MoSV-1D   
(MOmentum conservation Saint-Venant 1 Dimension)

**Manual Book**

****

By

© Telkom University

Version 1.0

|  |  |  |
| --- | --- | --- |
| By | : | Telkom University |
| Author | : | 1. Hendro Pratama Saragih 2. Dr. Didit Adytia 3. Dede Tarwidi, M.Sc. |
| Address | : | Jl. Telekomunikasi No.1, Terusan Buah Batu, Kec. Dayeuhkolot Bandung, Jawa Barat 40257, Indonesia |
| Alamat e-mail | : | adytia@telkomuniversity.ac.id |
| Laman website | : | https://telkomuniversity.ac.id/ |



Copyright © 2020 Telkom University

# **Contents**

[**Contents** 1](#_Toc39842078)

[**Figure Contents** 2](#_Toc39842079)

[**1.** **Introduction** 3](#_Toc39842080)

[**2.** **Requirements** 3](#_Toc39842081)

[**2.1.** **System Requirements** 3](#_Toc39842082)

[**2.2.** **Program Requirements** 3](#_Toc39842083)

[**3.** **Installing** 3](#_Toc39842084)

[**4.** **Graphic User Interface (GUI)** 6](#_Toc39842085)

[**4.1.** **Main GUI** 6](#_Toc39842086)

[**4.2.** **Xspace and Time** 7](#_Toc39842087)

[**4.3.** **Test Case** 7](#_Toc39842088)

[**4.4.** **Save Test Case** 8](#_Toc39842089)

[**4.5.** **Save Animation** 8](#_Toc39842090)

[**4.6.** **Button** 8](#_Toc39842091)

[**5.** **Operating** 9](#_Toc39842092)

[**5.1.** **Running Animation** 9](#_Toc39842093)

[**5.1.1.** **Filled Data** 9](#_Toc39842094)

[**5.1.2.** **Result** 10](#_Toc39842095)

[**5.2.** **Additional Features** 11](#_Toc39842096)

[**5.2.1.** **Save Test Case** 11](#_Toc39842097)

[**5.2.2.** **Load Test Case** 11](#_Toc39842098)

[**5.2.3.** **Save Animation** 12](#_Toc39842099)

[**5.2.4.** **Run Animation** 13](#_Toc39842100)

[**5.2.5.** **Clear Input Parameter** 13](#_Toc39842101)

[**5.2.6.** **Close GUI** 14](#_Toc39842102)

[**6.** **Test Result** 14](#_Toc39842103)

[**6.1.** **Solitary Wave** 14](#_Toc39842104)

[**6.2.** **Dam Break** 15](#_Toc39842105)

[**7.** **Source Code** 15](#_Toc39842106)

# **Figure Contents**

[Illustration 1 Setup MoSV 1D 4](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841672)

[Illustration 2 Installation 1 4](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841673)

[Illustration 3 Installation 2 5](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841674)

[Illustration 4 Installation 3 5](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841675)

[Illustration 5 Installation 4 6](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841676)

[Illustration 6 Homepage 6](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841677)

[Illustration 7 Xspace and Time 7](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841678)

[Illustration 8 Test Case 7](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841679)

[Illustration 9 Save Test Case 8](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841680)

[Illustration 10 Save Animation 8](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841681)

[Illustration 11 Button 8](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841682)

[Illustration 12 Filled Data 9](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841683)

[Illustration 13 Result Solitary Wave 10](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841684)

[Illustration 14 Result Dam Break 10](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841685)

[Illustration 15 Save & View Test Case 11](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841686)

[Illustration 16 Load Test Case 11](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841687)

[Illustration 17 Pop Up Warning 12](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841688)

[Illustration 18 Save & View Animation 12](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841689)

[Illustration 19 Run Animation 13](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841690)

[Illustration 20 Clear Input Parameter 13](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841691)

[Illustration 21 Pop Up Quit 14](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841692)

[Illustration 22 Solitary Wave 14](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841693)

[Illustration 23 Dam Break 15](https://d.docs.live.net/b0bee25d5c9adcde/_TA-GEOSCIENCE/_FINAL_1/ManualBook_MoSV-1D_English.docx#_Toc39841694)

# **Introduction**

This document is a guide to using and running MoSV 1D (Momentum Conservation Saint-Venant 1 Dimension) software. MoSV 1D software is based on the One Dimensional Shallow Water Equation (SWE 1D) model which is implemented numerically using the Finite Volume method with a staggered-grid scheme. The SWE equation and its numerical implementation, have been published in the following Journals and Proceedings:

1. Adytia, D. (2019). Momentum Conservative Scheme for Simulating Wave Runup and Underwater Landslide. Indonesian Journal on Computing (Indo-JC), 4(1), 29. https://doi.org/10.21108/indojc.2019.4.1.250
2. Adytia, D., Husrin, S., & Latifah, A. L. (2019). Dissipation of Solitary Wave Due To Mangrove Forest: A Numerical Study by Using Non-Dispersive Wave Model. ILMU KELAUTAN: Indonesian Journal of Marine Sciences, 24(1), 41. https://doi.org/10.14710/ik.ijms.24.1.41-50
3. Alfikri, M. Z., Adytia, D., & Subasita, N. (2019). Shock capturing staggered grid scheme for simulating dam-break flow and runup. Journal of Physics: Conference Series, 1192(1). https://doi.org/10.1088/1742-6596/1192/1/012041

# **Requirements**

The requirements needed to run MoSV 1D are as follows :

## **System Requirements**

* + 1. **Windows 10 All Version**

Windows 10 is the operating system needed to run this application. This operating system is used because it is considered compatible on every user's device when using this application.

## **Program Requirements**

* + 1. **MoSV 1D Setup**

MoSV Setup.exe is a program that needs to be installed to be able to run the MoSV 1D application. This application program is in the form of setup and is the main file for installation..

# **Installing**

Before using the MoSV 1D application, we must first install the application. Following are the steps to install the MoSV 1D application :

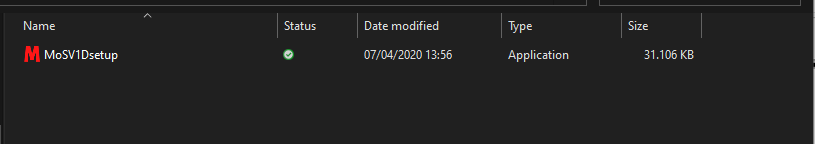
* 1. First double-click the MoSV 1D setup file and press **Run Administrator**.

Illustration 1 Setup MoSV 1D

* 1. Secondly click **Next**, the installation has been saved default in storage C.

