This manual explains how to set up the model, the needed input files, implemented functions, and examples of outputs from the model. The descriptions of the sediment transport model (STM) in this appendix are based on the current setting introduced in Chapter 5. The model (STM) contains five python files to execute the STM and a folder (e.g., a folder named “Erosion”) that contains the stand-alone WEPP codes written in FORTRAN (Table A1).

Table A1. List of the scripts for the STM

|  |  |
| --- | --- |
| File/folder | Description |
| Erosion | Stand-alone WEPP code written in FORTRAN |
| bc.py | Establishing boundary condition of the model domain |
| deposition.py | Calculating settling velocity, critical shear stress, and deposition rate |
| main.py | The main script to set up the model, import input data, call the implemented functions, and execute the model |
| solver.py | Numerically solving the 2D advection-dispersion equation |
| source.py | Executing stand-alone WEPP code to provide sediment source |

## Variable’s configuration

The variable can be either 1D, 2D, or 3D array in the model. The 3D array configuration is [*nt*, *nx*, *ny*], which means that the variable varies in time and space. Here, *nt* is the number of time steps used in the model, *nx* and *ny* are the number of grid points in *x* and *y* directions. The 2D array configuration is generally [*nx*,*ny*], which means that the variable is constant over time but varies in space. If 1D array configuration is [*nt*], then the variable varies in time but is constant in space.

## Model set-up

In this section, the needed variables for the model set-up are shown with the example codes. All the codes shown in this section can be found in the “main.py” script. First, the path that contains the list of the python scripts in Table A1 should be declared as follows:

path **=**r"F:\04\_sediment\_model\_ch3\WRE watersheds\WRE6\sediment\_transport\_model"

The model that simulates the hydrology should be provided. For instance, in Chapter 5, since the MIKE SHE was used to provide the needed hydrologic variables, the MIKE IO library was installed (<https://dhi.github.io/mikeio/>) and the path that contained the simulation results from MIKE SHE was defined as “MIKE\_SHE” as follows. The appropriate adjustments should be made if another hydrologic model is used.

**from** mikeio **import** Dfs2 # to read MIKE SHE outputs

MIKE\_SHE **=** r"F:\04\_sediment\_model\_ch3\WRE watersheds\WRE6\MIKE\_SHE\FR6.she - Result Files"

Next, the simulation period for the STM should be declared by assigning the year, month, day, and hour for the variables named “start” and “end” as the code below. The example below is for the first period of the sediment simulations in Chapter 5. Note that the start date is 4/8/1993 00:00. The current setting is on a daily basis. If the desired time step is hourly, then simply type ‘hours’ instead of ‘days’. Note that the changes to hourly simulation may need further modifications such as unit conversion in the STM.

# time domain (for sediment model)

start **=** datetime**(**1993**,**4**,**8**,**0**)**

end **=** datetime**(**1993**,**7**,**20**,**0**)**

time\_step **=** 'days' # days or hours

The spatial model domain and grid size should be provided. The example below is for one of the WRE watersheds in Chapter 5. Here, the watershed dimension is 80m wide x 200m long, and the watershed is divided into 20m grid cells.

# Spatial domain

x **=** 80 # entire length in meter

y **=** 200 # entire length in meter

dx **=** dy **=** 20 # grid size in meter

In addition, slope values should be defined as below. The example below is the uniform slope value across the model domain, with the slope of 2.9%. If spatially varied slope values are assigned, the following codes should be modified accordingly.

# Slope

S = np.zeros((nx,ny)) # slope

S[:,:] = 2.9 / 100 # slope

The start date of the hydrologic model (i.e., MIKE SHE) excluding the warm-up period should be provided in case the start dates of sediment and hydrologic simulations are different. In Chapter 5, the MIKE SHE simulation starts 1/1/1993.

sim\_begin **=** datetime**(**1993**,**1**,**1**)**

## Input files

All of the needed input files were imported in the main script in text files and should be located in the same directory as in the script files (Table A2). The 1D array, time-varying variables [*nt*], contain the same period as the MIKE SHE simulation period (i.e. daily values from 1993 to 1995), and the 2D array ([*nx*,*ny*]) should have the same dimensions as the model domain.

