MAST-1D Quick-Start User Manual

Version K16

May 2017

Contents

1 About MAST-1D

MAST-1D is a one-dimensional morphodynamic model that simulates channel and floodplain evolution. It is unique from other models in that it accounts for the grainsize-specific exchange of sediment between the channel and the floodplain via channel migration and/or width change. MAST-1D is computationally efficient, and it is feasible to simulate hundreds of years within several minutes in ideal setups. More details on the theoretical framework behind MAST can be found in Lauer (2016). A complete model description is provided with this User Guide.

MAST-1D version K16 is optimized for Windows machines. The main model code should be able to run on a Mac or Linux operating system; however, to optimize speed, the Python implementation provided will need to be changed. See Section 6 for details.

2 Downloading Python

MAST-1D is written in Python 2.7. It requires the following Python packages:

- csv
- datetime
- deepcopy
- itertools
- json
- math
- matplotlib
- numpy
- os
- pandas
- pickle
- sys
- tkFileDialog
- Tkinter
- xlrd

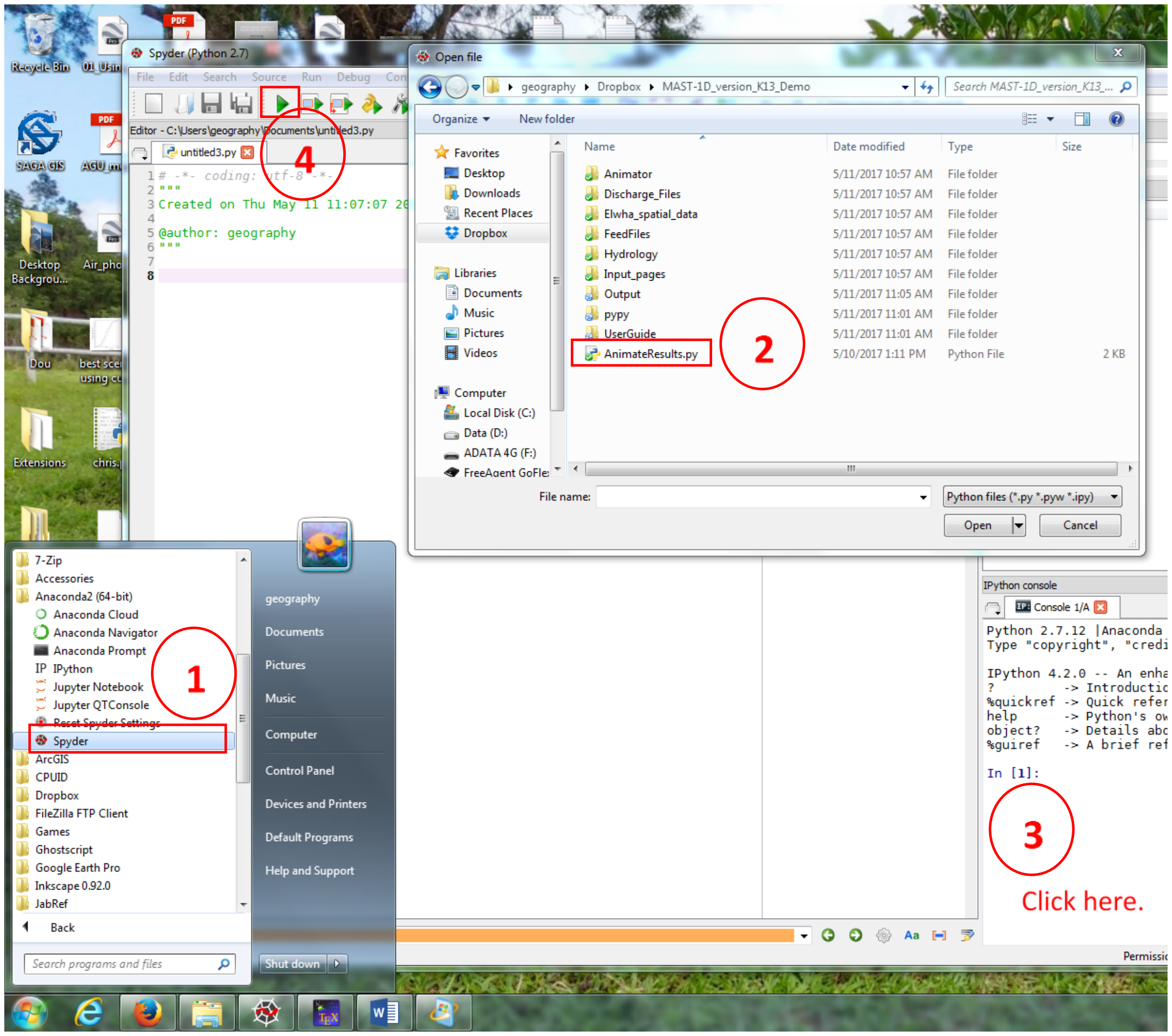
The model folder includes pypy for Windows, a Python implementation that is designed to optimize speed and memory usage. It is run out of an executable file and does not need to be downloaded. However, pypy does not support the packages required for model visualization, and therefore we recommend downloading a second implementation of Python 2.7.

