Sediment transport and Vegetation Operators

Mariela Perignon

July 24, 2014

This document describes the implementation of operators for sediment transport and vegetation drag in ANUGA, as well as some utilities that are useful for these operators but are written generically. The sediment transport operator can currently only be used with a single processor. Making it parallel safe would require modifying, at minimum, the method used for sediment flux. The vegetation drag operator should be parallel safe but has not been tested.

These operators use the equations presented in *Simpson and Castelltort* [2006] and *Davy and Lague* [2009] for calculating sediment transport and momentum sinks, and *Nepf* [1999] and *Kean and Smith* [2006] for vegetation drag.

1 Files

All files related to these operators are located in the directory `operators/sed_transport/`.

The operators and most of their shared functions are found in `sed_transport_operator.py`. The specific equations used for calculating entrainment, deposition, topographic change, vegetation drag, and momentum change are housed in this file. The operator functions that directly modify quantities are found in `sed_transport_mesh.py`. The file `sed_transport_config.py` lists default values and equation constants shared with the other files that can be modified by the user.

Utilities for transforming ASCII rasters of quantities other than elevation into point files are found in `operators/sed_transport/file_conversion/`. The file `sed_transport_utils.py` includes functions for setting the value of spatially-varied quantities from point files using nearest-Neighbour interpolation, implementations of Dirichlet and Reflective boundaries that accept boundary values of sediment concentration, and a modified version of `create_domain_from_regions` that allows for the creation of new quantities.

2 Architecture

2.1 Creating the domain

Both of these operators require that process-specific quantities be defined when the domain is created: *concentration* for sediment transport calculations (as an evolved quantity), *vegetation* for vegetation drag, and *diffusivity* if turbulence is used. Exceptions are raised if these quantities do not exist when the operators are created.

The existing function for creating a rectangular domain already accepts lists of quantities as an argument. For a rectangular domain where sediment transport, vegetation drag, and turbulence will be calculated, define `evolved_quantities` and `other_quantities` with the necessary process-specific quantities and create the domain as usual:

```
evolved_quantities = ['stage', 'xmomentum', 'ymomentum', 'concentration']

other_quantities = ['elevation', 'friction', 'height', 'xvelocity', 'yvelocity', \
                    'x', 'y', 'vegetation', 'diffusivity']

points, vertices, boundary = rectangular_cross(int(length/dx),
                                               int(width/dy),
                                               len1=length,
                                               len2=width)

domain = Domain(points,
                vertices,
                boundary,
                evolved_quantities = evolved_quantities,
                other_quantities = other_quantities)
```
Listing 1: Creating a rectangular domain

Creating a domain that uses a mesh as input for the quantity *elevation* is usually done with the function `create_domain_from_regions`. An equivalent function was created for these operators where the quantities can be defined as a function argument (see 3 for a description):

```
from anuga.operators.sed_transport.sed_transport_utils \
    import create_domain_from_regions_sed

evolved_quantities = ['stage', 'xmomentum', 'ymomentum', 'concentration']

other_quantities = ['elevation', 'friction', 'height', 'xvelocity', 'yvelocity', \
                    'x', 'y', 'vegetation', 'diffusivity']

domain = create_domain_from_regions_sed(project.bounding_polygon,
                                        boundary_tags = {'bottom': [0],
                                                         'side1': [1],
                                                         'side2': [2],
                                                         'top': [3],
                                                         'side3': [4],
                                                         'side4': [5] },
                                        maximum_triangle_area = project.default_res,
                                        mesh_filename = project.meshname,
                                        interior_regions = {},
                                        evolved_quantities = evolved_quantities,
                                        other_quantities = other_quantities,
                                        verbose = project.verbose)
```
Listing 2: Creating a domain from a mesh file

2.1.1 Process-specific quantities

concentration

Following *Simpson and Castelltort* [2006], *concentration* is given by:

$$\text{Concentration } = \frac{Q_s}{Q} h = Ch \tag{1}$$

where Q_s is the sediment discharge, Q is the total discharge, h is the flow depth, and C is the fractional concentration of sediment (by volume) in the flow. The operators store Ch as the quantity *concentration*. It is important to note that the sediment transport-specific Dirichlet boundary takes C, not Ch, as the input value.

vegetation

The values that are stored as the quantity *vegetation* are numeric codes (positive integers) for different types of vegetation, each with corresponding values of stem diameter and stem spacing in a look-up table. The default value for *vegetation* is zero, which is the code for no vegetation. The values of this quantity can be set using the existing functions in ANUGA or ones included in the directory `/file_conversion` and the file `sed_transport_utils.py` (see description in section 3).