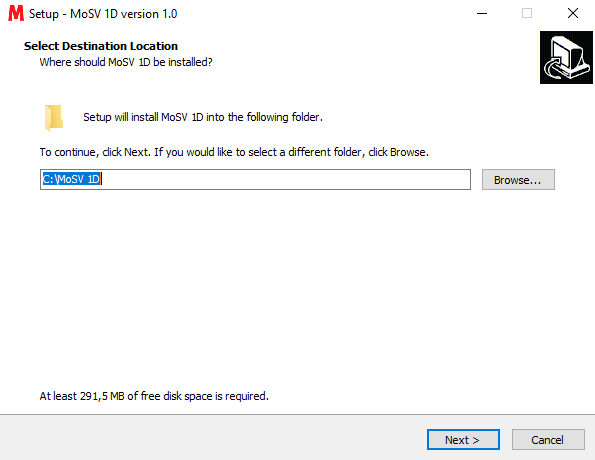


Illustration 2 Installation 1

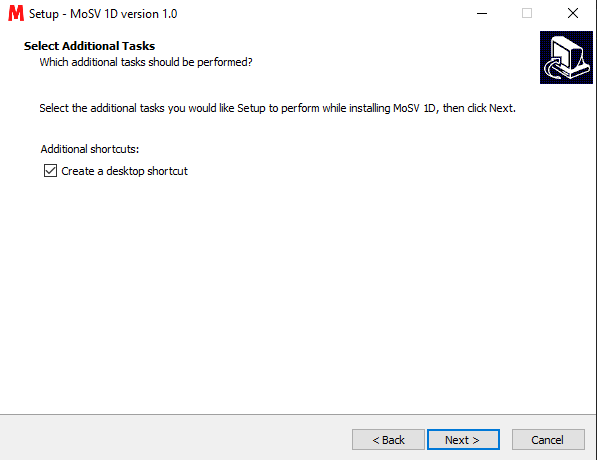
* 1. Third click **Next** and check additional shortcuts to create applications on the your desktop.

Illustration 3 Installation 2

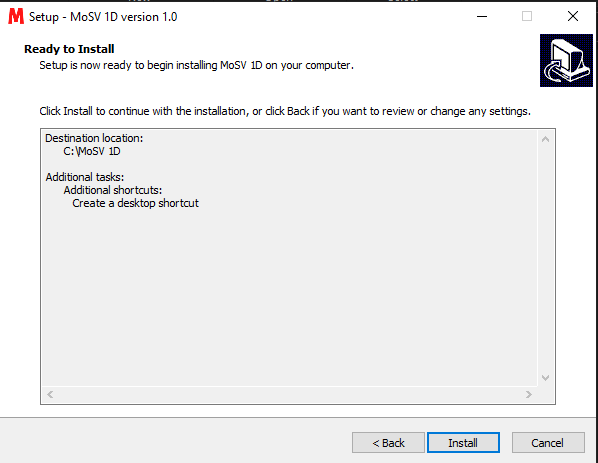
* 1. Fourth, press **Install** to install the MoSV 1D application.

Illustration 4 Installation 3

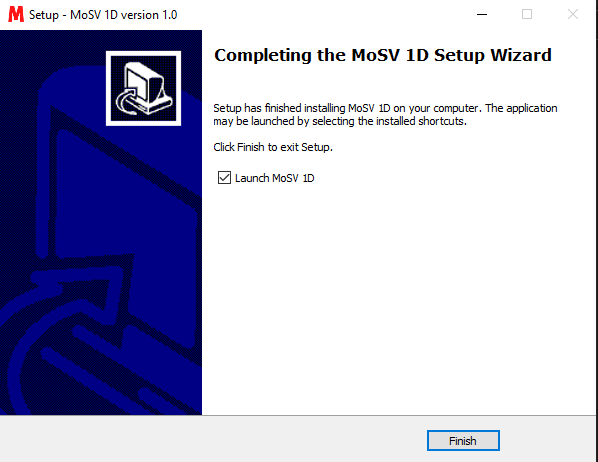
* 1. Finally press **Finish** when the installation process is complete.

Illustration 5 Installation 4

# **Graphic User Interface (GUI)**

The following are the parts of the MoSV 1D GUI (Graphical User Interface) :

## **Main GUI**

Illustration 6 Homepage

The picture above is the main display of the MoSV 1D application, there are 2 labels for input parameter namely Xspace & Time and Test Case. Then have several input entries and buttons.

## **Xspace and Time**

Illustration 7 Xspace and Time

Xspace and Time have 2 labels namely Xspace and Time, each of which has a different entry. Fill in all data in the provided entry fields using integer or float format. The Xspace section in illustration 7 contains three entry fields that must be filled. Xmin and Xmax in meters are the position of Xawal and Xakhir. For dx entries, it is a value that has a meter.

## **Test Case**

Illustration 8 Test Case

The Test Case section in illustration 8, has 2 choices namely Solitary Wave and Dam Break. In the Solitary Wave option there is A0 in meters which means amplitude, xC in meters means position from the peak of Soliton and xWide is width in meters. Then the second option Dam Break is Width which is the width has meters and Height is height has units of meters

## **Save Test Case**

Illustration 9 Save Test Case

The Save Test Case section in illustration 9 has a function to save and load the parameter file that was filled in earlier. Before saving parameters, the user must fill in the file name. The file that is saved and loaded is in the .txt extension.

## **Save Animation**

Illustration 10 Save Animation

The Save Animation section in illustration 10 has a function for saving files in the form of animation. In storing the animation file has an extension that is gif

## **Button**

Illustration 11 Button

In the illustration section 11 has 2 buttons namely the RUN button and the CLEAR button. Run button functions to run the animation in the application based on parameters that have been filled. The CLEAR key functions to delete entries in the parameters, wave, bathymetry and time sections, which aim to speed up work in repeating new test cases and animations.

# **Operating**

Here the example by running the software. There are 2 conditions to use this software : Running Animation and Save Test Case & Animation

## **Running Animation**

### **Filled Data**

Illustration 12 Filled Data

First we fill all the data entries in the domain, wave, bathymetry, time and dam break sections. Then when all the data has been filled, press the green RUN button. Then wait for the animated image to appear on the blank canvas. The results of the animation of each parameter, will display the animation form as illustrated 16 (Result).

