

Determining Valley-bottom sediment volumes

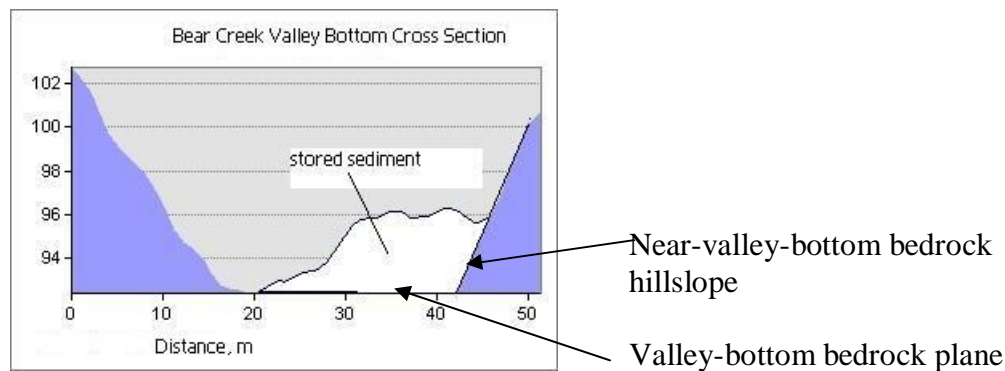
Goal: To determine the amount of sediment stored in valley-bottom reservoirs (e.g. terraces, debris fans, debris dams, etc.) of headwater streams in the Oregon Coast Range (specifically, parts of the upper watersheds of Knowles & Wasson Creeks).

Existing Data:

- 1m DEM (from filtered LiDAR points, i.e. bare-earth)
- Bedrock in the stream channel bottoms (digitized from field data)
- (ASCII files of filtered and unfiltered LiDAR last returns; currently, we don't plan on using these)
- Could convert data to another format (e.g. TIN)

Proposed Methodology:

- Interpolate between digitized bedrock points (on the DEM) to create a valley-bottom bedrock surface.
- Take cross sections on the DEM to get the lower portions of hillslopes; subtract out the lowest part of this slope that is much less steep (this is the surface of stored sediment; see below), then interpolate between remaining hillslopes segments to create (approximate) bedrock valley sides.
- Combine the bedrock valley-bottom (step A) with the bedrock valley side (step B), then subtract from the DEM to determine sediment volume.



Following is pseudocode for how to do the analysis. Some steps are clearer than others about exactly how to do it, and there is likely more than 1 way to do many of them.

Notes: -% *italicized text preceded with % indicate comments*

The term 'scanline' here refers to a cross-section polyline (i.e. the surface in the figure above)
left bank = LB, right bank = RB, each looking downstream

-Draw polyline that approximates the valley centerline (*note: by hand should be good enough*)

Pseudocode:

Define upstream end of centerline = start = 0

Set CD=1 % stands for 'Centerline Distance', which denotes at which point along the centerline you are

CenterlineLength= measured length of centerline

Create valley-bottom bedrock surface

-Focalmin DEM to get lowest elevation within specified distance (e.g. 3-5 m) for each cell

For each stream:

- Assign elevation to bedrock points
 - Separate top of bedrock steps from regular points

- Assign elevation to each point based on its underlying grid cell (Focalmin DEM for all points except top of bedrock steps, which uses the 'regular' DEM)
- Create a plane between two bedrock points using their elevations (lines in the plane perpendicular to the line connecting the two points have constant elevation); repeat for all successive pairs of bedrock points along the stream reach
 - *How to do this??*
 - Probably best to limit the width of the created plane (125m on each of the centerline should be sufficient;); (this limitation isn't necessary since it gets deleted anyways)
- Line intersecting two successive planes could either:
 - Bisect the angle between the two polyline segments; or
 - Simply be the intersection of the two planes
- Where valley curves between two bedrock points, need to follow it:
 - Slope could be constant between two points (ok if they are close enough); or
 - Approximate the localized slope
-

Determine near-valley-bottom sediment volumes

% Note: all subroutines that have a "LB" in them are to be repeated for "RB" (for simplicity, RB was not typed each time)

For i=0 to n (n=125) % creates initial scanlines to work with; important to have n large enough to span all tributary valley mouths

If i < > node % don't create scanlines at node (which is where the valley centerline changes direction)

Create Scanline(i) % to save computer memory, it could be the narrower of either 125 m on either side or ~ 25 m elevation gain

End

FSL = n % 'Front ScanLine' refers to the scanline that was most recently created (i.e. that is at the front as you move downstream)

If CD=<CenterlineLength, then

% SUBROUTINE 1: Following subroutine determines if there is a junction with a tributary valley on either side of the centerline

For each side of the valley % i.e. LB, RB

-Calculate average slopes on upper parts of Scanline(CD) by looking at points 5m elevation apart, between 10 & 20 m above lowest point on scanline(CD) (% 10 m because need to get above valley-bottom deposits which have shallow slopes)

-If average slope is less than 30 degrees % determines if it is a tributary junction

Set LBtrib(CD)= yes

-Else, set LBtrib(CD)=no

End

% following loop determines the transition between hillslopes and valley-bottom (tricky; might need to do this by hand)

For each side of valley

-If LB_{Trib}(CD)=no

-use scanline(CD-3)...scanline(CD+3) to determine break in slope *% need to use directional smoothing since data is noisy; only use scanlines that are not part of tributary valley*

-delineate point LB(CD)1 at hillslope – valley-bottom transition of scanline(CD)