If any entry of *vegetation* is greater than zero, the operators will attempt to import the csv file specified in the function argument *vegfile* of `Vegetation_operator`.

An exception is raised if the file is not found or if *vegfile* is not specified. In the case of a domain with two different types of vegetation ($vegetation > 1$) as well as bare areas ($vegetation = 0$), this file would contain a column *vegcode* with the corresponding code in the quantity *vegetation*, and two columns for the values of *stem diameter* and *stem spacing* (in meters).

```
vegcode, stem_diameter, stem_spacing
1, 0.01, 0.1
2, 0.01, 0.3
```

Listing 3: Example of vegfile

diffusivity

The effects of turbulence on the momentum of the flow can be calculated using these operators by harnessing the existing kinematic viscosity operator (`domain.set_use_kinematic_viscosity`) and dynamically modifying the values of the quantity *diffusivity* to reflect the conditions of the flow. An exception is raised by the operators if the argument *turbulence* is True but the quantity *diffusivity* does not exist. A message is then displayed and the run continues without calculating turbulence.

The values for *diffusivity* are calculated using the equation:

$$K_d = \sqrt{k_e}\, l + ad\, U_{ref}\, d_s \tag{2}$$

where l is the mixing length, U_{ref} is the flow velocity felt by the stems, d_s is the stem diameter, $ad = d_s^2/\Delta S^2$ is the fractional volume of the flow domain occupied by plants, and ΔS is the mean stem spacing. The velocity scale $\sqrt{k_e}$ is given by:

$$k_e = C_b\, U_{ref} + (\overline{C_D}\, ad)^{2/3}\, {U_{ref}}^2 \tag{3}$$

where C_b is the bed drag coefficient ($C_b = 0.001$) and $\overline{C_D}$ is the bulk drag coefficient for the vegetation array. If the stems are approximated as emergent cylinders, $\overline{C_D} = 1.2$. The mixing length l is given by:

$$l = h \ - \ \frac{h - h(ad - 0.005)}{0.005} \tag{4}$$

where h is the flow depth. The second term of all three equations is equal to zero when vegetation is not present ($ad = 0$). This equation is an approximation of the data shown in *Nepf* [1999], figure 6.

2.2 Creating the operators

The `Sed_transport_operator` and `Vegetation_operator` can be used independently or simultaneously in the same domain. They correspond to two "public" classes in the file `sed_transport_operator.py` that re-route the operator calls to one of three internal classes depending on whether one operator is being used alone or if both are called from the same file. These three internal classes, `Sed_transport_only`, `Vegetation_only`, and `Vegetation_and_Sed`, draw from the same set of shared functions. Using `Vegetation_and_Sed` instead of both single-process operators simultaneously, however, avoids potential problems with the order of operators and is more computationally efficient.

To choose which internal class to use, the creation of the `Sed_transport_operator` or `Vegetation_operator` checks if the other operator has already been added to `domain.fractional_step_operators`. If it is not found, it initializes its own independent internal class and adds it to `domain.fractional_step_operators`. If the other operator already exists, it cleans anything that operator had already created, removes it from `domain.fractional_step_operators`, and re-directs to the joint internal class.

Both `Sed_transport_operator` and `Vegetation_operator` accept boolean arguments, False as default, for calculating turbulence and other sinks and sources of momentum. Changes to momentum due to drag on vegetation are always recorded if the corresponding operator is created. A function for incorporating sediment diffusion in the sediment flux calculations is currently unstable and commented out in the code.