### **Result**

The following is the result of each parameter, will display the results of the animation as below :

#### **5.1.2.1 Result Solitary Wave**

Illustration 13 Result Solitary Wave

#### **5.1.2.2 Result Dam Break**

Illustration 14 Result Dam Break

## **Additional Features**

### **Save Test Case**

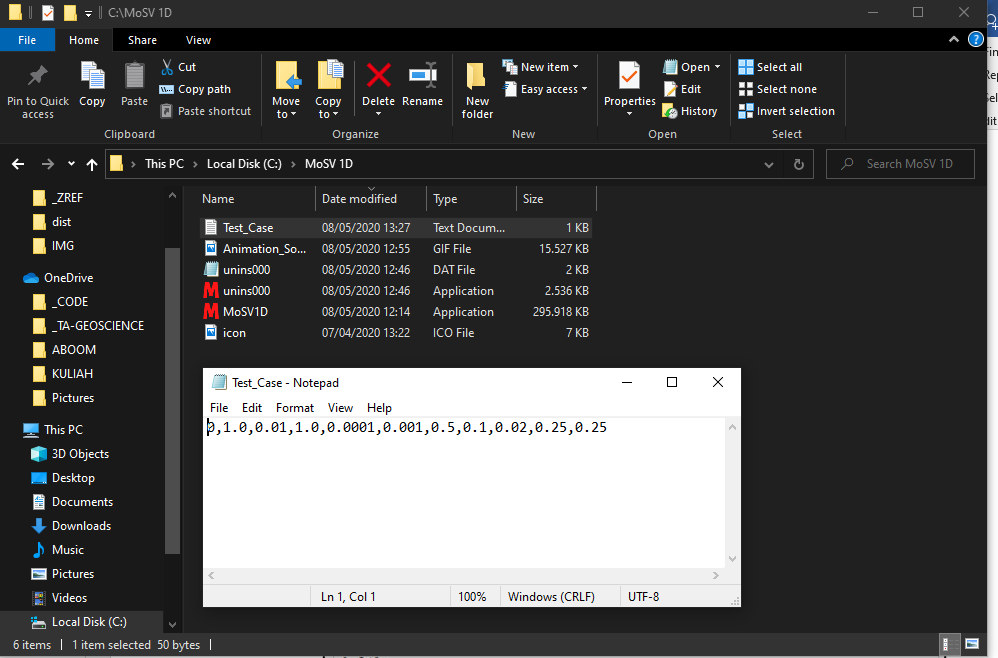
Save all 1D MoSV parameters in the form of a text (.txt) file by giving the file name first. The following is an example of a test case file with the name TestCase\_1 that has been saved.

Illustration 15 Save & View Test Case

### **Load Test Case**

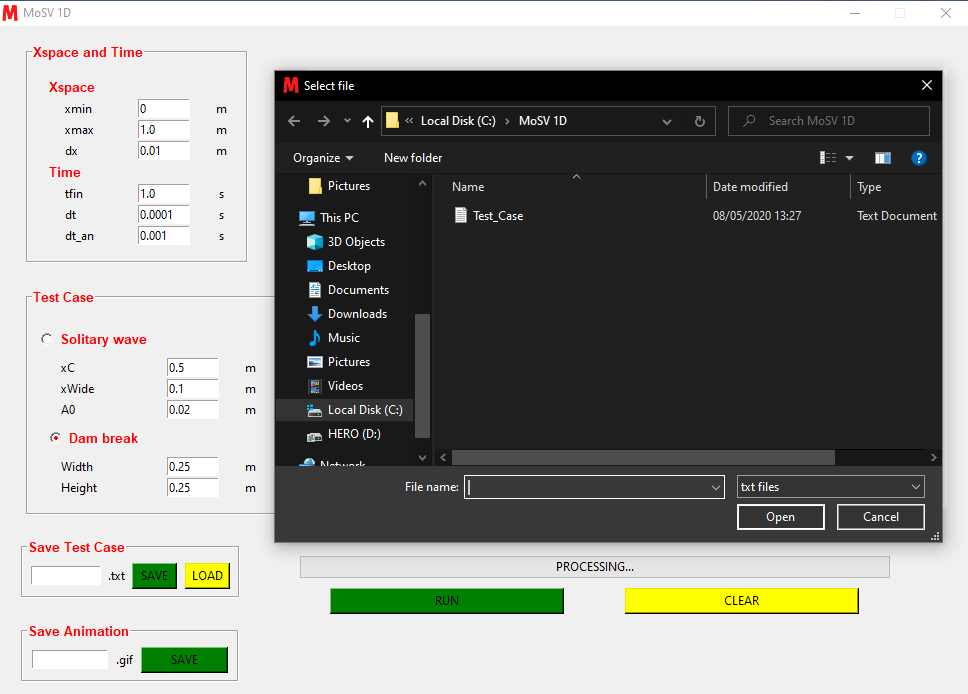
Load the test case file that has been previously stored and then retrieve all the data contained in the file to be filled in to the GUI parameter entry and then it will be a requirement to run the animation.

Illustration 16 Load Test Case

### **Save Animation**

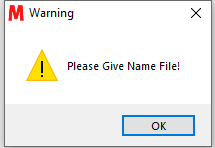
Save the results of the MoSV 1D animation in the form of a gif (.gif) file by giving the file name first. When pressing the save button before giving a name a pop-up will appear as follows :

Illustration 17 Pop Up Warning

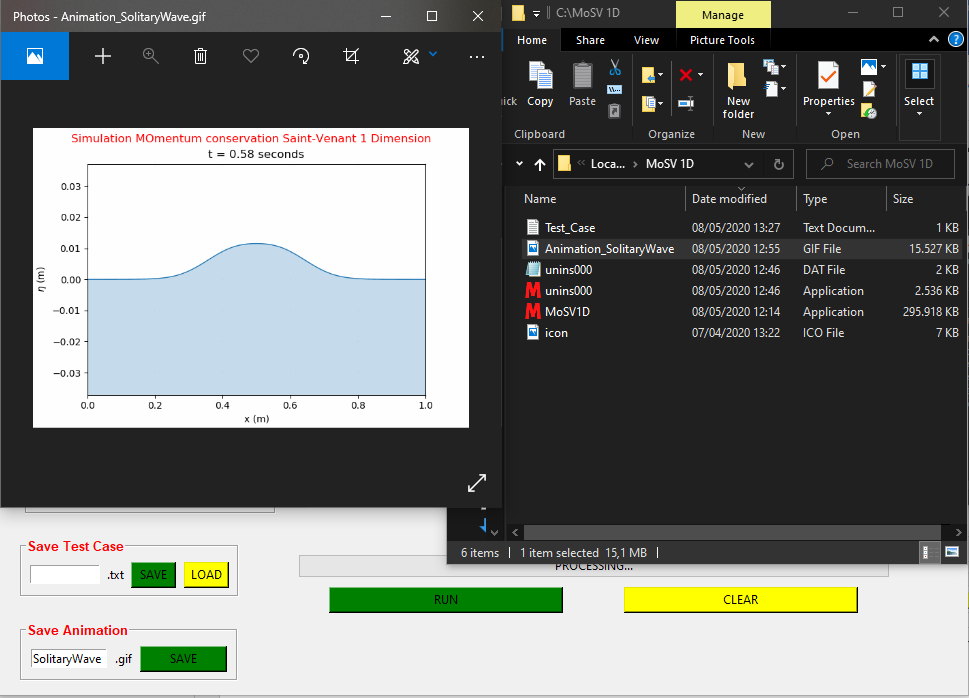
The following is an example of an animation file with the name Animation\_1 that has been saved.