Else, end

End

% following loop determines points defining lower part of bedrock valley slopes (note: assumes bedrock at the surface, which is a decent approximation since hillslope soils are thin (~ 0.5 m))

For each side of valley

If LB_{Trib}(CD)=no

-On scanline(CD), mark a point, LB(CD)2, higher (e.g. ~3 m elevation above) than LB(CD)1 of the cross-section; *% might need to be at either a higher or lower elevation than this)*

-Draw a line from the higher point (LB(CD)2) through the lower point (LB(CD)1, to the valley-bottom bedrock surface

-make a point, (LB(CD)3, at the intersection of scanline(CD) with the valley bottom bedrock

% Note: want to create a file to store all LB(CD)1 and LB(CD)3, but not LB(CD)2

End

%Following creates a 'virtual' hillslope across a tributary valley and determines sediment volumes there; Note: this subroutine needs controls to ensure the 2 planes that cross the tributary mouth do not protrude too far into either the mainstem or the tributary valley

If LB_{Trib}(CD)=yes AND LB_{Trib}(CD-1)= no *% this marks transition between scanlines that are at a tributary and those that are not*

US=CD-1 *% delineates last UpStream nontributary point*

For j=5 to 15 *% 5 to 15 ensures that the virtual hillslope plane is not defined by the rounded part of the mainstem to tributary valley wall transition*

-create US_LB(US-j)i (for i=1 and 3) *% creates points defining upstream plane that crosses tributary valley*

End

For LB_{Trib}(CD)=yes

-call SUBROUTINE 1 *% subroutine determines if at tributary valley*

-CD=CD+1

End

% following creates points which define downstream half of tributary plane

DS=CD *% delineates first DownStream nontributary point*

For j=5 to 15 *% 5 to 15 ensure that the virtual hillslope plane is not defined by the rounded part of the mainstem to tributary valley wall transition*

-create DS_LB(DS+j)i (for i=1 and 3) *% creates points defining downstream*

plane that crosses tributary valley

End

-construct a surface from 2 cross-tributary planes: (*% takes care of tributary junction that occurs at bend in valley*)

1 starting at US-5 (defined by points US_LB(US-5)1,3 and US_LB(US-15)1

1 starting at DS+5 (defined by points DS_LB(DS+5)1,3 and DS_LB(US+15)

Neither plane extends beyond its intersection with the other

For CD=US to DS

-Create point LB(CD)1 where scanline(CD) intersects cross-tributary plane

-Create BRScanline(CD) connecting points LB(CD)1_LB(CD)3_RB(CD)3_RB(CD)1
% this creates a BedRock scanline

-Subtract BRScanline(CD) from surface scanline(CD)

-store sediment volumes for each valley-bottom grid cell of the scanline (only if cell has not yet been populated *% makes sure not computing twice when centerline bends*)

-add all these volumes to get an integrated sediment volume/length of valley centerline (at point CD) and stream length *%need to add subroutine to do stream length*

-ValWidth(CD) = distance between LB(CD)1 and RB(CD)1 *% stores valley width as a function of location along valley centerline*

-CD=CD+1

-add 25 volume/length for a rolling window of 25 consecutive centerline points

End

End

% Takes care of valley centerline changing direction (i.e. at a node)

If CD at a node

- create, then connect, 2 toe-of-hillslope lines that mark bottom of virtual hillslope on outside of bend (e.g. one extending LB(CD-3) to LB(CD-1) and another extending LB(CD+3) to LB(CD+1)

-Create scanlines from node to each cell of virtual toe-of-hillslope lines, in angle between scanline(CD-1) and scanline(CD+1)

End

% Following creates bedrock-surface scanline and associated sediment volumes for non-tributary valley scanlines

If LBtrib(CD)=no AND RBtrib(CD)=no

-Create BRScanline(CD) connecting points LB(CD)1_LB(CD)3_RB(CD)3_RB(CD)1

-ValWidth(CD) = distance between LB(CD)1 and RB(CD)1 *% stores valley width as a function of location along valley centerline*

-Subtract BRScanline(CD) from surface scanline(CD)

-store sediment volumes for each valley-bottom grid cell of the scanline(CD) (only if cell has not yet been populated *% makes sure not computing twice at a node*)

-add all these volumes to get an integrated sediment volume/length of valley centerline (at CD) and stream length

-add 25 volume/length (i.e. at CD) for a rolling window of 25 consecutive centerline points

End

% move down the valley centerline to the next scanline

If FSL<CenterlineLength *% Only creates new scanlines if the front one is not at the end of the valley centerline length*

Delete from memory scanlines from before CD-n/d2 *% don't need to store 'old' scanlines (i.e. only need to keep scanlines in a rolling window of 125 m)*

Move 1 m downstream on valley centerline

FSL=FSL+1

If FSL< > node *% don't create scanlines at node (which is where the valley centerline changes direction)*

Create Scanline(FSL) *% to save computer memory, it could be the narrower of either 125 m on either side or ~ 25 m elevation gain*

End

CD=CD+1

End

Volume-weighted sampling locations

% Need to choose n sampling sites (probably n=30 for 4 catchments) from volume-weighted grid cells, then translate to nearest streambank.

Extra credit:

- Contributing area
- Stream-wise distance from outlet
- Local valley and stream slope (*should be relatively easy to incorporate, using local minimum DEM since trees across channel would get in way*)
- Determine cut bank height (*probably hard to do very accurately with data we have*)