As an example, creating only `Sed_transport_operator` in the run file will re-direct the operator call to the internal class `Sed_transport_only`:

```
from anuga.operators.sed_transport.sed_transport_operator \
    import Sed_transport_operator

op1 = Sed_transport_operator(domain,
                             erosion = True,
                             deposition = True,
                             turbulence = True,
                             momentum_sinks = True,
                             description = None,
                             label = None,
                             logging = False,
                             verbose = True)
```

Listing 4: Creating a sediment transport operator

The `Vegetation_operator` can also be created individually and will re-direct the operator call to the internal class `Vegetation_only`:

```
from anuga.operators.sed_transport.sed_transport_operator \
    import Vegetation_operator

op2 = Vegetation_operator(domain,
                          vegfile = 'vegcodes.txt',
                          turbulence = True,
                          momentum_sinks = True,
                          description = None,
                          label = None,
                          logging = False,
                          verbose = True)
```

Listing 5: Creating a vegetation drag operator

Vegetation drag and sediment transport can be calculated simultaneously by creating both operators, in any order. This redirects the operator call to the internal class `Vegetation_and_Sed`:

```
from anuga.operators.sed_transport.sed_transport_operator \
    import Vegetation_operator, Sed_transport_operator

op1 = Sed_transport_operator(domain,
                             erosion = True,
                             deposition = True,
                             turbulence = True,
                             momentum_sinks = True,
                             verbose = True)
op2 = Vegetation_operator(domain,
                          vegfile = 'vegcodes.txt',
                          turbulence = False,
                          momentum_sinks = False,
                          verbose = False)
```

Listing 6: Creating a vegetation drag operator

The function arguments *turbulence*, *momentum sinks*, and *verbose*, False by default, will be set to True if they are True for either of the operators.

2.3 Operator call

The operators act only on vertices with a flow depth greater than a minimum depth (set in `sed_transport_config.py` as `min_depth=0.005m` as default) in order to avoid extremely high erosion rates due to abnormally high velocities at low flow depths and to minimize the likelihood of thicknesses of aggradation that exceed the water depth, and permits the use of the operators without having to reduce the length of the timesteps.

The order of operations when both `Sed_transport_operator` and `Vegetation_operator` are called is:

1. Calculate drag on vegetation, compute the decreased flow velocity
2. Calculate erosion, deposition, and change in concentration

3. Compute sediment flux, update quantity *concentration*
4. Update quantity *elevation*
5. Compute sources and sinks of momentum, update quantities *xmomentum* and *ymomentum*
6. Compute turbulence, update quantity *diffusivity*

2.3.1 Drag on vegetation

The drag force F_D that vegetation imparts on the flow is given by:

$$F_D = \frac{1}{2}\, C_D\, a\, {U_{ref}}^2 \tag{5}$$

where $a = d_s/\Delta S^2$ is the projected plant area per unit volume (vegetation density per meter). This assumes that the vegetation can be approximated as a regular array of cylindrical stems that emerge from the flow. The drag force is allowed to reduce the flow velocity to zero but not to change its direction.

2.3.2 Sediment transport processes

The rate of change of elevation of the bed is a mass balance equation written as:

$$\frac{\partial z}{\partial t} = \frac{\dot{D} - \dot{E}}{1 - \phi} \tag{6}$$

where z is the elevation of the bed above some datum, ϕ is the porosity of the bed material, and $\dot{D}$ and $\dot{E}$ are the rates of sediment deposition and entrainment. The entrainment rate is given by:

$$\dot{E} = K_e\, (\tau^* - {\tau_c}^*) \tag{7}$$

where K_e is an erosion coefficient and ${\tau_c}^*$ is the dimensionless critical shear stress. τ^* is the dimensionless shear stress that the flow applies on the bed:

$$\tau^* = u^*\, \frac{1}{R\, g\, d_g} \tag{8}$$

where $u^* = \sqrt{f/8}$ is the shear velocity, f is the Darcy friction factor ($f = 0.05$), d_g is the grain diameter, $R = (\rho_s - \rho_w)/\rho_w$ is the submerged specific gravity of sediment, ρ_w and ρ_s are the densities of water and sediment, and g is the gravitational acceleration. K_e is given by:

$$K_e = {K_e}^*\, d_g\, \sqrt{R\, g\, d_g} \tag{9}$$

where ${K_e}^*$ is a dimensionless erosion coefficient (${K_e}^* = 12$).