There are a number of Python distributions available. Anaconda (<https://www.continuum.io/downloads>) contains all required packages and is highly recommended. It comes with the IDE Spyder. Good discussions of popular IDEs can be found at: <https://www.slant.co/topics/366/~best-python-ides> and <http://stackoverflow.com/questions/20719180/which-ide-for-scientific-computing-and-plotting-in-python>.

3 Quick Tutorial

3.1 Navigating the animation interface

1. Download Python 2.7 from Anaconda (<https://www.continuum.io/downloads>) or from another source. If not using Anaconda, make sure that the packages listed in Section 2 are downloaded.
2. Start Spyder (located within the Anaconda folder) or your IDE of choice and open **AnimateResults.py**, located within the parent folder. Run the script (Figure 1). Note that you may need to click on the IPython Console in the bottom right corner before pressing the run button if you just started the session.
3. The animation GUI should pop up (it may be hidden behind your IDE). Click **Choose run** and navigate to the MAST folder, then to Output -> Demo -> TrinityDemo (Figure 2).



```
# -*- coding: utf-8 -*-
"""
Created on Thu May 11 11:07:07 20

@author: geography
"""
```

Name	Date modified	Type	Size
Animator	5/11/2017 10:57 AM	File folder	
Discharge_Files	5/11/2017 10:57 AM	File folder	
Elwha_spatial_data	5/11/2017 10:57 AM	File folder	
FeedFiles	5/11/2017 10:57 AM	File folder	
Hydrology	5/11/2017 10:57 AM	File folder	
Input_pages	5/11/2017 10:57 AM	File folder	
Output	5/11/2017 11:05 AM	File folder	
pypy	5/11/2017 11:01 AM	File folder	
UserGuide	5/11/2017 11:01 AM	File folder	
AnimateResults.py	5/10/2017 1:11 PM	Python File	2 KB

```
Python 2.7.12 |Anaconda
Type "copyright", "credi

IPython 4.2.0 -- An enha
?         -> Introductic
%quickref -> Quick refer
help      -> Python's ow
object?   -> Details abc
%guiref   -> A brief ref

In [1]:
```

Figure 1 1. Open Spyder. 2. Open animation GUI. 3. Click in console if just starting session. 4. Run the script.

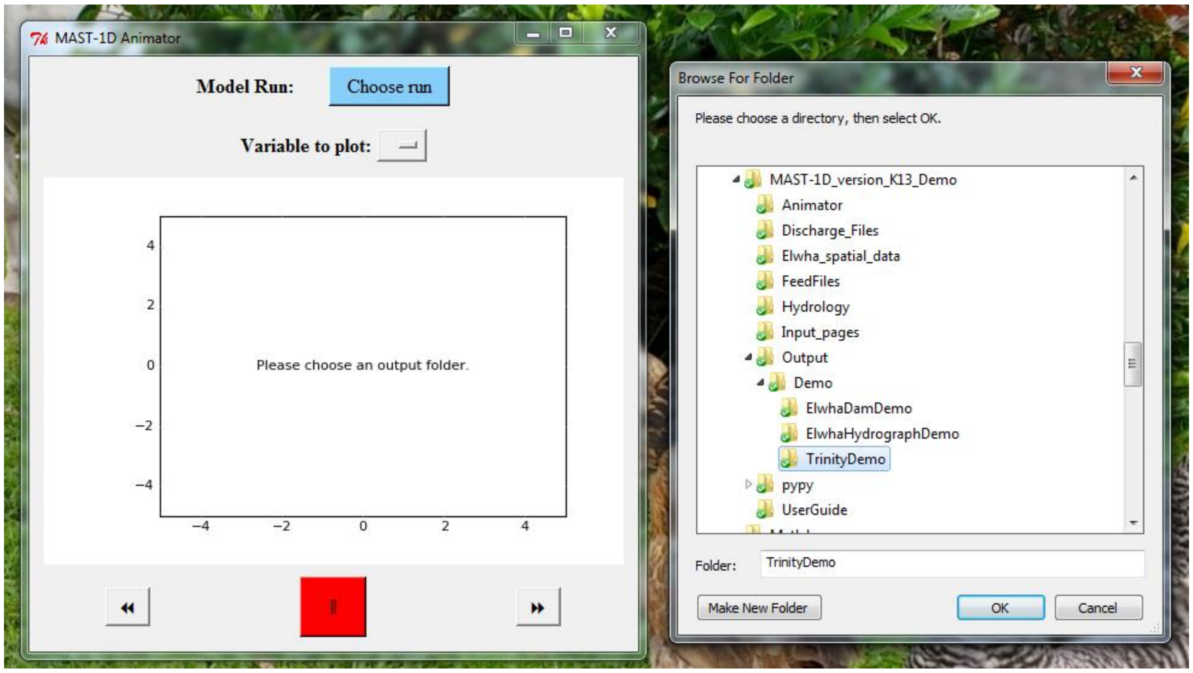


Figure 2 Selecting a run in the animation GUI.