Illustration 18 Save & View Animation

### **Run Animation**

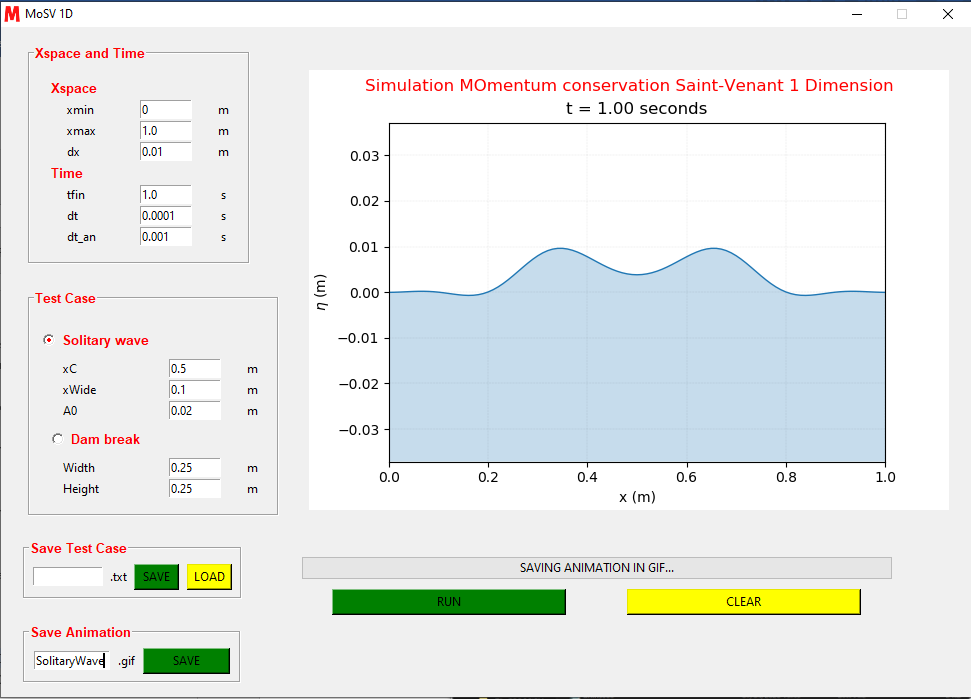
Running animation with a state where all parameters have been filled and pressing the RUN button so that the animation will appear in the form of canvas in the GUI.

Illustration 19 Run Animation

### **Clear Input Parameter**

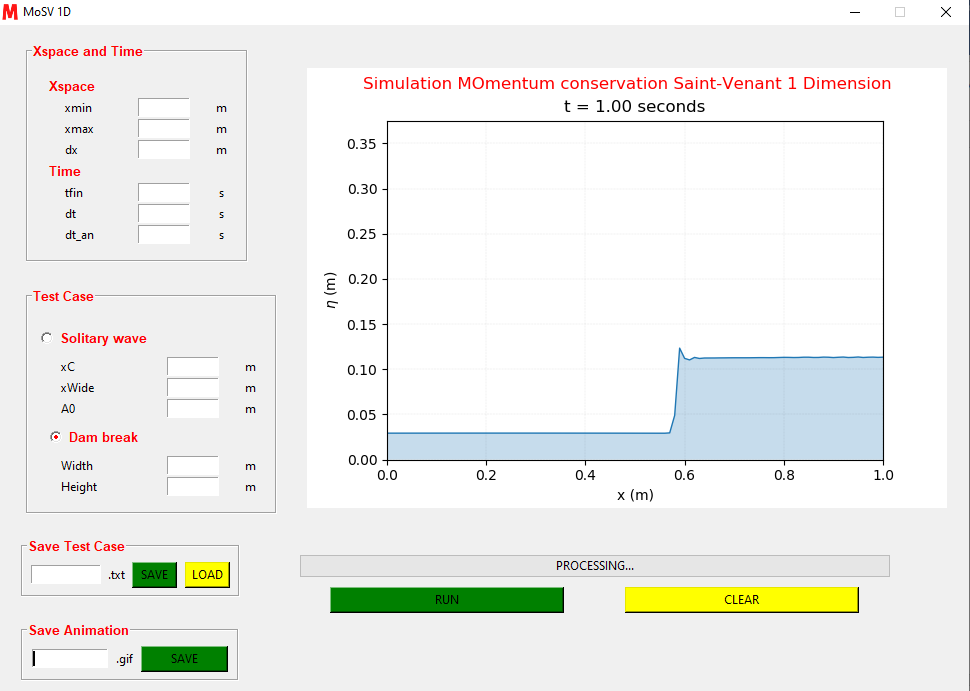
Remove the contents of all entry parameters in the Xspace, Time, Solitary Wave and dam break except. This section aims to make it easier for us to replace all parameters at once. Here is the result of clear animation:

Illustration 20 Clear Input Parameter

### **Close GUI**

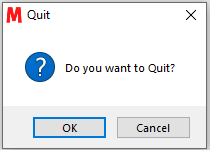
When pressing the close button on the top right, a Quit pop-up will appear. If the user asks OK then the application will exit, while pressing cancel returns the application. Here is a pop-up when pressing the close button on the application.

Illustration 21 Pop Up Quit

# **Test Result**

## **Solitary Wave**

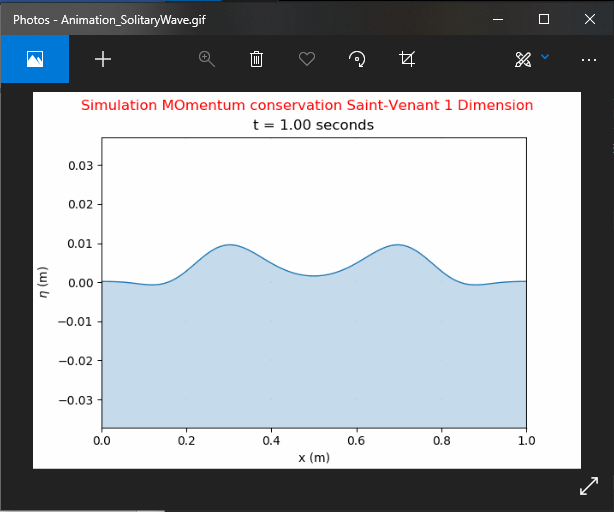
The following is the result of Solitary Wave based on input parameters as shown below :

