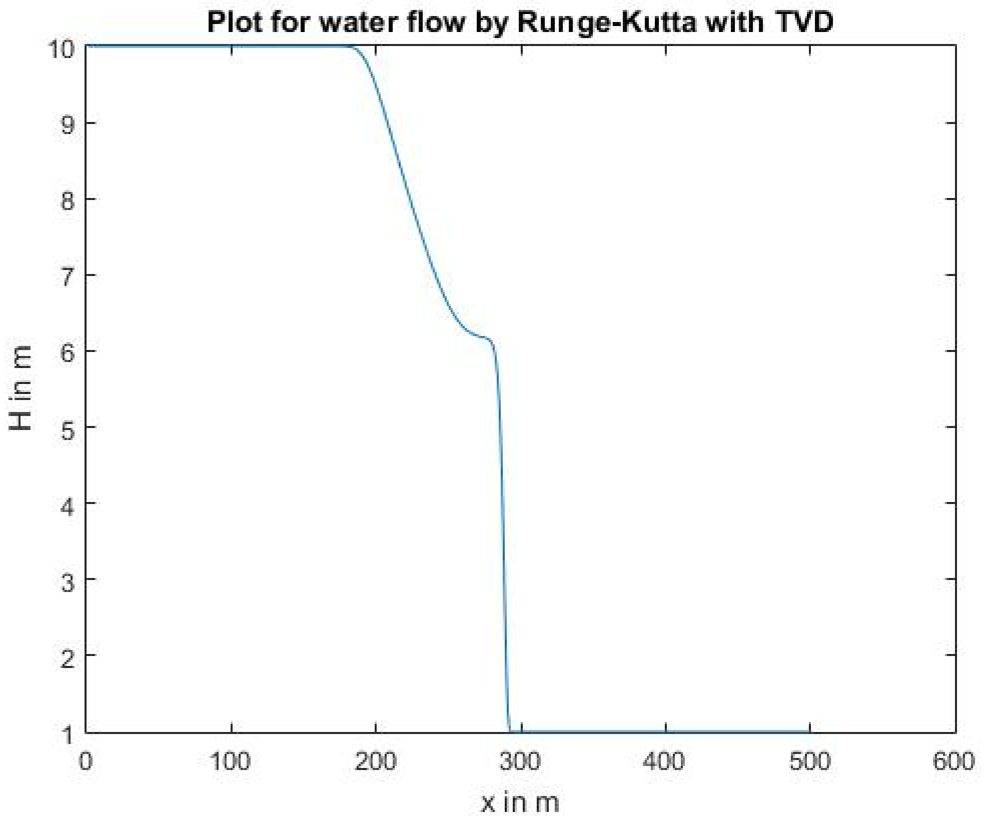


Figure 6c: R-K with TVD with mesh size dx=L/500 Water Surface

4.3 Weighted ENO:

On grid sensitivity analysis of Weighted ENO, the results achieved are show in fig. 7a to 7c. It is observed that the plot in fig. 7c resembles more accurate as compared to the analytical solution. Thus, as we make the mesh finer, we are making the solution more accurate.