Table A2. Needed text files as input data

|  |  |  |
| --- | --- | --- |
| File name | Configuration | Description and the unit used in text files |
| SAND.txt | 2D [*nx,ny*] | Fraction of sand for topsoil [%] |
| SILT.txt | 2D [*nx,ny*] | Fraction of silt for topsoil [%] |
| CLAY.txt | 2D [*nx,ny*] | Fraction of clay for topsoil [%] |
| orgmat.txt | 2D [*nx,ny*] | Fraction of organic matter for topsoil [%] |
| RCF.txt | 1D [*nt*] | Reside cover fraction [0-1] |
| canhgt.txt | 1D [*nt*] | Canopy height of plant [m] |
| RRC.txt | 1D [*nt*] | Soil random roughness [mm] |
| ki\_factor.txt | 1D [*nt*] | Adjusted factors for interrill erodibility [-] |
| kr\_factor.txt | 1D [*nt*] | Adjusted factors for rill erodibility [-] |
| crit\_factor.txt | 1D [*nt*] | Adjusted factors for critical shear stress [-] |

The text files are loaded and saved as the variables in the “main.py” as follows.

# Particle fraction for topsoil

sand **=** np**.**round**(**np**.**loadtxt**(**"SAND.txt"**)** **/** 100 **,**3**)**

silt **=** np**.**round**(**np**.**loadtxt**(**"SILT.txt"**)** **/** 100 **,**3**)**

clay **=** np**.**round**(**np**.**loadtxt**(**"CLAY.txt"**)** **/** 100 **,**3**)**

#orgmat = 1.724\*orgc # if only organic carbon content is available (from WEPP document)

orgmat **=** np**.**round**(**np**.**loadtxt**(**"orgmat.txt"**)** **/** 100**,**3**)**

vfs **=** 0.3 # fraction of very fine sand in the surface soil

# Time varying variables

RCF **=** np**.**loadtxt**(**"RCF.txt"**)** # residue cover fraction (0-1)

canhgt **=** np**.**loadtxt**(**"canhgt.txt"**)** # canopy height of plant (m)

RRC = np.loadtxt("RRC.txt") / 1000 # soil random roughness (m)

Note that the three adjusted factors listed in Table A2 were used as the constant calibration parameters for each period in Chapter 5, and thus the current setting deactivated the loading text files for these variables as follows. If the time-varying values for the adjustments are provided, the following codes should be modified accordingly.

# KIADJ = np.loadtxt('ki\_factor.txt') # adjusted factors for interrill erodibility

# KRADJ = np.loadtxt('kr\_factor.txt') # adjusted factors for rill erodibility

# SHCRTADJ = np.loadtxt('crit\_factor.txt') # adjusted factors for critical shear stress

KIADJ **=** 0.2 **\*** np**.**ones**(**len**(**RCF**))**

KRADJ **=** 0.2 **\*** np**.**ones**(**len**(**RCF**))**

SHCRTADJ **=** np**.**ones**(**len**(**RCF**))**

In addition, the concept of detention storage was used for the deposition term and thus the value can be specified in the following code.

detention = 0.0002 # detention storage for deposition term [m]

## Hydrologic variables

Hydrologic variables, which are daily changes in depth of overland flow (H), velocity fields in *x* and *y* directions (*u* and *v*) for the STM, and sub-daily characteristics of precipitation and runoff for stand-alone WEPP codes are required to run STM. In chapter 5, the MIKE SHE model was used to provide the needed hydrologic variables for the STM. These values were loaded as the following codes. Note that in the code below, the value of 1096 was used to remove the days for the warm-up period in the MIKE SHE model (1990-1992), and the code “1**:-**1**,** 2**:-**1” was used to remove the boundary values specified in MIKE SHE model. Depending on the use of warm-up period and the number of boundary layers, the code below should be adjusted accordingly.

dfs\_Q **=** Dfs2**(**MIKE\_SHE **+** "\FR6\_overland.dfs2"**)**

ds **=** dfs\_Q**.**read**()**

H **=** ds**.**data**[**0**]** # depth of overland water <Water Depth> (meter)

u **=** ds**.**data**[**1**]** # overland flow in x-direction (m/sec)

v **=** ds**.**data**[**2**]** # overland flow in y-direction (m/sec)