Illustration 22 Solitary Wave

## **Dam Break**

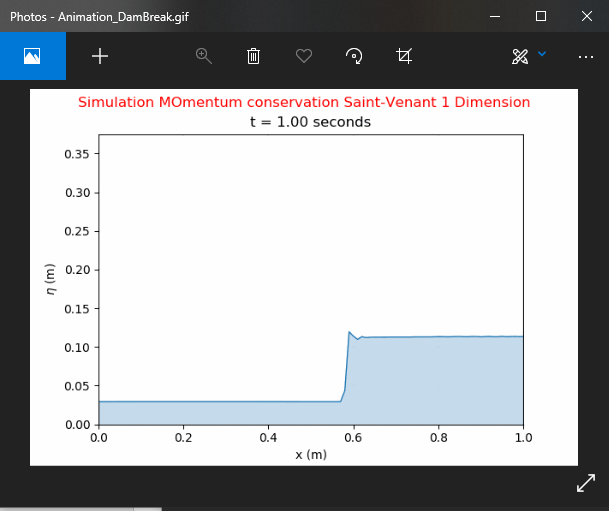
The following is the result of Dam Break based on input parameters as shown below :

Illustration 23 Dam Break

# **Source Code**

|  |
| --- |
| from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  from matplotlib.figure import Figure  import tkinter as tk  from tkinter import \*  from tkinter import ttk  import numpy as np  import matplotlib  matplotlib.use("TKAgg")  from tkinter import messagebox  from tkinter.filedialog import askopenfile  from tkinter.filedialog import askopenfilename  from tkinter.filedialog import asksaveasfilename  import matplotlib.pyplot as plt  import matplotlib.animation as animation  import math as ma  import time  root = tk.Tk()  root.geometry("970x670")  root.resizable(0,0)  # ===========================#  # EXTERNAL FUNCTION #  # ===========================#  def initial\_Condition(x, Nx, A0, xC, xwide, dam\_height, dam\_width, IC\_option) :  y = [0]\*(Nx+1)  if IC\_option == 'dam\_break' :  for i in range(0, Nx+1):  if x[i] <= dam\_width:  y[i] = dam\_height  else :  y[i] = 0.  elif IC\_option == 'solitary\_wave' :  y = A0\*np.exp((-((x-xC)/xwide)\*((x-xC)/xwide)))    return np.asarray(y)  def bathymetry(x, Nx, IC\_option) :  y = [0]\*(Nx+1)  if IC\_option == 'dam\_break' :  for i in range(0, Nx+1):  y[i] = 0. # flat bottom  elif IC\_option == 'solitary\_wave' :  hshall = 0.02  hC = 0.75  hwide = 0.1  y = 20 - hshall\*np.exp((-((x-hC)/hwide)\*((x-hC)/hwide)))  return np.asarray(y)  def call\_SWE(xleft, xright, Nx, dx, dt, dt\_anim, tfin, A0, xC, xwide, dam\_height, dam\_width,  IC\_option) :  # 0. INPUT PARAMETERS  # wave initial condition  x = [0]\*(Nx+1)  x = np.linspace(xleft, xright, Nx + 1)  eta\_list = list()  time\_list = list()  # parameters  g = 9.81  eps\_thin = 0.0001  # 1. PREPARATION  h = [0]\*(Nx+1) # bottom / bathymetry / topography  H = [0]\*(Nx+1) # total depth in FULL grid at time n  H\_new = [0]\*(Nx+1) # total depth in FULL grid at time n+1  H\_star = [0]\*(Nx+2) # total depth in 'half-grid'  H\_bar = [0]\*(Nx+2) # averaged total depth at HALF grid    u = [0]\*(Nx+2) # hor velocity in HALF grid at t=n+1  u\_new = [0]\*(Nx+2) # hor velocity in HALF grid at t=n+1  u\_star = [0]\*(Nx+1) # velocity in full grid  u\_du = [0]\*(Nx+2) # advection term in half-grid  q = [0]\*(Nx+2) # approx. for h\*u in HALF grid  q\_bar = [0]\*(Nx+1) # average of q1 in FULL grid  eta = [0]\*(Nx+1) # wave elevation in FULL grid at time n  eta\_new = [0]\*(Nx+1) # wave elevation in FULL grid at time n+1  Ht = [0]\*(Nx+1)    # 2. INITIAL CONDITION & BATHYMETRY  eta = initial\_Condition(x, Nx, A0, xC, xwide, dam\_height, dam\_width, IC\_option)  # 2.1. Initial condition  h = bathymetry(x, Nx, IC\_option) # 2.2. Bathymetry/bottom  t = 0  step = 0    H = h + eta # 2.3. total depth in full grid    # MAIN CALCULATION; TIME STEPPING  max\_iteration = ma.ceil(tfin/dt)  every = ma.ceil(dt\_anim/dt)  # ----------- TREATMENT FOR RUNUP ----------------------------------------  eta = np.maximum(eta, -h) # treatment for runup  H = eta + h # Water thickness  Ht = H[:]  ithin = np.where(H<eps\_thin) # find indices for dry area (runup/overtopping)  Ht[ithin] = eps\_thin # thin layer is only used when solving    while (t <= tfin) :  t = t + dt  step += 1  # UPDATE VARIABLES  H = eta + h # Water thickness  Ht = H[:]  ithin = np.where(H<eps\_thin) # find indices for dry area  Ht[ithin] = eps\_thin # thin layer is only used when solving    # 1. CONTINUITY EQ.  # 1.1. Prepare for UPWIND of q=h\*u  for j in range(1, Nx+1) :  if (u[j] >= 0) :  H\_star[j] = H[j-1]  else :  H\_star[j] = H[j]  q[j] = H\_star[j]\*u[j]  # Neumann condition  H\_star[0] = H\_star[1]  H\_star[Nx+1] = H\_star[Nx]  # no-flux through boundary, or zero velocity  q[0] = 0  q[Nx+1] = 0    for j in range(0, Nx+1) :  eta\_new[j] = eta[j] - (dt/dx)\*(q[j+1]-q[j])  H[j] = eta\_new[j] + h[j]  # Calculate q-bar (FLUX) for calculating (u-start upwind condition)  # CALCULATE q-bar  q\_bar[j] = 0.5\*(q[j] + q[j+1])  # UPWIND for u-star  if(q\_bar[j] >= 0) :  u\_star[j]=u[j]  else :  u\_star[j]=u[j+1]  # UPDATE H\_new  H\_new = h + eta\_new # eta at time n+1 % UPDATED total depth  # 2. MOMENTUM EQ.  # 2.1. Horizontal Momentum  for j in range(1, Nx+1) :  H\_bar[j] = 0.5\*(H[j] + H[j-1])  if(H\_bar[j] >= eps\_thin) : #for runup  # Advection term - wet dry procedure using momentum  conservative scheme  # Layer-1 (upper), HALF grid  u\_du[j] = (1./(H\_bar[j]\*dx))\*((q\_bar[j]\*u\_star[j] - q\_bar[j-  1]\*u\_star[j-1]) - (u[j]\*(q\_bar[j] - q\_bar[j-1])))  # Calc full eq. of 2nd dynamic eq. (Hydrostatic)  u\_new[j] = u[j] - (dt/dx\*g\*( eta\_new[j] - eta\_new[j-1] )) –  (dt\*u\_du[j]) #Layer-1  else : # if the total depth is not positive (no water) --> no velocity  u\_new[j] = 0  # hardwall boundary conditions  u\_new[0] = 0  u\_new[Nx+1] = 0  # updating values of H, eta, u  eta = eta\_new  H = H\_new  u = u\_new    if (step % every == 0) :  eta\_list.append(eta\_new[:])  time\_list.append(t)  return (eta\_list, time\_list)  # ===========================#  # EXTERNAL FUNCTION #  # ===========================#  def on\_closing():  if messagebox.askokcancel("Quit", "Do you want to Quit?"):  root.destroy()  class Window(tk.Frame):  def \_\_init\_\_(self, master):  tk.Frame.