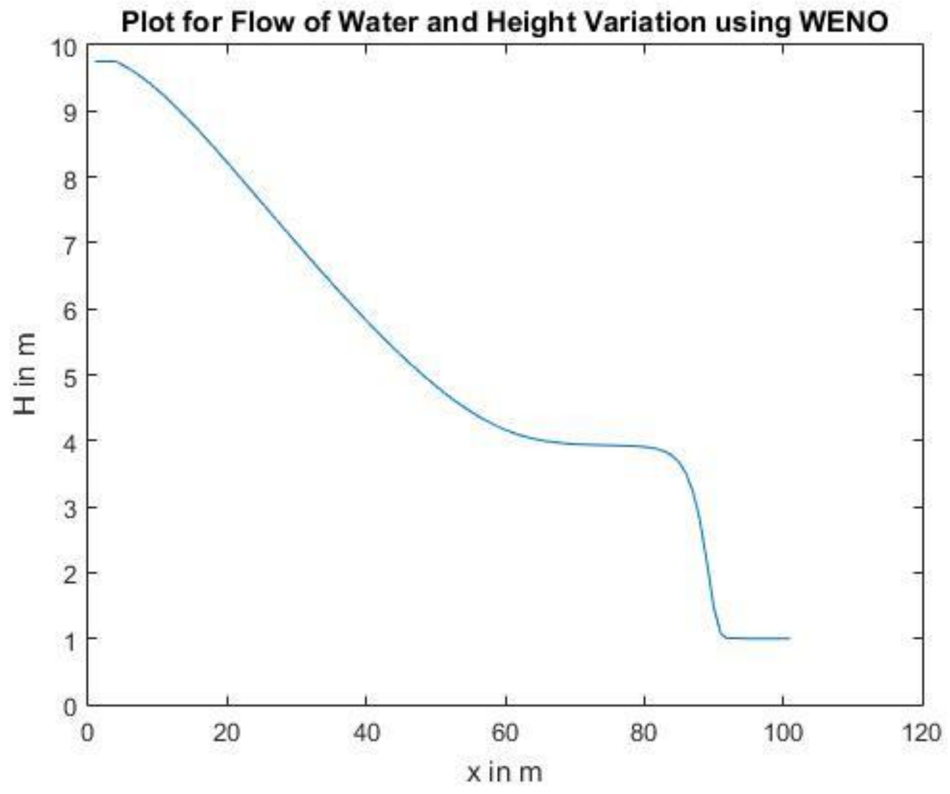


Figure 7a: WENO with mesh size dx=L/100 Water Surface

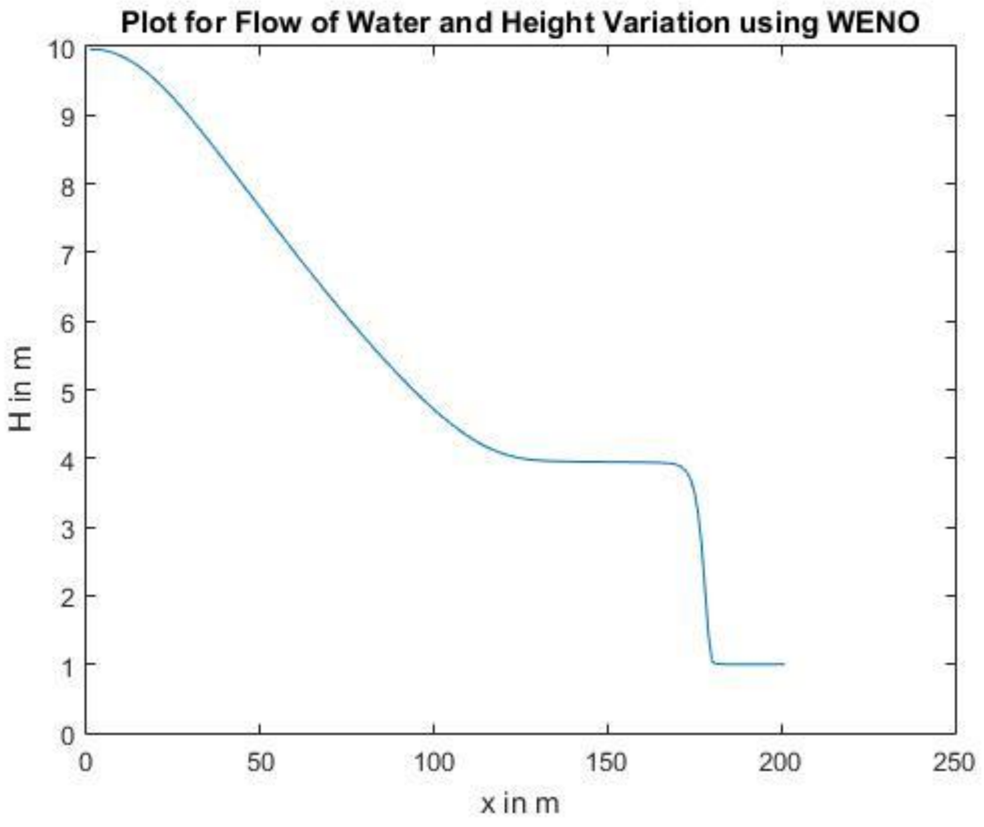


Figure 7b: WENO with mesh size dx=L/200 Water Surface

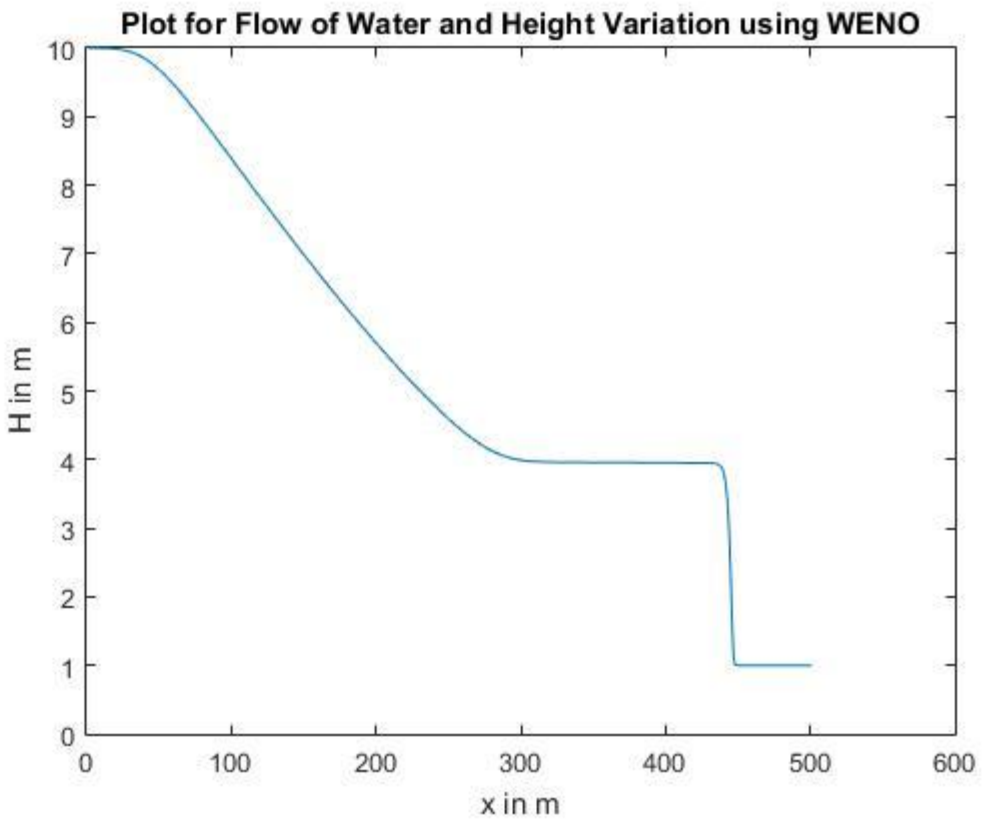


Figure 7c: WENO with mesh size dx=L/500 Water Surface

In all the methods for finer mesh size we have kept $\Delta t = 0.001$. Only the value of Δx was changed and the mesh was made finer. The results with changes in Δt are discussed below.

5.0 EFFECT OF Δt ON THE SOLUTION:

5.1 Godunov Method

We have now changed the value of Δt to analyze the effects on the solution. Changing the value of Δt will change the CFL Number and thus we can easily see the results for various CFL numbers. The value of Δt chosen for the analysis was 0.01, 0.001 and 0.0001. The value of $\Delta x = L/100$ is kept same for all the changes in Δt. The results are shown in fig. 8a to 8c. It can be observed that for higher value of Δt we get higher CFL number and the solution is more accurate. When we increase Δt beyond 0.1, the results are abrupt which implies that the stability criteria is void.