The rate of aggradation $\dot{D}$ can be written as:

$$\dot{D} = D^*\, C\, v_s \tag{10}$$

where D^* is a dimensionless number equal to 1 if the sediment flux is uniformly distributed throughout the depth of the flow, and v_s is the settling velocity of grains in the fluid [*Ferguson and Church*, 2004]:

$$v_s = \frac{R\, g\, {d_g}^2}{C_1\, \nu + (0.75\, C_2\, R\, g\, {d_g}^3)^2} \tag{11}$$

where C_1 and C_2 are 18 and 0.4 for smooth spheres, and ν is the kinematic viscosity of the fluid.

The time rate of change of concentration of sediment in the flow can be written as:

$$\frac{\partial Ch}{\partial t} = \dot{E} - \dot{D} - \frac{\partial q_{s_x}}{\partial x} - \frac{\partial q_{s_y}}{\partial y} \tag{12}$$

where q_{s_x} and q_{s_y} are sediment discharges per unit width in the x and y directions.

2.3.3 Sources and sinks of momentum

Four sources or sinks of momentum can be included by these operators into the equations for conservation of momentum: (1) friction loss, (2) loss due to spatial variations in sediment concentration, (3) loss of momentum due to the exchange of mass between the flow and the bed, and (4) loss due to fluid drag on vegetation. Changes to the momentum of the flow due to turbulence are calculated when the function argument *turbulence* is True.

1. Loss due to friction is computed using the existing `manning_friction_implicit` function
2. Loss due to spatial variations in concentration is given by the equation:

$$P_C = \frac{(\rho_s - \rho_w)gh^2}{2(\rho_w(1-C) + \rho_s C)} \nabla \cdot C \tag{13}$$

3. Loss due to the exchange of mass between the flow and the bed is given by:

$$P_e = \frac{\dot{D} - \dot{E}}{1-\phi} \left(\frac{\rho_w \phi + \rho_s(1-\phi)}{\rho_w(1-C) + \rho_s C} - 1 \right) \vec{v} \tag{14}$$

   where $\vec{v}$ is the flow velocity.

4. Loss of momentum due to fluid drag on vegetation is given by:

$$P_d = -\vec{v}\vec{F_D}h \tag{15}$$

   where $\vec{F_D}$ is the drag force of vegetation on the flow. The quantities *xmomentum* and *ymomentum* are modified by this equation whenever the `Vegetation_operator` is used, regardless of the `mometum_sinks` argument.

2.3.4 Updating quantities

All operators listed in `fractional_step_operators` are called at every timestep by `domain.evolve` after the flow calculations have been performed and the conserved quantities updated.

Both the `Sed_transport_operator` and `Vegetation_operator` directly modify the values of several quantities through functions in `sed_transport_mesh.py`.

- *Momentum* is modified by using the `explicit_update` (re-set to zero) and `update` methods of the quantities *xmomentum* and *ymomentum*.
- The shape of the bed is altered by using the `set_values` function at the vertices for the quantity *elevation*, followed by `smooth_vertex_values()` to eliminate discontinuities in the topography and distribute the updated elevations to the centroids and edges.
- The flux of sediment across the cell edges is computed by (1) obtaining the total volume of sediment in the flow within each cell from the values of the quantity *concentration* at the centroids, (2) finding the normal component of the flow velocities at the edges of the cells, (3) calculating, from the values of *concentration* at the edges, the volume of sediment that crosses each edge within the timestep, and (4) integrating the flux in and out of each cell across all edges to (5) change the volume of sediment present in the flow within each cell. This volume is then (6) converted back to a sediment concentration (of the form Ch) at the centroid of each cell, and (7) the updated value at each vertex is calculated by averaging the value at the centroids around it. (8) The quantity *concentration* is updated using the function `set_values` at the vertices.

3 Utilities

The file `sed_transport_utils.py` contains a set of functions, similar to existing ones, that are tailored to the needs of these specific operators.

`create_domain_from_regions_sed(...)`
: This function is equivalent to `__init__.create_domain_from_regions` but accepts lists of quantities to be created within the domain. See example in section 2.1.

`Reflective_boundary_Sed(domain)`

This boundary type is equivalent to `Reflective_bondary` but manages the boundary value for *concentration*.