\_\_init\_\_(self, master)  self.master = master  self.init\_window()  self.running = False  self.ani = None  def start(self):  # clear list  self.eta\_list.clear()  self.time\_list.clear()  self.Axleft=float(self.xleft.get())  self.Axright=float(self.xright.get())  self.Adx=float(self.dx.get())  self.ANx= ma.ceil((self.Axright - self.Axleft)/self.Adx)  self.Atfin=float(self.tfin.get())  self.Adam\_width=float(self.dam\_width.get())  self.Adam\_height=float(self.dam\_height.get())    self.AA0=float(self.A0.get())  self.AxC=float(self.xC.get())  self.Axwide=float(self.xwide.get())    self.Adt=float(self.dt.get())  self.Adt\_anim = float(self.dt\_anim.get())    self.x = [0]\*(self.ANx+1)  self.x = np.linspace(self.Axleft, self.Axright, self.ANx + 1)  self.eta\_list, self.time\_list = call\_SWE(self.Axleft, self.Axright, self.ANx, self.Adx,  self.Adt, self.Adt\_anim, self.Atfin, self.AA0, self.AxC, self.Axwide,  self.Adam\_height, self.Adam\_width, self.IC\_option)  self.eta\_len = len(self.eta\_list)  if self.IC\_option == 'dam\_break' :  self.ymax = 1.5\*np.amax(self.eta\_list[1][:])  self.ymin = 0.0  else :  self.ymax = 2\*np.amax(self.eta\_list[1][:])  self.ymin = -self.ymax  fig, (ax1) = plt.subplots(1, 1, figsize=(6.4, 4.4))  fig.suptitle('Simulation MOmentum conservation Saint-Venant 1 Dimension',  color='r', fontsize=12)  def animate(i):  ax1.cla()  ax1.set\_title("t = %3.2f seconds" %self.time\_list[i])  ax1.fill\_between(self.x, self.ymin, self.eta\_list[i][:], alpha=0.25)  ax1.axis((self.Axleft, self.Axright,self.ymin, self.ymax))  ax1.grid(True)  ax1.grid(color='black', linestyle='-.', linewidth=0.2, alpha=0.2)  ax1.set(xlabel='x (m)', ylabel=r'$\eta$' + ' (m)')  ax1.plot(self.x, self.eta\_list[i][:], lw=1)  self.canvas = FigureCanvasTkAgg(fig, master=self.frame8)  self.canvas.get\_tk\_widget().grid(column=0,row=0, padx=10)  self.ani = animation.FuncAnimation(fig, animate, np.arange(1, self.eta\_len),  interval=25, blit=False, repeat=False)  # self.running = True  # self.B5.config(text='RUN.')  self.ani.\_start()    def on\_click(self):  # if self.ani is None:  self.progressBar['maximum'] = 100  for i in range(101):  self.style.configure('text.Horizontal.TProgressbar',  text='PROCESSING...')  time.sleep(0.01)  self.progressBar["value"] = i  self.progressBar.update()  self.progressBar["value"] = 0  return self.start()  def clear\_text(self):  self.E0.delete(0, 'end')  self.E1.delete(0, 'end')  self.E2.delete(0, 'end')  self.E3.delete(0, 'end')  self.E4.delete(0, 'end')  self.E5.delete(0, 'end')  self.E6.delete(0, 'end')  self.E7.delete(0, 'end')  self.E8.delete(0, 'end')  self.E9.delete(0, 'end')  self.E10.delete(0, 'end')  def validate\_float(self, action, index, value\_if\_allowed,  prior\_value, text, validation\_type, trigger\_type, widget\_name):  # action=1 -> insert  if(action=='1'):  if text in value\_if\_allowed:  try:  float(value\_if\_allowed)  return True  except ValueError:  return False  else:  return False  else:  return True  def save(self):  file = self.Ename.get()  param0=self.E0.get()  param1=self.E1.get()  param2=self.E2.get()  param3=self.E3.get()  param4=self.E4.get()  param5=self.E5.get()  param6=self.E6.get()  param7=self.E7.get()  param8=self.E8.get()  param9=self.E9.get()  param10=self.E10.get()  with open(file + '.txt', 'w') as file\_object:  file\_object.write(param0)  file\_object.write(',')  file\_object.write(param1)  file\_object.write(',')  file\_object.write(param2)  file\_object.write(',')  file\_object.write(param3)  file\_object.write(',')  file\_object.write(param4)  file\_object.write(',')  file\_object.write(param5)  file\_object.write(',')  file\_object.write(param6)  file\_object.write(',')  file\_object.write(param7)  file\_object.write(',')  file\_object.write(param8)  file\_object.write(',')  file\_object.write(param9)  file\_object.write(',')  file\_object.write(param10)  def loadfile(self):  filename = filedialog.askopenfilename(title = "Select file",filetypes = (("txt  files","\*.txt"),("all files","\*.\*")))  print(filename)  readfile = open(filename, "r")  for line in readfile:  Type = line.split(",")  self.xleft.set(Type[0])  self.xright.set(Type[1])  self.dx.set(Type[2])  self.tfin.set(Type[3])  self.dt.set(Type[4])  self.dt\_anim.set(Type[5])  self.xC.set(Type[6])  self.xwide.set(Type[7])  self.A0.set(Type[8])  self.dam\_width.set(Type[9])  self.dam\_height.set(Type[10])  def save\_animate\_gif(self):  file\_name\_gif = self.Aniname\_gif.get()  if len(self.Aniname\_gif.get())==0:  messagebox.showwarning("Warning", "Run First & Give The File  Name!")  else:  self.progressBar['maximum'] = 100  self.style.configure('text.Horizontal.TProgressbar', text='SAVING  ANIMATION IN GIF...')  for i in range(101):  time.sleep(0.005)  self.progressBar["value"] = i  self.progressBar.update()  self.progressBar["value"] = 0  animation\_gif = self.ani.save(file\_name\_gif+'.gif',  writer='imagemagick', extra\_args=['-vcodec', 