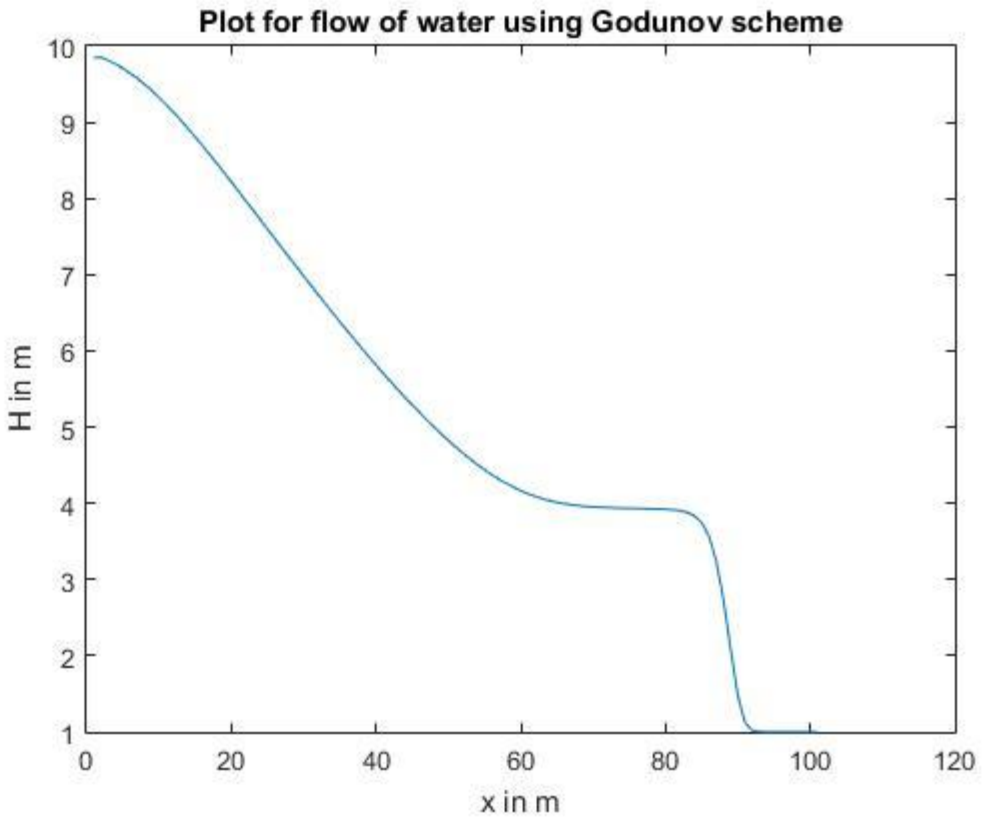


Figure 8a: Godunov Method with dt=0.01 Water Surface

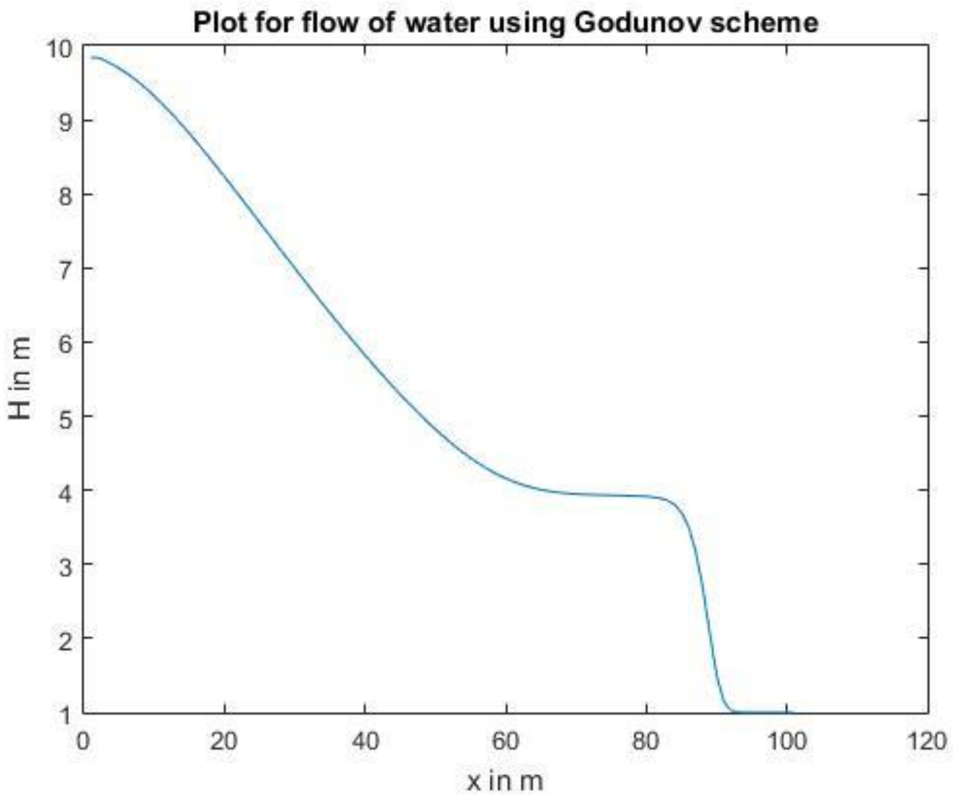


Figure 8b: Godunov Method with dt=0.001 Water Surface

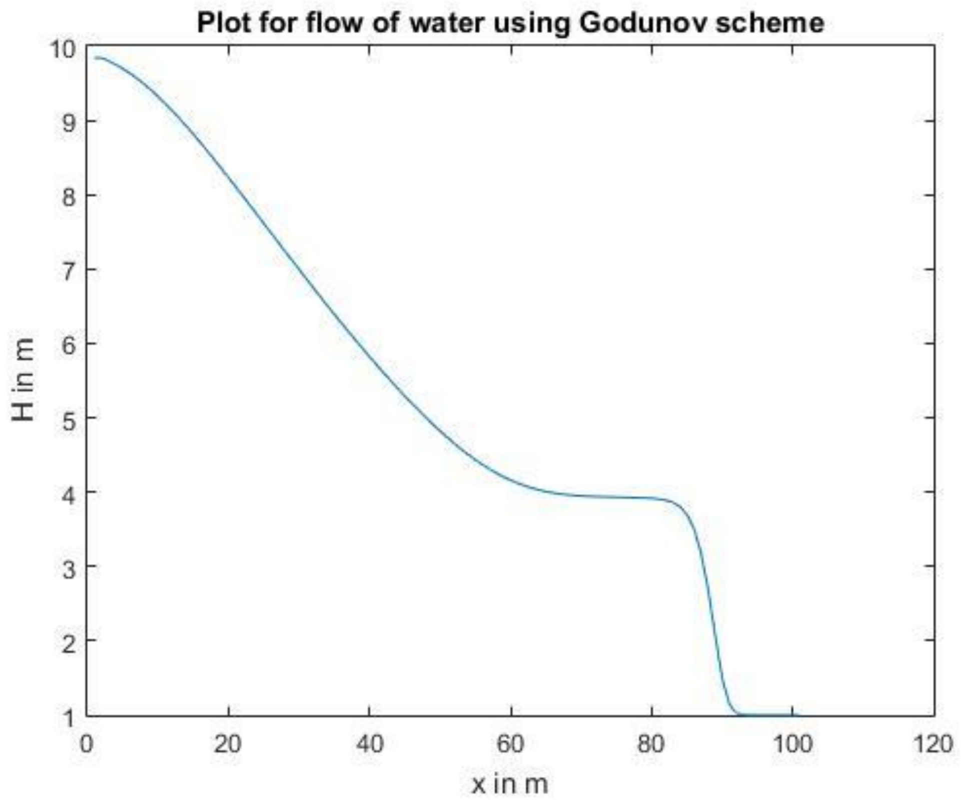


Figure 8c: Godunov Method with dt=0.0001 Water Surface

5.2 R-K with TVD:

Same as Godunov method, the values of Δt are changed for R-K with TVD and the plots obtained are shown in fig. 9a to 9c. As we increase the Δt we are increasing CFL number which provides more accurate solution. However, if the stability criteria is not met, the results are not stable.