4. The animation should appear for the Trinity River (Table 1). This run loosely simulates sediment starvation due to dam emplacement and subsequent gravel augmentation. Incoming sediment feed is reduced to zero at year 40 and is increased back to capacity at year 240. The time variable *t* shows how much time has elapsed in years.
5. The **Step Forward**, **Pause/Play**, and **Step Backward** buttons on the bottom of the display window allow the user to stop the animation and step through time incrementally (Figure 3). Experiment with these buttons.
6. Many variables are available for plotting, and can be customized by the user (see Section 4, XII). Table 2 lists the variables presented in the demo runs. Under the **Variable to Plot** dropdown menu, explore some of the variables.
7. Explore the other runs in the Demo folder by selecting **Choose run**. Descriptions of the runs are presented in Table 1.

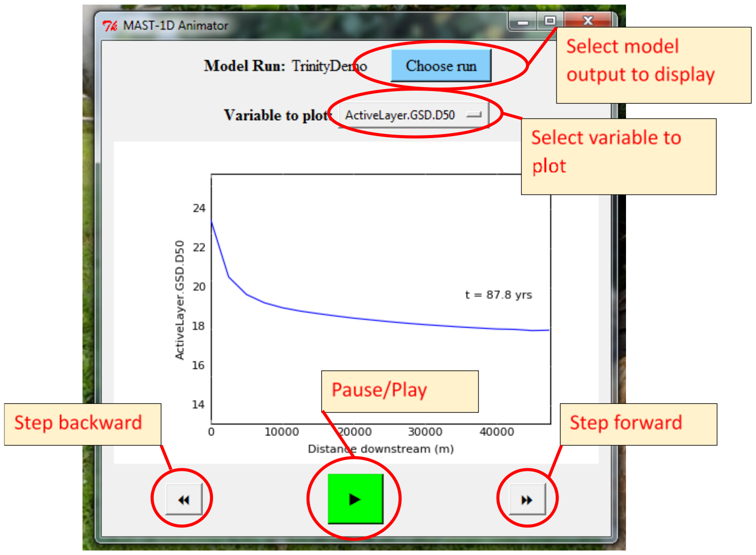


Figure 3 Components of animation GUI

Table 1 Description of demo runs. The 'standard format' for sediment feed and water surface elevations refer to the built-in variables used to set boundary conditions (see Section 4, V).

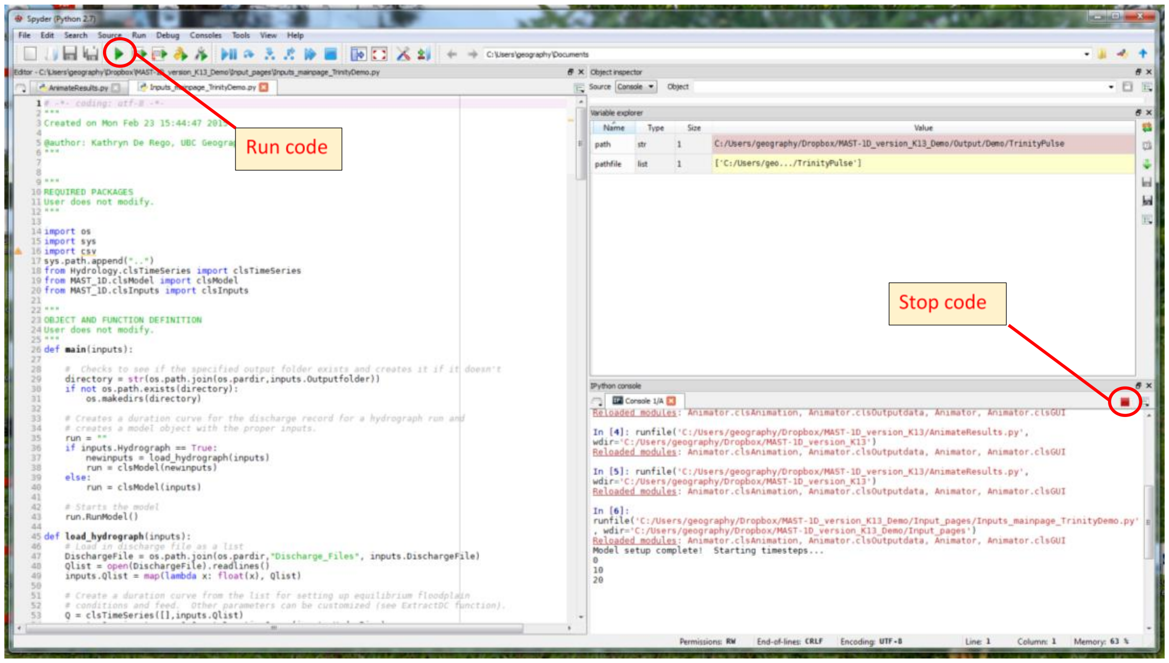
Run	Description
Trinity River	Simulates a reduction in sediment supply caused by dam emplacement and recovery via gravel augmentation along the Trinity River, California. The standard feed format is used to determine upstream sediment feed. Hydraulics are represented by a flow duration curve, and channel width and migration rate are held constant.
Elwha Dam Removal	Simulates dam emplacement and subsequent removal along the Elwha River, Washington. The standard feed format is used to represent the reduction in sediment supply caused by damming. The feed for the pulse of sediment following dam removal is determined using a custom function. Hydraulics are represented by a flow duration curve with a set downstream water surface elevation, which is modified using a custom function. Channel width and migration rate are held constant over time.
Elwha Hydrograph	Simulates Elwha River response to a 5-year hydrograph. Sediment feed is supplied at capacity using the standard feed format. The downstream water surface elevation is calculated for each flow in the hydrograph using the standard format. Separate functions for bank erosion and vegetation encroachment allow width to change over time.