'libx264'])  return animation\_gif    def ICselection(self):  if self.varIC.get() == 1 :  self.IC\_option = 'solitary\_wave'  else :  self.IC\_option = 'dam\_break'  def init\_window(self):  # ==============================#  # PARAMETER INPUT #  # ==============================#  self.IC\_option = 'solitary\_wave'  self.eta\_list = list()  self.time\_list = list()  self.running = False  self.ani = None  self.xleft = StringVar()  self.xright = StringVar()  self.Nx = StringVar()  self.dx = StringVar()  self.tfin = StringVar()  self.A0 = StringVar()  self.xC = StringVar()  self.xwide = StringVar()  self.dt = StringVar()  self.dt\_anim = StringVar()  self.hC = StringVar()  self.h0 = StringVar()  self.hshall = StringVar()  self.hwide = StringVar()  self.dam\_width = StringVar()  self.dam\_height = StringVar()  self.file\_name = StringVar()  self.ani\_name\_mp4 = StringVar()  self.ani\_name\_gif = StringVar()  self.master.title("MoSV 1D")  root.iconbitmap("icon.ico")  vcmd = (self.register(self.validate\_float),'%d', '%i', '%P', '%s', '%S', '%v', '%V',  '%W')  # --------------------------------------------#  # PARAMETER INPUT #  # --------------------------------------------#  tk.Label(self,text="Simulation of SWE 1D",pady=10, padx=50, font  ="Helvetica 20").grid(column=0, row=0,columnspan=2)    border=Frame(root, pady=5 , padx=2)  border.grid(row=0,column=0,padx=15,pady=10,columnspan=2)  self.frame = LabelFrame(border, text = "Xspace and Time", font='Helvetica  10 bold', fg='red', padx=8, pady=15)  self.frame.grid(row=0, column=0, padx=10,pady=2,sticky=W)  self.LX = Label(self.frame, text='Xspace', font='Helvetica 10 bold', fg='red')  self.LX.grid(row=0, column=0,columnspan=2, sticky=W, padx= 10)  self.L0 = Label(self.frame, text='xmin').grid(row=1, column=0, sticky=W,  padx= 26)  self.E0 = tk.Entry(self.frame, textvariable=self.xleft, width=8,  validatecommand = vcmd)  self.E0.grid(row=1, column=1, padx= 10)  self.E0.insert(0,'0')  self.S0 = Label(self.frame, text='m').grid(row=1, column=2,padx=8)  self.L1 = Label(self.frame, text='xmax').grid(row=2, column=0, sticky=W,  padx= 26)  self.E1 = tk.Entry(self.frame, textvariable=self.xright, width=8,  validatecommand = vcmd)  self.E1.grid(row=2, column=1, padx= 10)  self.E1.insert(0,'1.0')  self.S1 = Label(self.frame, text='m').grid(row=2, column=2,padx=8)  self.L2 = Label(self.frame, text='dx').grid(row=3, column=0, sticky=W,  padx= 26)  self.E2 = tk.Entry(self.frame, textvariable=self.dx, width=8, validatecommand  = vcmd)  self.E2.grid(row=3, column=1, padx= 10)  self.E2.insert(0,'0.01')  #self.E3.configure(state='readonly')  self.S2 = Label(self.frame, text='m').grid(row=3, column=2,padx=8)  # --------------------------------------------#  # Time #  # --------------------------------------------#  self.LY = Label(self.frame, text='Time',font='Helvetica 10 bold', fg='red')  self.LY.grid(row=4, column=0, sticky=W, padx= 10,columnspan=2)  self.L3 = Label(self.frame, text='tfin').grid(row=5, column=0, sticky=W,  padx= 26)  self.E3 = Entry(self.frame, textvariable=self.tfin, width=8, validatecommand  = vcmd)  self.E3.grid(row=5, column=1, padx= 15)  self.E3.insert(0,'1.0')  self.S3= Label(self.frame, text='s').grid(row=5, column=2,padx=8)  self.L4 = Label(self.frame, text='dt').grid(row=6, column=0, sticky=W,  padx= 26)  self.E4 = Entry(self.frame, textvariable=self.dt, width=8, validatecommand =  vcmd)  self.E4.grid(row=6, column=1, padx= 15)  self.E4.insert(0,'0.0001')  #self.E12.configure(state='readonly')  self.S4 = Label(self.frame, text='s').grid(row=6, column=2,padx=6)  self.L5 = Label(self.frame, text='dt\_an').grid(row=7, column=0, sticky=W,  padx= 26)  self.E5 = Entry(self.frame, textvariable=self.dt\_anim, width=8,  validatecommand = vcmd)  self.E5.grid(row=7, column=1, padx= 15)  self.E5.insert(0,'0.001')  #self.E13.configure(state='readonly')  self.S5 = Label(self.frame, text='s').grid(row=7, column=2,padx=6)  self.frames = Frame(border, padx=8, pady=15)  self.frames.grid(row=1, column=0, padx=10,pady=10,sticky=W)  # --------------------------------------------#  # Wave Initial condition #  # --------------------------------------------#  self.frame1 = LabelFrame(border, text="Test Case", font='Helvetica 10 bold',  fg='red', padx=8, pady=15)  self.frame1.grid(row=2, column=0, padx=10,pady=2,sticky=W)  self.varIC = IntVar()  self.varIC.set(1)  self.radiobtn1 = Radiobutton(self.frame1, text="Solitary wave",  font='Helvetica 10 bold', fg='red', variable=self.varIC, value=1,  command=self.ICselection)  self.radiobtn1.grid(row=0, column=0, padx=0, pady=5)    #self.L4 = Label(self.frame1, text='Solitary wave', font='Helvetica 10 bold',  fg='red')  #self.L4.grid(row=2, column=0, columnspan=2, sticky=W, padx= 10)  self.L6 = Label(self.frame1, text='xC').grid(row=1, column=0, sticky=W,  padx= 22)  self.E6 = tk.Entry(self.frame1, textvariable=self.xC, width=8,  validatecommand = vcmd)  self.E6.grid(row=1, column=1, padx= 15)  self.E6.insert(0,'0.5')  self.S6 = Label(self.frame1, text='m').grid(row=1, column=2,padx=8)  self.L7 = Label(self.frame1, text='xWide').grid(row=2, column=0, sticky=W,  