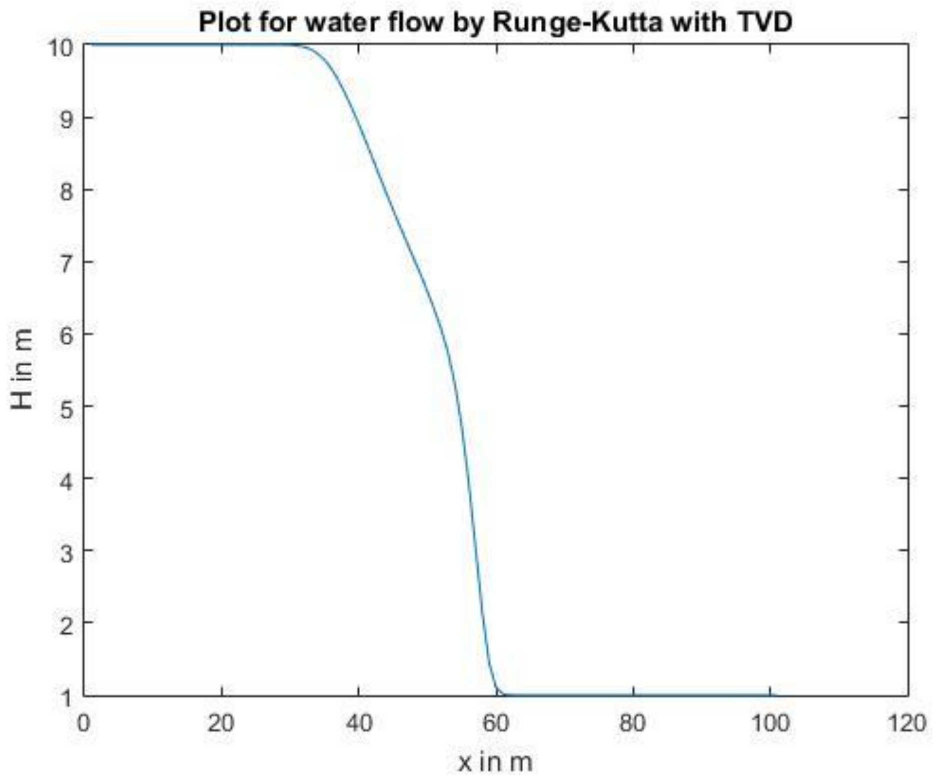


Figure 9a: R-K with TVD with dt=0.01 Water Surface

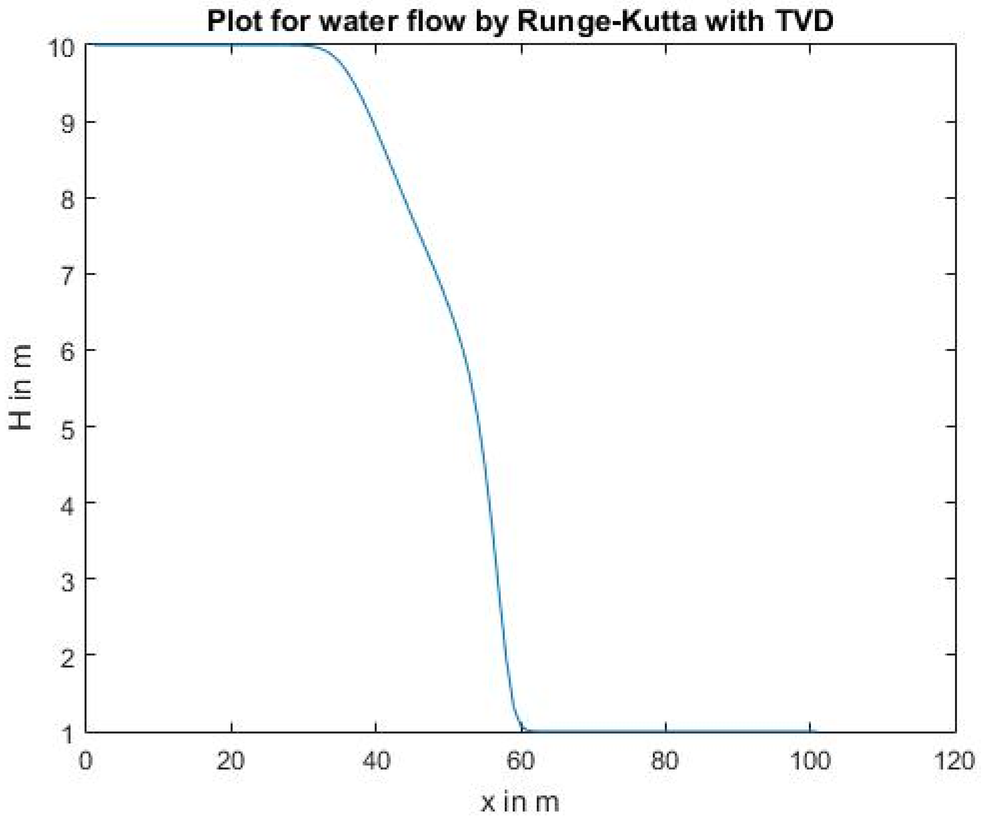


Figure 9b: R-K with TVD with dt=0.001 Water Surface

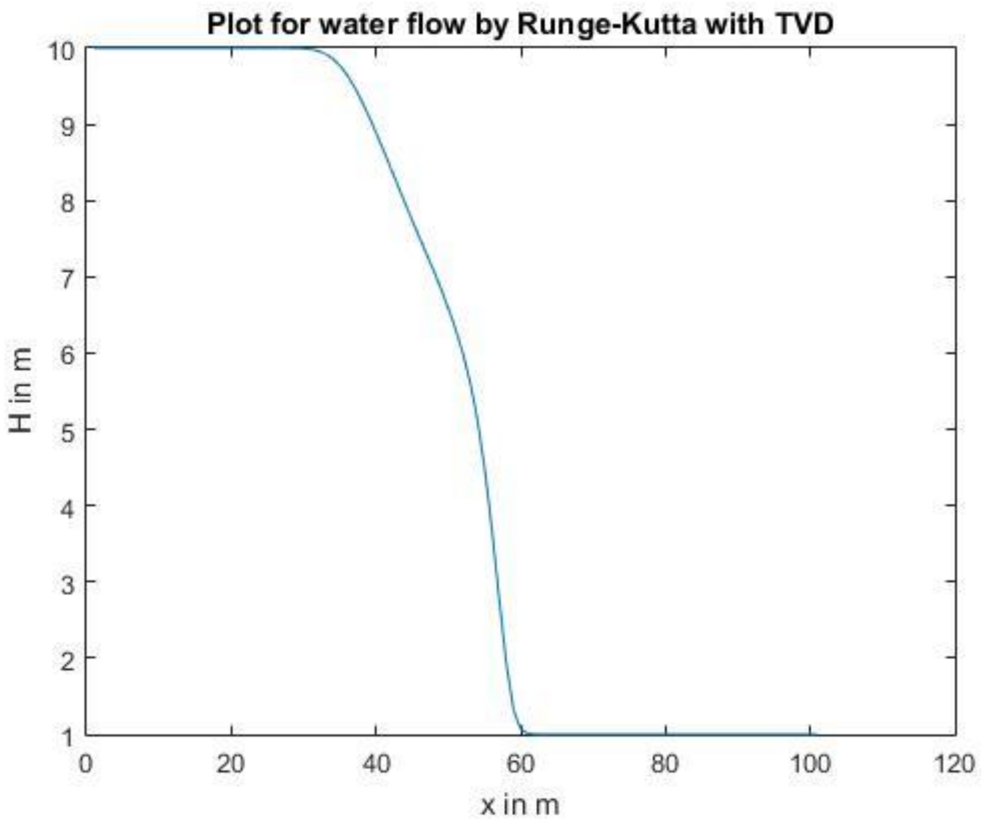


Figure 9c: R-K with TVD with dt=0.0001 Water Surface

5.3 WENO Method:

The value of Δt was again changed for WENO method and taken as 0.01, 0.001 and 0.0001. The plots obtained are shown in fig. 10a to 10c. It can be inferred from the plots that the method gives more accurate solution for higher CFL number. However higher CFL number increases the computational cost and time.