Table 2 Select variables for plotting

Variable	Description
ActiveLayer.GSD.D50	D50 of the active layer (mm)
ActiveLayer.GSD.D84	D84 of the active layer (mm)
Bc	Channel width (m)
Bf	Floodplain width (m)
CumulativeBedChange	Net aggradation/degradation (m)
DC.Hc[-1]	Channel flow depth for largest flow in duration curve (m)
DC.Hc[0]	Channel flow depth for the smallest flow in duration curve (m)
DC.Qw[0]	Total discharge for smallest flow in duration curve (m3/s)
DC.Qwf[-1]	Floodplain discharge for largest flow in duration curve (m3/s)
DC.Sf[0]	Friction slope for smallest flow in duration curve
DC.Uc[-1]	Channel flow velocity for largest flow in duration curve (m/s)
DC.Uc[0]	Channel flow velocity for smallest flow in duration curve (m/s)
DC.WSE[-1]	Water surface elevation for largest flow in duration curve (m)
DC.WSE[0]	Water surface elevation for smallest flow in duration curve (m)
Floodplain.GSD.D50	Floodplain D50 (mm)
Floodplain.GSD.D84	Floodplain D84 (mm)
Floodplain.GSD.F[0]	Fraction of mud in floodplain
Floodplain.L	Floodplain height (m)
Load.QsavBedTot	Total sediment transport rate (m3/s
Load.QsavBedTotFeed	Total sediment feed rate (m3/s)
Slope	Bed slope
Substrate[-1].C.GSD.D50	D50 of the uppermost channel substrate layer (mm)
etabav	Bed elevation (m)

3.2 Running MAST-1D in 'regular' Python and pypy

1. In Spyder (or your IDE of choice), open Input_pages -> Inputs_mainpage_TrinityDemo.py. **Input_mainpage** files are where the model parameters are defined. The code that the user modifies begins on Line 60.
2. Run **Inputs_mainpage_TrinityDemo.py** in the Spyder IDE and note the timestep counters printing in the IPython console on the lower right window (Figure 4). Running the code within Spyder is useful when setting up the run and debugging. Halt the code by pressing the red stop button above the console.


Spyder (Python 2.7)
File Edit Search Source Run Debug Consoles Tools View Help
C:\Users\geography\Documents
Editor - C:\Users\geography\Dropbox\MAST-1D_version_K13_Demo\Input_pages\Inputs_mainpage_TrinityDemo.py
AnimateResults.py
Inputs_mainpage_TrinityDemo.py
Run code
Object inspector
Source Console
Object
Variable explorer
IPython console
Console 1/A
Stop code
Permissions: RW
End-of-lines: CRLF
Encoding: UTF-8
Line: 1
Column: 1
Memory: 63 %

```
# -*- coding: utf-8 -*-
"""
Created on Mon Feb 23 15:44:47 2015

@author: Kathryn De Rego, UBC Geograp
"""


"""
REQUIRED PACKAGES
User does not modify.
"""

import os
import sys
import csv
sys.path.append("..")
from Hydrology.clsTimeSeries import clsTimeSeries
from MAST_1D.clsModel import clsModel
from MAST_1D.clsInputs import clsInputs

"""
OBJECT AND FUNCTION DEFINITION
User does not modify.
"""
def main(inputs):

    #  Checks to see if the specified output folder exists and creates it if it doesn't
    directory = str(os.path.join(os.pardir,inputs.Outputfolder))
    if not os.path.exists(directory):
        os.makedirs(directory)

    # Creates a duration curve for the discharge record for a hydrograph run and
    # creates a model object with the proper inputs.
    run = ""
    if inputs.Hydrograph == True:
        newinputs = load_hydrograph(inputs)
        run = clsModel(newinputs)
    else:
        run = clsModel(inputs)

    # Starts the model
    run.RunModel()

def load_hydrograph(inputs):
    # Load in discharge file as a list
    DischargeFile = os.path.join(os.pardir,"Discharge_Files", inputs.DischargeFile)
    Qlist = open(DischargeFile).readlines()
    inputs.Qlist = map(lambda x: float(x), Qlist)

    # Create a duration curve from the list for setting up equilibrium floodplain
    # conditions and feed.  Other parameters can be customized (see ExtractDC function).
    Q = clsTimeSeries([],inputs.Qlist)
```