# remove warm-up years (first 3 years (1990-1992))+shift and boundary data

H **=** H**[**1096**+**shift**:,**1**:-**1**,** 2**:-**1**]**

u **=** u**[**1096**+**shift**:,**1**:-**1**,** 2**:-**1**]**

v **=** v**[**1096**+**shift**:,**1**:-**1**,** 2**:-**1**]**

In addition to the MIKE SHE outputs, sub-daily characteristics of precipitation should be provided as text files (Table A3). Note that in Chapter 5, sub-daily precipitation was estimated using the CLIGEN model, and the outputs were loaded as the text file. The time domain was the same as the hydrologic simulations (1993-1995). For the sub-daily runoff characteristics, it was assumed that the effective runoff duration was equal to the effective rainfall duration. Also, the peak runoff rate was estimated based on the modified rational method, with the characteristics of the study sites. If the sub-daily properties for precipitation and runoff are obtainable, the code should be changed accordingly.

Table A3. Needed text files for sub-daily precipitation

|  |  |  |
| --- | --- | --- |
| File name | Configuration | Description and the unit used in text files |
| rain\_intensity.txt | 1D [*nt*] | Effective rainfall intensity [m/s] |
| rain\_duration.txt | 1D [*nt*] | Effective rainfall duration [s] |

## Implemented functions

Most of the functions in four python files, excluding the “main.py” files, listed in Table A1 are called and executed in the “main.py”. In the “bc.py”, the boundary conditions must be defined accordingly, and 2D advection-dispersion equations are numerically solved in “solver.py” with the implemented boundary conditions. In addition, source and sink terms are calculated in “source.py” and “deposition.py”, respectively (Table A1). This section describes the python files listed in Table A1, except for the main file. The boundary condition is currently set as the zero-gradient boundary condition except for the bottom of the watershed. The bottom of the watershed is currently set as zero concentration. To change the boundary conditions, the scripts should be modified in the “bc.py” file. In the deposition.py file, three variables are calculated. First, the settling velocity and critical shear stress (defined as “settling(Ds)” and “crit\_shear\_stress(Ds)”) should be called before the function for the deposition rate (defined as “deposit(Vs, c, H, tau\_b, tau\_cr, threshold)”) is called in the “main.py”. These are functions of particle diameter (Ds) to estimate the settling velocity and critical shear stress for each sediment class. Lastly, the deposition rate, which is a function of the settling velocity (Vs), sediment concentration (c), water depth (H), bed and critical shear stress (tau\_b and tau\_cr), and detention storage (threshold), can be calculated. Once the function for deposition rate is executed, the distributed deposition rate across the model domain (dep) is provided as the sink term.

In the solver.py file, there are five functions to solve the advection-dispersion equation (Table A4). The STM numerically solves 2D advection-diffusion equation. Excluding source and sink terms, the equation can be written as,

To numerically solve this problem, an explicit finite difference scheme with central differencing for diffusion term, the Lax-Wendroff scheme (Lax and Wendroff, 1960) for advection term, is used as follows.

Transposing all the terms to the right side of the equation except the concentration at the next time level () yields,

To check the stability conditions, the maximum courant number for each grid is calculated at each time step in the “main.py” (variable named as “check”), and if the “check” is greater than stability criteria (currently set as 0.25, as in the code below), the default time step is repeatedly reduced by half until the stability criterion is satisfied.

# Stability criteria (Courant number)

stability\_criteria **=** 1**/**4

In addition, instead of solving x and y directions simultaneously, x and y advections were solved separately. This scheme is called “directional splitting” and it solves for one-directional flow first and then copies the solutions and then solves for the other one later. The Von Neumann stability analysis shows using directional splitting makes the numerical scheme more stable and less restrictive than solving both directions simultaneously (Durran, 1999).