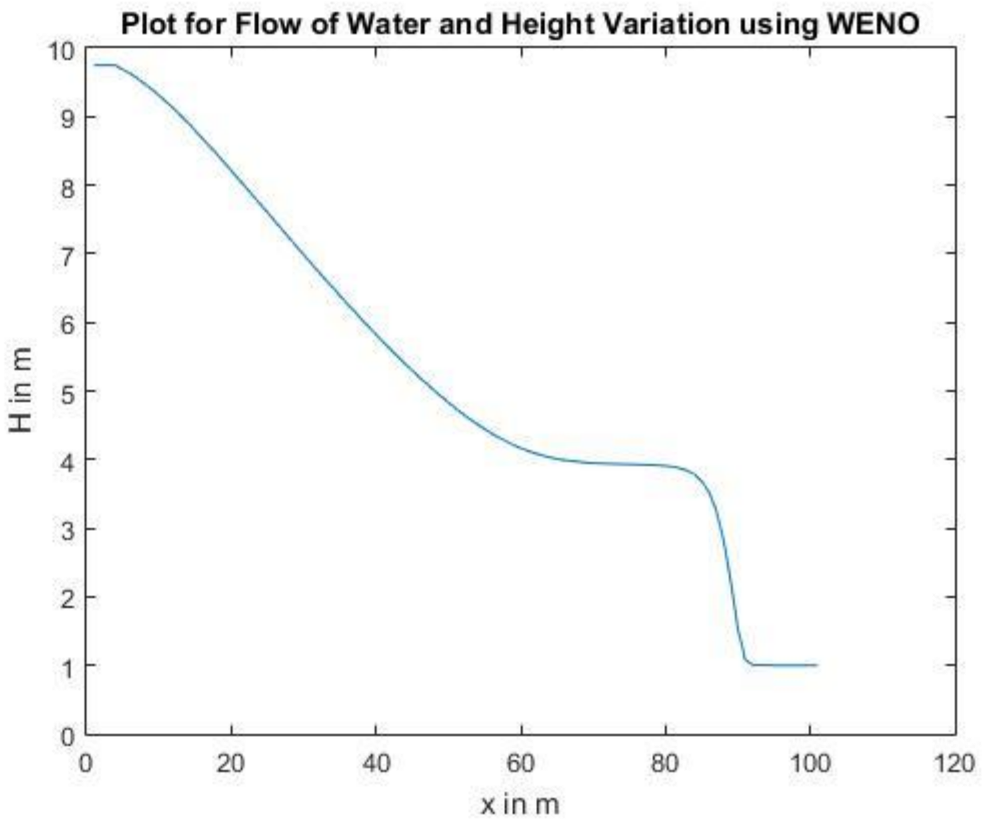


Figure 10a: WENO with dt=0.01 Water Surface

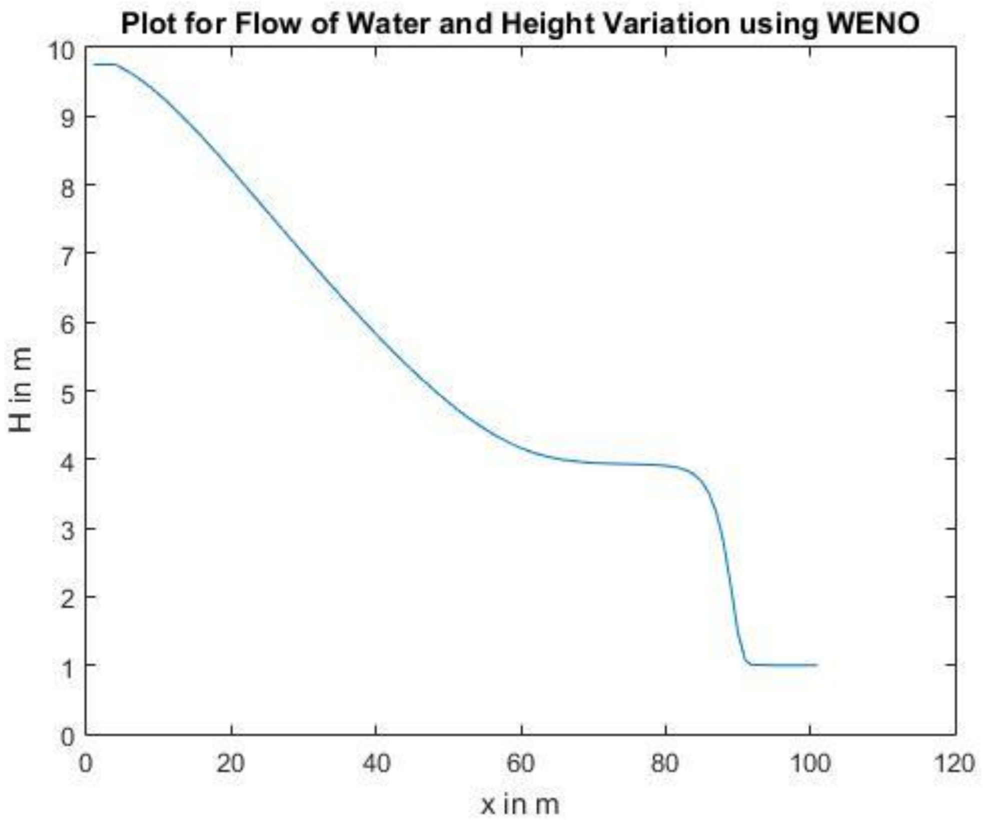


Figure 10b: WENO with dt=0.001 Water Surface

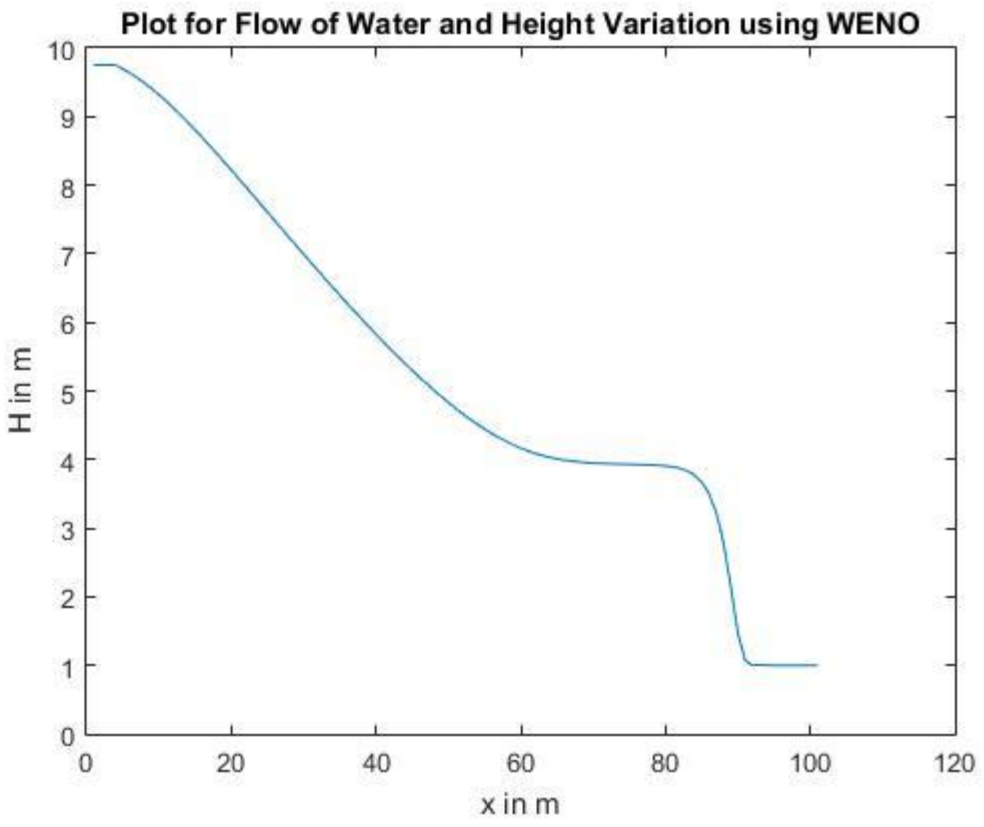


Figure 10c: WENO with dt=0.0001 Water Surface

6.0 SUMMARY AND CONCLUSION:

In the report, three numerical methods were presented to numerically solve the 1-D Dam-Break problem which is governed by Shallow-water equations. The methods were discussed in detail and then implemented using MATLAB to understand which method provides the best results. The solution was compared to the analytical solution proposed by S. Vincent et al. (2001). Results were compared for various CFL numbers.

Methods like Lax-Wendroff, MacCormack, etc which are second order accurate in both time and space are inappropriate to solve such nonlinear hyperbolic PDEs which involve discontinuities. Implementing such methods will lead to spurious oscillations. These oscillations are due to the schemes being not monotonic. We thus implemented higher order methods with limiters that help in producing oscillation free results. Godunov first order accurate scheme provides oscillation free results compared to R-K with TVD.

Further we used WENO which is higher order method compared to both Godunov and R-K with TVD. However, computation cost and time increases for such higher order methods. Further discontinuities can be removed by implementing more higher order methods like 5th Order accurate WENO.

7.0 REFERENCES:

1) Vincent, S., Caltagirone, J. P., & Bonneton, P. (2001). Numerical modelling of bore propagation and run-up on sloping beaches using a MacCormack TVD scheme. *Journal of hydraulic research*, *39*(1), 41-49.

2) Shu, C. W. (1998). *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws* (pp. 325-432). Springer Berlin Heidelberg.

3) Zoppou, C., & Roberts, S. (2000). Numerical solution of the two-dimensional unsteady dam break. *Applied Mathematical Modelling*, *24*(7), 457-475.

4) Godunov, S. K. (1959). A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, *89*(3), 271-306.

5) LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems* (Vol. 31). Cambridge university press.

6) Toro, E. F. (2009). *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media.

7) Nsom, B., Ndong, W., & Ravelo, B. (2008). Modelling the zero-inertia, horizontal viscous dam-break problem. *WSEAS Transactions on Fluid Mechanics*, *3*(2), 77-89.

8) Harten, A., Lax, P. D., & Leer, B. V. (1983). On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM review*,*25*(1), 35-61.

9) Saramito, P., Smutek, C., & Cordonnier, B. (2013). Numerical modeling of shallow non-Newtonian flows: Part I. The 1D horizontal dam break problem revisited. *International Journal of Numerical Analysis & Modeling, Series B*,*4*(3), 283-298.

10) Edom, E. (2008). *Numerical calculation of the dam-break Riemann problem with a detailed method and comparison with a simplified method* (Doctoral dissertation, Diploma Thesis of Leibniz Universität Hannover).

11) Fadhli, M., Mustafa, M., WB, W. N., & Kartono, A. (2013). Numerical method for dam break problem using Godunov approach.

12) www.mathworks.com/help/matlab/

13) www.cfd-online.com

14) www.wolframalpha.com

15) Notes by Dr. Iman Borazjani.

16) www.wikipedia.org

APENDICIES:

A-1 GODUNOV METHOD

```
%Godunov scheme for 1st order accurate
clear all; close all; clc;
N=101;
L=100;
delta_x=L/(N-1);
g=9.81;
delta_t=0.001;
t=4;
c=delta_t/delta_x;
% intial values of h and v
for i=1:N
    for l=1
        if i*delta_x<50
            Qold(1,i)=10;
            Qold(1+1,i)=0;
        else
            Qold(1,i)=1;
            Qold(1+1,i)=0;
        end
    end
end
tinitial=0;
while tinitial<=t
    tinitial=tinitial+delta_t;
    for i=1:N
        c(i)=sqrt(g.*Qold(1,i));
    end
    %Calculating eigen values
    for i=1:N
        eigen(1,i)=(Qold(2,i)./Qold(1,i))+c(i);
        eigen(2,i)=(Qold(2,i)./Qold(1,i))-c(i);
    end
    alpha=max(abs(eigen));
    for i=1:N
        Eold(1,i)=Qold(2,i);
        Eold(2,i)=((Qold(2,i)^2/Qold(1,i))+(0.5*g*(Qold(1,i)^2)));
    end
    for i=1:N-1
        F1(1,i)=0.5*(Eold(1,i)+Eold(1,i+1))-0.5*max(alpha(i),alpha(i+1))*(Qold(1,i+1)-Qold(1,i));
        F1(2,i)=0.5*(Eold(2,i)+Eold(2,i+1))-0.5*max(alpha(i),alpha(i+1))*(Qold(2,i+1)-Qold(2,i));
    end
    for i=2:N
        F2(1,i)=0.5*(Eold(1,i)+Eold(1,i-1))-0.5*max(alpha(i),alpha(i-1))*(Qold(1,i)-Qold(1,i-1));
        F2(2,i)=0.5*(Eold(2,i)+Eold(2,i-1))-0.5*max(alpha(i),alpha(i-1))*(Qold(2,i)-Qold(2,i-1));
    end
    for i=2:N-1
        Qnew(1,i)=Qold(1,i)-(delta_t/delta_x)*(F1(1,i)-F2(1,i));
        Qnew(2,i)=Qold(2,i)-(delta_t/delta_x)*(F1(2,i)-F2(2,i));
    end
    for l=1:2
        i=1;
        Qnew(l,i)=Qold(l,i)-(delta_t/delta_x)*(F1(l,i+1)-F1(l,i));
    end
    for l=1:2
        i=N;
        Qnew(l,i)=Qold(l,i)-(delta_t/delta_x)*(F2(l,i)-F2(l,i-1));
    end
    Qold=Qnew;
    for i=1:N
        Unew(1,i)=Qnew(2,i)/Qold(1,i);
    end
end
figure(1);
```

```
plot(Qnew(1,:));
title('Plot for flow of water using Godunov scheme');
xlabel('x in m'); ylabel('H in m');
figure(2);
plot(Unew(1,:));
title('Plot for velocity of water');
xlabel('x in m'); ylabel('U in m/s');
```