Name	Type	Size	Value
path	str	1	C:/Users/geography/Dropbox/MAST-1D_version_K13_Demo/Output/Demo/TrinityPulse
pathfile	list	1	['C:/Users/geo.../TrinityPulse']

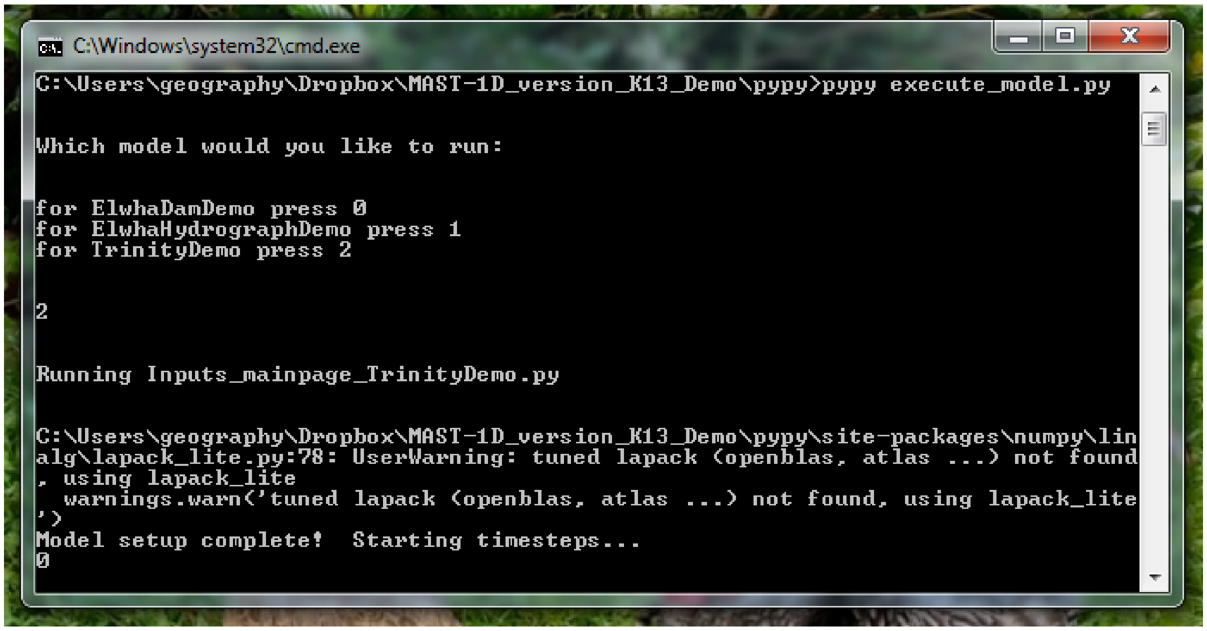
```
Reloaded modules: Animator.clsAnimation, Animator.clsOutputdata, Animator, Animator.clsGUI

In [4]: runfile('C:/Users/geography/Dropbox/MAST-1D_version_K13/AnimateResults.py',
wdir='C:/Users/geography/Dropbox/MAST-1D_version_K13')
Reloaded modules: Animator.clsAnimation, Animator.clsOutputdata, Animator, Animator.clsGUI

In [5]: runfile('C:/Users/geography/Dropbox/MAST-1D_version_K13/AnimateResults.py',
wdir='C:/Users/geography/Dropbox/MAST-1D_version_K13')
Reloaded modules: Animator.clsAnimation, Animator.clsOutputdata, Animator, Animator.clsGUI

In [6]:
runfile('C:/Users/geography/Dropbox/MAST-1D_version_K13_Demo/Input_pages/Inputs_mainpage_TrinityDemo.py'
, wdir='C:/Users/geography/Dropbox/MAST-1D_version_K13_Demo/Input_pages')
Reloaded modules: Animator.clsAnimation, Animator.clsOutputdata, Animator, Animator.clsGUI
Model setup complete!  Starting timesteps...
0
10
20
```