padx= 22)  self.E7 = Entry(self.frame1, textvariable=self.xwide, width=8,  validatecommand = vcmd)  self.E7.grid(row=2, column=1, padx= 15)  self.E7.insert(0,'0.1')  self.S7 = Label(self.frame1, text='m').grid(row=2, column=2,padx=8)  self.L8 = Label(self.frame1, text='A0').grid(row=3, column=0, sticky=W,  padx= 22)  self.E8 = Entry(self.frame1, textvariable=self.A0, width=8, validatecommand  = vcmd)  self.E8.grid(row=3, column=1, padx= 15)  self.E8.insert(0,'0.02')  self.S8 = Label(self.frame1, text='m').grid(row=3, column=2,padx=8)  # --------------------------------------------#  # Dam Break #  # --------------------------------------------#  self.radiobtn2 = Radiobutton(self.frame1, text="Dam break", font='Helvetica  10 bold', fg='red', variable=self.varIC, value=2,  command=self.ICselection, anchor=tk.W,  justify=tk.LEFT)  self.radiobtn2.grid(row=4, column=0, padx=0, pady=5)  self.L9 = Label(self.frame1, text='Width').grid(row=5, column=0, sticky=W,  padx= 22)  self.E9 = Entry(self.frame1, textvariable=self.dam\_width, width=8,  validatecommand = vcmd)  self.E9.grid(row=5, column=1, padx= 15)  self.E9.insert(0,'0.25')  self.S9= Label(self.frame1, text='m').grid(row=5, column=2,padx=8)  self.L10 = Label(self.frame1, text='Height').grid(row=6, column=0,  sticky=W, padx= 22)  self.E10 = Entry(self.frame1, textvariable=self.dam\_height, width=8,  validatecommand = vcmd)  self.E10.grid(row=6, column=1, padx= 15)  self.E10.insert(0,'0.25')  self.S10 = Label(self.frame1, text='m').grid(row=6, column=2,padx=8)  #=======================================#  # SAVE & LOAD FILE #  #=======================================#  self.frames1 = Frame(border, padx=8, pady=15)  self.frames1.grid(row=4, column=0, padx=10,pady=10,sticky=W)  self.frame5 = LabelFrame(border,text="Save Test Case",font='Helvetica 10  bold', fg='red', padx=4,pady=7)  self.frame5.grid(row=5, column=0,padx=5, columnspan=2, sticky=W)  # self.Lname = Label(self.frame5, text='Test Case').grid(row=0, column=0,  padx= 4)  self.Ename = Entry(self.frame5, textvariable=self.file\_name, width=11)  self.Ename.grid(row=1, column=0, padx= 4)  self.Pname = Label(self.frame5, text='.txt').grid(row=1, column=1)  self.B0 = Button(self.frame5,  text='SAVE',width=5,bg='green',command=self.save)  self.B0.grid(row=1, padx=4,column=2)  self.B1 = Button(self.frame5,  text='LOAD',width=5,bg='yellow',command=self.loadfile)  self.B1.grid(row=1, padx=4,column=3)  #========================= ==#  # SAVE ANIMATION #  #========================== =#  self.frames1 = Frame(border, padx=8, pady=15)  self.frames1.grid(row=6, column=0, padx=10,pady=10,sticky=W)  self.frame6 = LabelFrame(border,text="Save Animation", font='Helvetica 10  bold', fg='red', padx=4,pady=7)  self.frame6.grid(row=7, column=0,padx=5, columnspan=2, sticky=W)  self.Aniname\_gif = Entry(self.frame6, width=12)  self.Aniname\_gif.grid(row=0, column=1, padx= 5)  self.Pname\_gif = Label(self.frame6, text='.gif').grid(row=0, column=2)  self.B0 = Button(self.frame6,  text='SAVE',width=11,bg='green',command=self.save\_animate\_gif)  self.B0.grid(row=0, padx=5,column=3)  #================ =====#  # BUTTON #  #=============== ======#  self.frame7 = Frame(border,padx=8,pady=6)  self.frame7.grid(row=5, column=1, rowspan=2 ,padx=5,pady=2,sticky=W)  self.style = ttk.Style(root)  self.style.layout('text.Horizontal.TProgressbar',  [('Horizontal.Progressbar.trough',  {'children': [('Horizontal.Progressbar.pbar',  {'side': 'left', 'sticky': 'ns'})],  'sticky': 'nswe'}),  ('Horizontal.Progressbar.label', {'sticky': ''})])  self.style.configure('text.Horizontal.TProgressbar', text='MOSV1D')  self.progressBar = ttk.Progressbar(self.frame7, orient="horizontal",  length=590,mode="determinate", style='text.Horizontal.TProgressbar')  self.progressBar.grid(column = 0, row = 0, pady=10, columnspan=2)    self.B5 =  Button(self.frame7,text='RUN',width=32,bg='green',command=self.on\_click)  self.B5.grid(row=1, padx=6,column=0)  self.B1 =  Button(self.frame7,text='CLEAR',width=32,bg='yellow',command=  self.clear\_text)  self.B1.grid(row=1, padx=6,column=1)  #===================== ===== #  # PLOT & ANIMATION #  #======================== ==#  self.frame8 = Frame(border,padx=10,pady=10)  self.frame8.grid(row=0, column=1, rowspan=5, padx=0,pady=0,sticky=W)  self.Axleft=float(self.xleft.get())  self.Axright=float(self.xright.get())  self.Adx=float(self.dx.get())  self.ANx= ma.ceil((self.Axright - self.Axleft)/self.Adx)  self.x = [0]\*(self.ANx+1)  self.x = np.linspace(self.Axleft, self.Axright, self.ANx + 1)    fig, (ax1) = plt.subplots(1, 1, figsize=(6.4, 4.4))  ax1.set\_title("t = 0.00 seconds")  ax1.axis((self.Axleft, self.Axright, -1, 1))  ax1.grid(True)  ax1.grid(color='black', linestyle='-.', linewidth=0.2, alpha=0.2)  ax1.set(xlabel='x (m)', ylabel=r'$\eta$' + ' (m)')    fig.suptitle('Simulation MOmentum conservation Saint-Venant 1 Dimension',  color='r', fontsize=12)  self.canvas = FigureCanvasTkAgg(fig, master=self.frame8)  self.canvas.get\_tk\_widget().grid(column=0,row=0, padx=10)    root.protocol("WM\_DELETE\_WINDOW", on\_closing)  Window(root)  root.mainloop() |