A-2 RUNGE-KUTTA WITH TVD

```
%Runge-Kutta with TVD Scheme
close all; clear all; clc;
N=101;
L=100;
dx=L/(N-1);
g=9.81;
dt=0.001;
t=4;
c=dt/dx;
% intial values of h and v
for i=1:N
    if i*dx<50
        h(i)=10;
        v(i)=0;
    else
        h(i)=1;
        v(i)=0;
    end
end
for i=1:N
    H_old(i)=0;
    U_old(i)=0;
    H_new(i)=0;
    U_new(i)=0;
    alphah1(i)=0;
    alphah2(i)=0;
    alphav1(i)=0;
    alphav2(i)=0;
    sigmah1(i)=0;
    sigmah2(i)=0;
    sigmav1(i)=0;
    sigmav2(i)=0;
    rh1(i)=0;
    rh2(i)=0;
    rv1(i)=0;
    rv2(i)=0;
    Gh1(i)=0; Gh2(i)=0; Gv1(i)=0; Gv2(i)=0;
    phih1(i)=0; phih2(i)=0; phiv1(i)=0; phiv2(i)=0;
end
for i=1:N
    H_old(i)=h(i);
    U_old(i)=v(i)*h(i);
end
for i=1:N
    H1(i)=0; H2(i)=0; H3(i)=0; H4(i)=0; %Defining R-K terms for height
    U1(i)=0; U2(i)=0; U3(i)=0; U4(i)=0; %Defining R-K terms for velocity
end
tinitial=0;
while tinitial<=t
    tinitial=tinitial+dt;
    %Defining Roe-Sweby Flux Limiter
    for i=1:N-1
            alphah1(i)=(U_old(i+1)-U_old(i))/(H_old(i+1)-H_old(i));
    end
    for i=1:N-1
```

```
               alphav1(i)=((((U_old(i+1).^2)/H_old(i+1))+0.5*g*H_old(i+1).^2)-
   ((U_old(i).^2)./H_old(i)+0.5*g*H_old(i).^2)./(U_old(i+1)-U_old(i)));
end
for i=2:N-1
    alphah2(i)=(U_old(i)-U_old(i-1))/(H_old(i)-H_old(i-1));
end
for i=1:N-1
               alphav2(i)=((((U_old(i).^2)/H_old(i)+0.5*g*H_old(i).^2)-((U_old(i-
   1).^2)./H_old(i-1)+0.5*g*H_old(i-1).^2)).*(U_old(i)-U_old(i-1)));
end
%Finding values of sigma
for i=1:N
    if alphah1(i)==0
        sigmah1(i)=0;
    else
        sigmah1(i)=alphah1(i)/abs(alphah1(i));
    end
end
for i=1:N
    if alphav1(i)==0
        sigmav1(i)=0;
    else
        sigmav1(i)=alphav1(i)/abs(alphav1(i));
    end
end
for i=2:N
    if alphah2(i)==0
        sigmah2(i)=0;
    else
        sigmah2(i)=alphah2(i)/abs(alphah2(i));
    end
end
for i=2:N
    if alphav2(i)==0
        sigmav2(i)=0;
    else
        sigmav2(i)=alphav2(i)/abs(alphav2(i));
    end
end
%Calculating the values of r
for i=2:N-2
    if (H_old(i+1)-H_old(i)==0)
        rh1(i)=0;
    else
        rh1(i)=(H_old(i+1+sigmah1(i))-H_old(i+sigmah1(i)))/(H_old(i+1)-H_old(i));
    end
end
for i=2:N-2
    if (U_old(i+1)-U_old(i)==0)
        rv1(i)=0;
    else
        rv1(i)=(U_old(1+i+sigmav1(i))-U_old(i+sigmav1(i)))/(U_old(i+1)-U_old(i));
    end
end
for i=3:N-2
    if (H_old(i)-H_old(i-1)==0)
        rh2(i)=0;
    else
        rh2(i)=(H_old(i+sigmah2(i))-H_old(i-1+sigmah2(i)))/(H_old(i)-H_old(i-1));
    end
end
for i=3:N-2
    if (U_old(i)-U_old(i-1)==0)
        rv2(i)=0;
    else
        rv2(i)=(U_old(i+sigmav2(i))-U_old(i-1+sigmav2(i)))/(U_old(i)-U_old(i-1));
    end
end
%Calculate values of G
for i=3:N-2
    Gh1(i)=(rh1(i)+abs(rh1(i)))/(1+rh1(i));
end
for i=3:N-2
    Gh2(i)=(rh2(i)+abs(rh2(i)))/(1+rh2(i));
end
for i=3:N-2
    Gv1(i)=(rv1(i)+abs(rv1(i)))/(1+rv1(i));
end
for i=3:N-2
    Gv2(i)=(rv2(i)+abs(rv2(i)))/(1+rv2(i));
end
%Calculate the values of Flux Limiter phi(i+1/2) and phi(i-1/2)
for i=3:N-2
    phih1(i)=((Gh1(i)/2)*(abs(alphah1(i))+c*(alphah1(i).^2))-alphah1(i))*(H_old(i+1)-H_old(i));
end
for i=3:N-2
    phih2(i)=((Gh2(i)/2)*(abs(alphah2(i))+c*(alphah2(i).^2))-alphah2(i))*(H_old(i)-H_old(i-1));
end
for i=3:N-2
    phiv1(i)=((Gv1(i)/2)*(abs(alphav1(i))+c*(alphav1(i).^2))-alphav1(i))*(U_old(i+1)-U_old(i));
end
for i=3:N-2
    phiv2(i)=((Gv2(i)/2)*(abs(alphav2(i))+c*(alphav2(i).^2))-alphav2(i))*(U_old(i)-U_old(i-1));
end
%Implementing Runge-Kutta Method
for i=1:N
    H1(i)=H_old(i); U1(i)=U_old(i);
end
for i=2:N-1
    H2(i)=H_old(i)-((0.25*c*0.5.*(H1(i+1).*U1(i+1)-H1(i-1).*U1(i-1)))/(H1(i)));
    U2(i)=U_old(i)-(0.25*c*0.5.*(((U1(i+1).^2/H1(i+1))+0.5*g*(H1(i+1).^2))-((U1(i-1).^2./H1(i-
1))+0.5*g*H1(i-1).^2)));
end
H2(1)=H_old(1); U2(1)=U_old(1);
H2(N)=H_old(N-1); U2(N)=U_old(N-1);
for i=2:N-1
    H3(i)=H_old(i)-(((1/3)*c*0.5.*(H2(i+1).*U2(i+1)-H2(i-1).*U2(i-1)))./(H2(i)));
    U3(i)=U_old(i)-((1/3)*c*0.5.*(((U2(i+1).^2./H2(i+1))+0.5*g*H2(i+1).^2)-((U2(i-1).^2./H2(i-
1))+0.5*g*H2(i-1).^2)));
end
H3(1)=H_old(1); U3(1)=U_old(1);
H3(N)=H_old(N-1); U3(N)=U_old(N-1);
for i=2:N-1
    H4(i)=H_old(i)-((0.5*c*0.5.*(H3(i+1).*U3(i+1)-H3(i-1).*U3(i-1)))./(H3(i)));
    U4(i)=U_old(i)-(0.5*c*0.5.*(((U3(i+1).^2./H3(i+1))+0.5*g*H3(i+1).^2)-((U3(i-1).^2./H3(i-
1))+0.5*g*H3(i-1).^2)));
end
H4(1)=H_old(1); U4(1)=U_old(1);
H4(N)=H_old(N-1); U4(N)=U_old(N-1);
for i=2:N-1
    H_new(i)=H_old(i)-0.25*(c*0.5.*(H4(i+1).*U4(i+1)-H4(i-1).*U4(i-1))./(H4(i)));
    U_new(i)=U_old(i)-0.25*(c*0.5.*(((U4(i+1).^2./H4(i+1))+0.5*g*H4(i+1).^2)-((U4(i-
1).^2./H4(i-1))+0.5*g*H4(i-1).^2)));
end
H_new(1)=H_old(1); U_new(1)=U_old(1);
H_new(N)=H_old(N-1); U_new(N)=U_old(N-1);
%Adding the flux limiter
for i=1:N
    H_new(i)=H_new(i)-0.5*c.*(phih1(i)-phih2(i));
    U_new(i)=U_new(i)-0.5*c.*(phiv1(i)-phiv2(i));
end
for i=1:N
    H_old(i)=H_new(i);
    U_old(i)=U_new(i);
end
Unewfinal=U_new./H_new;
end
figure(1);
plot(H_new);
title('Plot for water flow by Runge-Kutta with TVD');
xlabel('x in m'); ylabel('H in m');
figure(2);
plot(Unewfinal);
title('Plot for velocity by Runge-Kutta with TVD');
xlabel('x in m'); ylabel('Velocity U in m/s');
```