Figure 4 Running and stopping Inputs_mainpage_TrinityDemo.py

3. MAST-1D can also be run using pypy, a much faster version of Python. In the MAST-1D folder, double click on **RunModel.bat** (make sure that you have halted the code in the IPython console). A command prompt window will appear, prompting the user to select one of the runs within the Inputs folder (Figure 5). Press **2** then **Enter** for **TrinityDemo** and watch the run begin.
4. While **TrinityDemo** is still running in the command prompt window, run **AnimateResults.py** in Spyder and select **TrinityDemo** as the run. Note that the animation will plot the available output, even if the model run is not complete. To update the animator as the run progresses, close and rerun it.
5. We will now modify the Trinity River run to simulate a pulse of sediment instead of supply reduction. In the **TrinityDemo.py** file, change the second entry in the **inputs.LoadFactor** list from 0 to 2. The values in the **inputs.LoadFactor** list represent the upstream sediment feed as a multiple of the sediment transport capacity. The **inputs.LoadFactorCount** list indicates when each feed change occurs. In our new run, we will begin by feeding sediment into the model at capacity (**inputs.LoadFactor** = 1.). At timestep 200, the feed will be increased to twice the capacity (**inputs.LoadFactor** = 2.). It will be reduced back to capacity at timestep 1200. On lines

118 and 119, change **inputs.dt** and **inputs.dtcount** to [.2, .1] and [200], respectively. This is instructing the model to reduce the timestep when the sediment pulse hits.

6. On line 313, change **RunName** from **"TrinityDemo"** to **"TrinityPulse."** Save the file as **Inputs_mainpage_TrinityPulse.py** (File -> SaveAs) and run pypy through **RunModel.bat**. Your new TrinityPulse run should appear as an option. Run the model and view it in the animator.



```
C:\Users\geography\Dropbox\MAST-1D_version_K13_Demo\pypy>pypy execute_model.py


Which model would you like to run:


for ElwhaDamDemo press 0
for ElwhaHydrographDemo press 1
for TrinityDemo press 2


2


Running Inputs_mainpage_TrinityDemo.py


C:\Users\geography\Dropbox\MAST-1D_version_K13_Demo\pypy\site-packages\numpy\lin
alg\lapack_lite.py:78: UserWarning: tuned lapack (openblas, atlas ...) not found
, using lapack_lite
  warnings.warn('tuned lapack (openblas, atlas ...) not found, using lapack_lite
')
Model setup complete!  Starting timesteps...
0
```

Figure 5 Command prompt window running pypy

4 Adding inputs

Model parameters are set in the **Inputs_Mainpage_** files located in the **Inputs** folder. The user may save an unlimited amount of input files, but they all must be saved directly in the **Inputs** folder with the **Inputs_Mainpage_** prefix.

Parameters are organized into 12 categories. Unless otherwise stated, all variables must be set. A general guide to filling in each section is presented here. The heading numbers correspond to those in the code—refer to demo **Inputs_Mainpage_** files for specific details on each variable.

I: Import spatial data

The user has 3 options for setting the initial conditions of the run: 1) beginning from a homogenous state where all nodes are identical, 2) importing a file with select attributes for each node, or 3) importing the final node conditions from a completed MAST-1D run.

FOR HOMOGENOUS NODES: Comment out section I.A and set **inputs.initialcond** to **False** in I.B. **Inputs.priorReach** can be left alone—it will not be called in the code.

TO IMPORT NODE ATTRIBUTES: Adapt section I.A to your needs. Variables can be any scalar attribute of the Node class (see MAST_1D -> clsNode) and should be supplied as a list of values from upstream to downstream. A 'representative' value for each variable must also be supplied where noted elsewhere in order to calculate boundary conditions.

TO IMPORT RESULTS FROM A PREVIOUS RUN: Set **inputs.initialcond** to **True** in I.B and set **inputs.priorReach** to the full pathname of the **save.Reach** file, which can be found within the output folder (Output/YourCompletedRun). **Important note:** Importing a prior run sets parameters for the Reach object (Table 3) but not the boundary conditions. Sections II-XI must still be filled in in order to calculate the sediment feed at the upstream boundary.

II: Set channel geometry

All variables in this section must be set, even if importing custom data or a prior Reach object, as they are used to calculate boundary conditions.

III: Set flow type

Select **True** to run a hydrograph and **False** for a flow duration curve.

IV: Set timestep parameters

MAST-1D tracks time differently for hydrographs and duration curves. If using a flow duration curve, set the number and length of timesteps in IV.A. The timestep can be set to change during the run; it will initially be set to the first value of **inputs.dt**, and go down the list at the timesteps specified in **inputs.dtcount**.

If running a hydrograph, put one (arbitrary) value into **inputs.dt** in IV.A and leave **inputs.dtcount** as an empty list. Then set the values in IV.B.

V: Set boundary conditions

The upstream sediment feed parameter, **inputs.LoadFactor**, is a multiple of the sediment transport capacity given the parameters provided in sections II, V-VIII, and X. The feed can be changed throughout the run by adding values to **inputs.LoadFactor** and the timestep/date of change to

inputs.LoadFactorCount. The downstream water surface elevation (WSE) can also be set to change over time if desired. If **inputs.SetBoundary** is set to False, MAST-1D will calculate the WSE based on the flow and will ignore WSE parameters provided in the input.