`Dirichlet_boundary_Sed([stagexmomymomC])`

This boundary is equivalent to the existing `Dirichlet_boundary` but accepts a fourth entry for fractional concentration (C, not Ch) at the boundary. This boundary type should only be used for inflow boundaries. Flow outlets should be declared using the regular Dirichlet boundary type, which will allow sediment to be carried across the boundary by the flow instead of fixing the concentration at the cell edges. An exception is raised if a `Dirichlet_boundary_Sed` is has a fixed stage that is lower than the lowest elevation in the domain.

```
from anuga.operators.sed_transport.sed_transport_utils \
    import Reflective_boundary_Sed, Dirichlet_boundary_Sed

Bd = Dirichlet_boundary_Sed([1527.3, 0., 0., 0.3])  # Inlet (30% sed)
Bi = anuga.Dirichlet_boundary([1520, 0., 0.])           # Open outlet
Br = Reflective_boundary_Sed(domain)                 # Reflective wall

domain.set_boundary({'bottom' : Bi,
                    'side1' : Br,
                    'side2' : Br,
                    'top' : Bd,
                    'side3' : Br,
                    'side4' : Br,
                    'exterior' : Br})
```

Listing 7: Using operator-specific boundary types

`set_quantity_NNeigh(quantity_name,filename)`

Equivalent to the function `set_quantity`, it assigns values from the point file *filename* to quantity *quantity name*. In contrast to `set_quantity`, which interpolates between the values in the point file to obtain the value at the centroids, this function uses a Nearest Neighbour algorithm to assign each centroid the value of the nearest point in the file. This is necessary when assigning values from an ASCII file to the quantity *vegetation*.

Two files in the directory `operators/sed_transport/file_conversion` can be used to transform ASCII files into point files that represent quantities other than *elevation*.

`generic_asc2dem(name_in,quantity_name,...)`

Equivalent to `anuga.asc2dem`, this function transforms an `.asc` file into a .dem file. While the original function assumes that the values in the files will be assigned to the quantity *elevation* (and codes this into the files), this function allows the user to define the name of that quantity in the argument *quantity name*.

`generic_dem2pts(name_in,quantity_name,...)`

Equivalent to `anuga.dem2pts`, this function transforms a `.dem` file into a .pts file for future use with the quantity *quantity name*. This function does not create that quantity or set its values.

```
from anuga.operators.sed_transport.file_conversion.generic_asc2dem
    import generic_asc2dem
from anuga.operators.sed_transport.file_conversion.generic_dem2pts
    import generic_dem2pts
from anuga.operators.sed_transport.sed_transport_utils
    import set_quantity_NNeigh

generic_asc2dem('veg.asc',
                quantity_name = 'vegetation',
                use_cache = False,
                verbose = True)

generic_dem2pts('veg.dem',
                quantity_name = 'vegetation',
                use_cache = False,
                verbose = True)
```

```
set_quantity_NNeigh(domain,
                    'vegetation',
                    filename='veg.pts')
```
Listing 8: Using file conversion and set quantity utilities

4 Tests

A suite of unit tests and analytical solutions will be added to the operator package in the immediate future.

References

Davy, P., and D. Lague (2009), Fluvial erosion/transport equation of landscape evolution models revisited, *Journal of Geophysical Research-Earth Surface*, *114*(F3), F03,007, doi:10.1029/2008JF001146.

Ferguson, R., and M. Church (2004), A simple universal equation for grain settling velocity, *Journal of Sedimentary Research*, *74*(6), 933–937, doi:10.1306/051204740933.

Kean, J., and J. Smith (2006), Form drag in rivers due to small-scale natural topographic features: 2. Irregular sequences, *Journal of Geophysical Research*, *111*(F4), F04,010, doi:10.1029/2006JF000490.

Nepf, H. (1999), Drag, turbulence, and diffusion in flow through emergent vegetation, *Water Resources Research*, *35*(2), 479–489, doi:10.1029/1998WR900069.

Simpson, G., and S. Castelltort (2006), Coupled model of surface water flow, sediment transport and morphological evolution, *Computers & Geosciences*, *32*, 1600–1614.