Table A4. Functions implemented in solver.py

|  |  |
| --- | --- |
| Function | Description |
| D\_C(tau\_b, H,S) | Calculating the diffusion coefficient  Variables in parentheses: (bed shear stress, water depth, slope) |
| diffusion(b,D,dx,dy,dt) | Numerically solving 2D diffusion equation  Variables in parentheses: (concentration distribution, diffusion coefficient, grid size in x, grid size in y, time step) |
| LW(s1d, u1d, dx, nx, dt) | Lax-Wendroff method to solve 1D advection equation, this function is called inside of advection\_x and advection\_y functions |
| advection\_x(a, b, c, u, nx, dx, dt) | Numerically solving advection in x direction  Variables in parentheses: (concentration distribution, cumulative time step for concentration, cumulative time step for velocity fields, velocity in x-direction, grid size in x, time step) |
| advection\_y(a, b, c, u, ny, dy, dt) | Numerically solving advection in y-direction  Variables in parentheses: (concentration distribution, cumulative time step for concentration, cumulative time step for velocity fields, velocity in y-direction, grid size in y, time step) |

In the source.py file, the script is written to run the stand-alone WEPP code and returns the sediment source for five sediment classes across the model domain. The needed input variables are explained in Table A5. Note that the length and width of the hillslope are the same as the grid dimension. The detailed calculations can be referred to the code.

Table A5. The function implemented in source.py

|  |  |
| --- | --- |
| Function | Description |
| WEPP\_HE(length,Q, Q\_P, Q\_T, P\_I, P\_T, slope, vfs, sand, silt, clay, orgmat, RCF, canhgt, KIADJ, KRADJ, SHCRTADJ, RRC, FRCTRL) | Variables in parentheses:  (hillslope length: set as grid size in x [m], water depth [m], peak runoff rate [m/s], effective runoff duration [s], effective rainfall intensity [m/s], effective rainfall duration [s], slope [m/m], fraction of very fine sand [-], sand fraction [-], silt fraction [-], clay fraction [-], organic matter [-], residue cover fraction [-], canopy height of plant [m], adjustment for interrill erodibility, adjustment for rill erodibility, adjustment for critical shear stress, soil random roughness [m], total rill friction factor [-] |

## Comparisons with sediment observations

The scripts written in “main.py” include calculating Nash-Sutcliffe model (NSE) and percentage bias (PBIAS), and the values are printed and automatically inserted in some of the figures shown in Chapter 5. To compare the simulation results with sediment observations, a text file (‘Sed\_observed.txt’) should be inserted in the same directory of the scripts. The observation contains daily observations between 1993 and 1995. The following shows the implemented code to load the observed sediment data.

sed\_observed **=** np**.**loadtxt**(**'Sed\_observed.txt'**)** # 1993 to 1995

## Execution of the model

Once the model is set up and the needed input files are provided, the model can be executed. The model prints the progress in estimating sediment production based on the stand-alone WEPP code as follows.

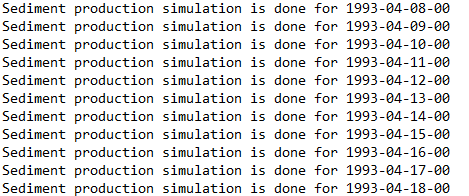


Figure A1. Progress in simulating sediment production reported in STM

Once the simulation of sediment production is finished, the sediment transport module is simulated. Note that if the time step is reduced to meet the specified stability criteria, the relevant information is also printed as follows.

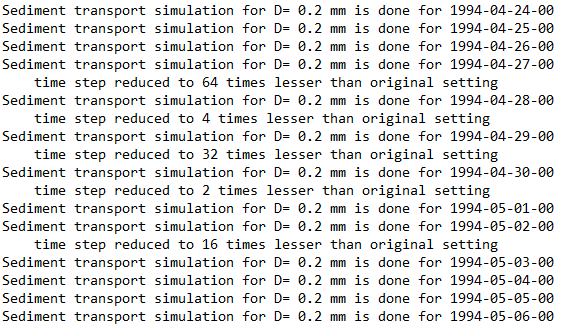


Figure A2. Progress in sediment transport module reported in STM

Once the simulation is completed, some of the results are printed as follows. In addition to NSE and PBIAS, sediment delivery ratio and the fractions of each sediment class to total sediment source and yield are provided (Figure A3).

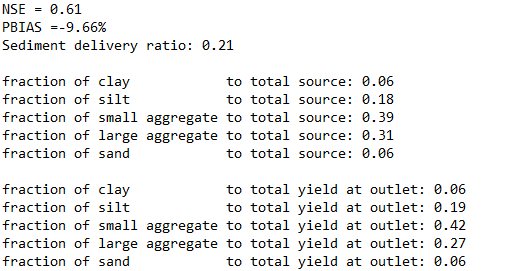


Figure A3. Simulation results reported in STM