VI: Set hydraulic roughness parameters

The user can decide to use either the Manning or Chezy equation when calculating hydraulics. Note that the Chezy equation has not been tested in this version of MAST-1D.

VII: Set discharge

If using a flow duration curve, set the flows and frequencies in VII.A. If using a hydrograph, provide the name of a file holding the discharge data (including the file extension). It should be stored in the **Discharge_Files** folder. Flows should be separated by a newline character. Make sure that there is not a newline after the final discharge value.

VIII: Set grainsize and sediment transport parameters

All variables in this section must be set, even if importing custom data or a prior Reach object, as they are used to calculate boundary conditions.

IX: Set migration/width change parameters

There are three ways to characterize channel migration in MAST-1D: 1) banks are fixed; 2) the channel migrates at a constant rate and channel width remains constant; and 3) bank erosion and channel narrowing occur at different rates, leading to changes in channel width.

TO ASSUME FIXED BANKS: Set **inputs.WidthChange** to **False** and **inputs.migration** to 0. The other parameters can be ignored.

TO ASSUME A CONSTANT MIGRATION RATE: Set **inputs.WidthChange** to **False**. The **inputs.migration** variable represents the annual migration rate. The other parameters can be ignored.

TO ALLOW WIDTH CHANGE: Set **inputs.WidthChange** to True. The **inputs.migration** variable can be ignored. All other parameters must be set.

X: Set reservoir parameters

All parameters must be set except for **inputs.AlphaPartlyAlluvial**. MAST-1D contains a function that limits bed degradation by allowing nodes to become 'partly alluvial.' It is turned off by default, but the user can write custom code to activate it (Refer to II VI.C. in MAST-1D -> clsModel).

XI: Set tracers

The production and decay rates of a user-defined number of tracers can be set. The tracer functions have not been tested in this version of MAST-1D.

XII: Set output specifications:

There are three types of output files:

1) STANDARD OUTPUT: These files are written both for flow duration curve and hydrograph runs and have a prefix of 'Out_.' They contain values for each node at equal intervals for a user-defined number of time intervals. The user can specify which variables to write to file. Any attribute in the Node class or its child classes can be specified (see MAST-1D -> clsNode).

2) VALIDATION OUTPUT: These files work the same way as 1) except that the user defines which dates to write data instead of specifying a number of printouts. If using this output, be sure to specify the correct start date in III. These files have a prefix of 'Out_Validate.'

3) **DAILY OUTPUT:** When in hydrograph mode, MAST-1D stores daily values of channel width, cumulative channel narrowing, cumulative channel widening, active layer GSD, floodplain GSD, GSD of the uppermost substrate layer, discharge, total bed material load, total feed, and fractional sediment load for user-defined nodes. They are written as json files with the prefix 'save.Daily.' Note that these files can be several megabites.

5 Customizing MAST-1D

MAST-1D is written in an object-oriented framework, making it relatively simple to customize it to users' specific needs. The model is stored in the **MAST_1D** folder. General descriptions of the attributes and methods in the most important classes are presented in Table 3 and a UML object diagram is provided in Figure 6. To get familiar with the code, start by going through **clsModel**, which calls in the functions that set up and run MAST-1D. Also refer to the **StepDownstream** function in **clsReach**, which contains the series of steps iterated through during each timestep.

5.1 Example customization: simulating dam removal sediment feed

Sediment feed following an exponential decay curve was desired to simulate a pulse of sediment following dam removal. This would not have been feasible with the default MAST-1D variables found in section V of **Inputs_mainpage_**. Therefore, a custom function was created to calculate feed at each timestep, and lines of code were written to call the function at the proper time. Code was written in four locations:

1. The exponential decay function (called **UpdateElwhaDamRemovalDurationCurve**) is located in **clsLoad**.
2. At the end section VI.D (Activate dam removal) in **clsModel**, a piece of code is written that triggers the **FeedType** variable to change to **'DamRemovalDuration'** after 5100 timesteps have elapsed.
3. An 'if' statement is added to section VI.E (Update sediment feed) of **clsModel**. It uses the **FeedType** variable to call **UpdateElwhaDamRemovalDurationCurve**.
4. A Boolean variable called **Removal** is defined in **Inputs_Mainpage_**. It triggers the dam removal code in section VI.D of **clsModel**.