A-3 WENO METHOD:

```
%WENO Scheme
function Weno_trial
clear all; close all; clc;
N=101;
L=100;
delta_x=L/(N-1);
g=9.81;
delta_t=0.001;
t=4;

% intial values of h and v
for i=1:N
    for l=1
        if i*delta_x<50
            Qold(l,i)=10;
            Qold(l+1,i)=0;
        else
            Qold(l,i)=1;
            Qold(l+1,i)=0;
        end
    end
end
tinitial=0;
z=0;
while tinitial<=t
    tinitial=tinitial+delta_t;
    z=z+1;
    [delta_Eold]=Weno(Qold, delta_t);
    for l=1:2
        for i=4:N-2
            Q2(l,i)=Qold(l,i)-(1/4)*(delta_t/delta_x)*delta_Eold(l,i);
        end
        Q2(l,1:3)=Q2(l,4);
        Q2(l,N-1:N)=Q2(l,N-2);
    end
    for i=1:N
        c(i)=sqrt(g.*Q2(1,i));
    end
    %Calculating eigen values
    for i=1:N
        eigen(1,i)=(Q2(2,i)./Q2(1,i))+c(i);
        eigen(2,i)=(Q2(2,i)./Q2(1,i))-c(i);
    end
    d(z)=max(max(abs(eigen)));
    [delta_E1]=Weno(Q2,delta_t);
    for l=1:2
        for i=4:N-2
            Q3(l,i)=Qold(l,i)-(1/3)*(delta_t/delta_x)*delta_E1(l,i);
        end
        Q3(l,1:3)=Q3(l,4);
        Q3(l,N-1:N)=Q3(l,N-2);
    end
    for i=1:N
        c(i)=sqrt(g.*Q3(1,i));
    end
    for i=1:N
        eigen(1,i)=(Q3(2,i)./Q3(1,i))+c(i);
        eigen(2,i)=(Q3(2,i)./Q3(1,i))-c(i);
    end
    e(z)=max(max(abs(eigen)));
    [delta_E2]=Weno(Q3,delta_t);
    for l=1:2
        for i=4:N-2
            Q4(l,i)=Qold(l,i)-(1/2)*(delta_t/delta_x)*delta_E2(l,i);
        end
        Q4(l,1:3)=Q4(l,4);
        Q4(l,N-1:N)=Q4(l,N-2);
    end
    for i=1:N
        c(i)=sqrt(g.*Q4(1,i));
    end
    for i=1:N
        eigen(1,i)=(Q4(2,i)./Q4(1,i))+c(i);
        eigen(2,i)=(Q4(2,i)./Q4(1,i))-c(i);
    end
    f(z)=max(max(abs(eigen)));
    [delta_E3]=Weno(Q4,delta_t);
    for l=1:2
        for i=4:N-2
            Qnew(l,i)=Qold(l,i)-(delta_t/delta_x)*delta_E3(l,i);
        end
        Qnew(l,1:3)=Qnew(l,4);
        Qnew(l,N-1:N)=Qnew(l,N-2);
    end
    Qold=Qnew;
    for i=1:N
        Unew(1,i)=Qnew(2,i)/Qold(1,i);
    end
end
figure(1);
plot(Qnew(1,:));
title('Plot for Flow of Water and Height Variation using WENO');
xlabel('x in m'); ylabel('H in m');
figure(2);
plot(Unew(1,:));
title('Plot for Velocity of Water using WENO');
xlabel('x in m'); ylabel('U in m/s');
end


function [df]= Weno(Qold,delta_t)
N=101;
g=9.81;
for i=1:N
    cold(i)=sqrt(g.*Qold(1,i));
end
%Calculating eigen values
for i=1:N
        eigen(1,i)=(Qold(2,i)./Qold(1,i))+cold(i);
        eigen(2,i)=(Qold(2,i)./Qold(1,i))-cold(i);
end
beta=max(abs(eigen));
for i=1:N
    E(1,i)=Qold(2,i);
    E(2,i)=((Qold(2,i)^2/Qold(1,i))+(0.5*g*(Qold(1,i)^2)));
end
```

```
weight0=1/10; weight1=6/10; weight2=3/10;
for l=1:2
    for i=3:N-2
        b0(l,i)=13/12*(E(l,i-2)-2*E(l,i-1)+E(l,i))^2+1/4*(3*E(l,i-2)-4*E(l,i-1)+E(l,i))^2;
        b1(l,i)=13/12*(E(l,i-1)-2*E(l,i)+E(l,i+1))^2+1/4*(3*E(l,i-1)-E(l,i+1))^2;
        b2(l,i)=13/12*(E(l,i)-2*E(l,i+1)+E(l,i+2))^2+1/4*(E(l,i)-4*E(l,i+1)+E(l,i+2))^2;
    end
end
ep=10^-6;
for l=1:2
    for i=3:N-2
        wir0(l,i)=weight0/((ep+b0(l,i))^2+(ep+b1(l,i))^2+(ep+b2(l,i))^2);
        wir1(l,i)=weight1/((ep+b0(l,i))^2+(ep+b1(l,i))^2+(ep+b2(l,i))^2);
        wir2(l,i)=weight2/((ep+b0(l,i))^2+(ep+b1(l,i))^2+(ep+b2(l,i))^2);
    end
end
for l=1:2
    for i=3:N-2
        wzero(l,i)=wir0(l,i)/(wir0(l,i)+wir1(l,i)+wir2(l,i));
        wone(l,i)=wir1(l,i)/(wir0(l,i)+wir1(l,i)+wir2(l,i));
        wtwo(l,i)=wir2(l,i)/(wir0(l,i)+wir1(l,i)+wir2(l,i));
    end
end
for l=1:2
    for i=3:N-2
        s0(l,i)=1/3*E(l,i-2)-7/6*E(l,i-1)+11/6*E(l,i);
        s1(l,i)=-1/6*E(l,i-1)+5/6*E(l,i)+1/3*E(l,i+1);
        s2(l,i)=1/3*E(l,i)+5/6*E(l,i+1)-1/6*E(l,i+2);
    end
end
for l=1:2
    for i=3:N-2
        Epos(l,i)=wzero(l,i)*s0(l,i)+wone(l,i)*s1(l,i)+wtwo(l,i)*s2(l,i);
    end
end
for l=1:2
    for i=4:N-1
        c0(l,i)=13/12*(E(l,i-3)-2*E(l,i-2)+E(l,i-1))^2+1/4*(3*E(l,i-3)-4*E(l,i-2)+E(l,i-1))^2;
        c1(l,i)=13/12*(E(l,i-2)-2*E(l,i-1)+E(l,i))^2+1/4*(3*E(l,i-2)-E(l,i))^2;
        c2(l,i)=13/12*(E(l,i-1)-2*E(l,i)+E(l,i+1))^2+1/4*(E(l,i-1)-4*E(l,i)+E(l,i+1))^2;
    end
end
for l=1:2
    for i=4:N-1
        vir0(l,i)=weight0/((ep+c2(l,i))^2+(ep+c1(l,i))^2+(ep+c0(l,i))^2);
        vir1(l,i)=weight1/((ep+c2(l,i))^2+(ep+c1(l,i))^2+(ep+c0(l,i))^2);
        vir2(l,i)=weight2/((ep+c2(l,i))^2+(ep+c1(l,i))^2+(ep+c0(l,i))^2);
    end
end
for l=1:2
    for i=4:N-1
        vzero(l,i)=vir0(l,i)/(vir0(l,i)+vir1(l,i)+vir2(l,i));
        vone(l,i)=vir1(l,i)/(vir0(l,i)+vir1(l,i)+vir2(l,i));
        vtwo(l,i)=vir2(l,i)/(vir0(l,i)+vir1(l,i)+vir2(l,i));
    end
end
for l=1:2
    for i=4:N-1
        t0(l,i)=1/3*E(l,i-3)-7/6*E(l,i-2)+11/6*E(l,i-1);
        t1(l,i)=-1/6*E(l,i-2)+5/6*E(l,i-1)+1/3*E(l,i);
        t2(l,i)=1/3*E(l,i-1)+5/6*E(l,i)-1/6*E(l,i+1);
    end
end
for l=1:2
    for i=4:N-1
        Eneg(l,i)=vzero(l,i)*t0(l,i)+vone(l,i)*t1(l,i)+vtwo(l,i)*t2(l,i);
    end
```

```
end
for i=3:N-2
    Fplushalf(1,i)=0.5*(Epos(1,i)+Eneg(1,i+1))-0.5*max(beta(i),beta(i+1))*(Qold(1,i+1)-Qold(1,i));
    Fplushalf(2,i)=0.5*(Epos(2,i)+Eneg(2,i+1))-0.5*max(beta(i),beta(i+1))*(Qold(2,i+1)-Qold(2,i));
   % Fplushalf(3,i)=0.5*(Epos(3,i)+Eneg(3,i+1))-0.5*max(beta(i),beta(i+1))*(Qold(3,i+1)-Qold(3,i));
end
for i=4:N-1
    Fminhalf(1,i)=0.5*(Eneg(1,i)+Epos(1,i-1))-0.5*max(beta(i),beta(i-1))*(Qold(1,i)-Qold(1,i-1));
    Fminhalf(2,i)=0.5*(Eneg(2,i)+Epos(2,i-1))-0.5*max(beta(i),beta(i-1))*(Qold(2,i)-Qold(2,i-1));
    %Fminhalf(3,i)=0.5*(Eneg(3,i)+Epos(3,i-1))-0.5*max(beta(i),beta(i-1))*(Qold(3,i)-Qold(3,i-1));
end
for k=1:2
    for i=4:N-2
        df(k,i)=(Fplushalf(k,i)-Fminhalf(k,i));
    end
end
end
```