Table 3 Select objects in MAST-1D

Object class	**Description**
clsModel	Each run is composed of a clsModel instance whose functions run the model. This class sets up the model parameters given inputs provided in Inputs_mainpage, keeps track of time in the model, manages boundary conditions, and holds the code that determines when to write output.
clsReach	Each run contains one clsReach object named Reach, which holds all nodes and modulates interactions between them. This class contains the important 'StepDownstream' function, which calls the functions perfomed on the nodes at each timestep (calculate sediment transport, calculate channel/floodplain exchange, etc). The functions to set up the initial conditions and calculate hydraulics via the backwater equation are also housed in clsReach.
clsNode	The Reach object contains multiple instances of clsNode, in which all of the information for each node (width, elevation, grainsize distributions, etc.) are stored. Node objects contain active layer, floodplain, and substrate reservoirs, as well as a sediment load. The functions in this class modulate all exchange between reservoirs within each node (e.g. width changes, sediment input/output due to channel migration, and aggradation/degradation).
clsLoad	Each Node object has a clsLoad instance within it. Load objects hold the sediment load for the Node, as well as functions to calculate sediment transport and upstream sediment feed.

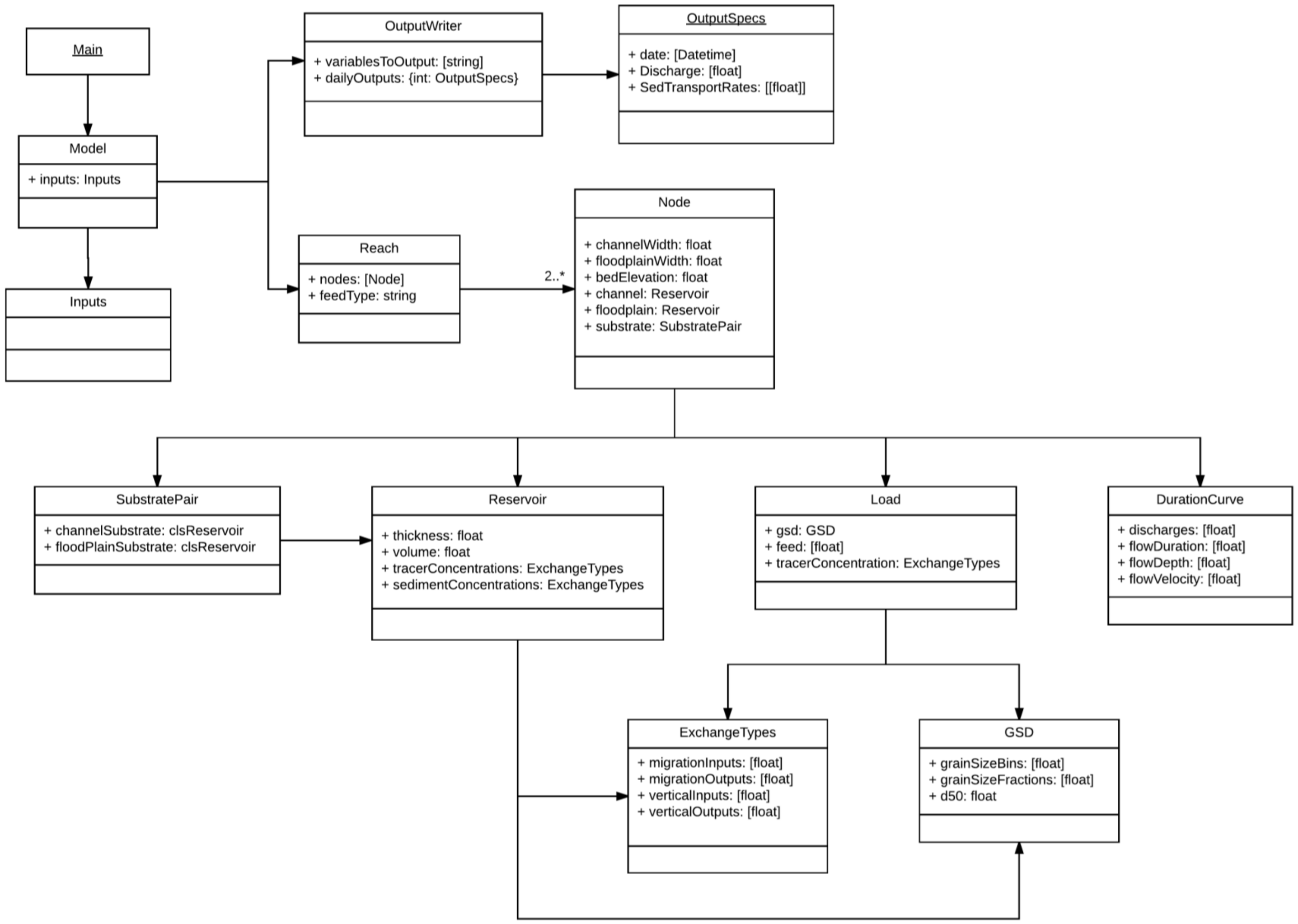


Figure 6 UML diagram for MAST-1D. Only select attributes are listed.

6 Setting up MAST-1D for Macs and Linux

The raw MAST-1D code should be compatible with Mac and Linux operating systems. However, the implementation of pypy (in the **pypy** folder) is Windows-specific. Download the necessary version of PyPy2.7 at < https://pypy.org/download.html >. Follow the instructions on the webpage to install Numpy for pypy.