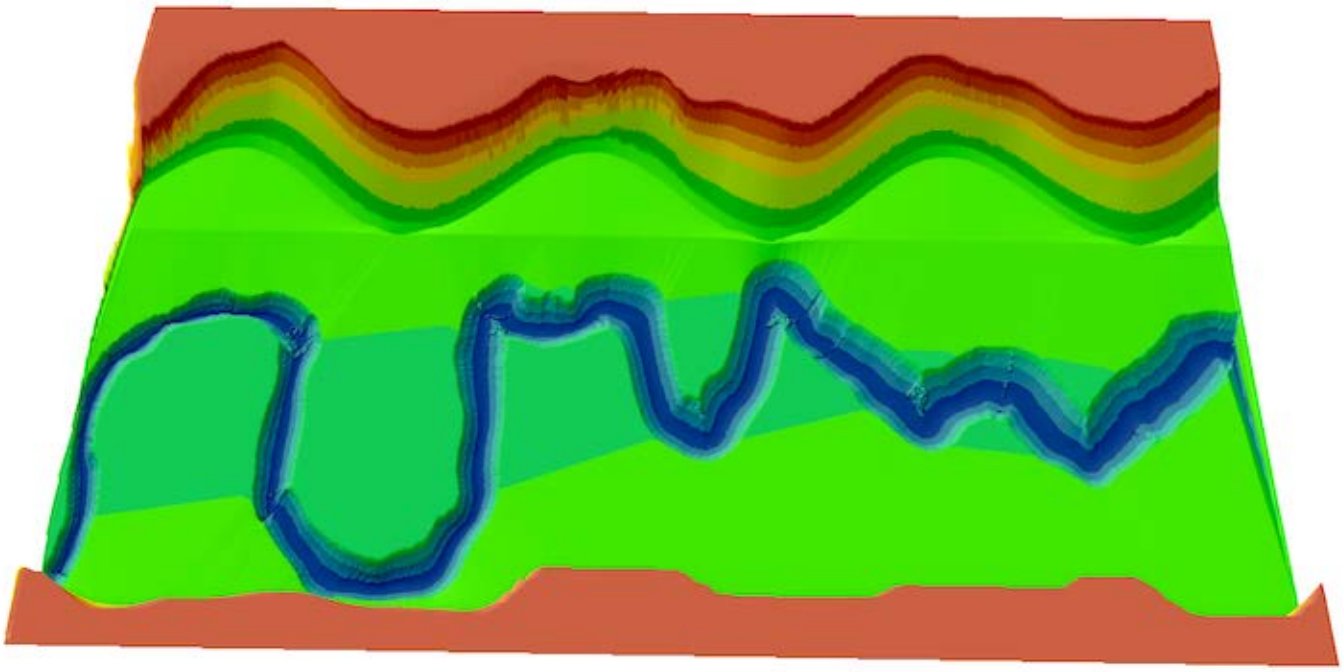
College of Agricultural & Environmental Sciences
Department of Land, Air, and Water Resources
UC Davis

River Builder User's Manual For Version 1.0.0



May
2020

By: Gregory B. Pasternack and
Muwei Zhang

UCDAVIS

Cite As: Pasternack, G. B. and Zhang, M. 2020. River Builder User's Manual For Version 1.0.0. University of California, Davis, CA.

Location (URL):

Creators:	Prof. Gregory Brian Pasternack and Muwei Zhang
Title:	River Builder User's Manual For Version 1.0.0
Publisher:	
Publication year:	2020
Resource type:	Text/Text
Description [Other]:	Software manual for the use of River Builder to create synthetic river valleys in Python.
Subjects:	geomorphology, river topography, river design, river engineering
Rights:	The Regents of the University of California, Davis campus, 2014-19

For information contact **gpast@ucdavis.edu**

1 USER'S MANUAL PURPOSE

The purpose of this manual is to provide you with an explanation of River Builder 1.0.0 software that has the capability to design rivers that meet regional, reach-scale geomorphic expectations, but go far beyond that to have multiple scales and layers of organized sub-reach variability. Certainly, there are still many river archetypes the software cannot yet produce- notably step-pool, cascade, anastomosing, and braided rivers. We welcome future developments that expand the software's functionality more through time.

In most cases we expect people will use this software to design examples of real rivers to yield improved natural outcomes, but in fact the underlying geometric modeling is capable of allowing you to let your imagination run wild and design rivers for other purposes, such as testing dysfunctional river archetypes and creating unnatural, imaginary rivers for fictional purposes (e.g., video games, movies, animations, etc.). Whether you need a digital river design to fix a degraded real river or to save yourself millions of dollars in manual artistic effort for your next 4K resolution open-world fantasy game, this software can be very useful to meet your needs.

2 INTRODUCTION

Designing alluvial river channels that behave naturally is a central challenge facing river scientists and engineers as well as animators and video game developers in the 21st century (Brown and Pasternack, 2019). Though organized and responsive to driving forces, rivers exhibit complex patterns and processes from the scale of an individual grain of sediment to that of a large continent. Despite roughly a century of research it remains highly uncertain as to which patterns and processes are most important to design and build explicitly versus which ones should be allowed to emerge on their own after construction. Too little research has focused on variability in rivers. Nevertheless, our understanding of the fluvial patterns and processes as well as our ability to quantify them is increasing rapidly. We can now design much more dynamic rivers than ever before.

It is beyond the scope of this manual to explain the entire scientific foundation of this software. Brown and Pasternack (2019) provide a thorough literature review on the history and diversity of approaches to designing rivers, so readers new to this content are directed there for an in depth presentation. This introduction section offers a simplified, generalized overview of the context of river design necessary for this software. The website at http://pasternack.ucdavis.edu has a lot of free educational video podcasts about rivers and related topics as well as web pages that provide more explanation of underlying concepts. Much more is available on the internet and in numerous textbooks. Most people will likely seek to just get started with the software and learn on an as-needed basis as they move along, so that is understandable.

2.1 River Architect Software

Before getting into how this software works, it is important to clearly state that this software has one and only one use- ***to create channel designs***. That's it. You know what you want a channel to be, and once you specify that then this program will procedurally render that for you. If you do not know what you want, but you want some channels, you can always take a look at the established set of template designs we provide. The main output is a comma-delimited (.csv) point coordinate file. You can import this into GIS, CAD, hydraulic modeling, or other software to

achieve other functionality.

Because users also need to analyze rivers as well as create them, we have also created a sibling software platform called "River Architect" to evaluate a river with respect to ecohydraulic, geomorphic, and economic considerations (Schwindt et al., 2020). River Architect is programmed in Python3 and is now available open source and free to the public on Github at the link below.

https://riverarchitect.github.io/main_page

2.2 Vanilla Rivers

Decades of empirical study of longitudinal and lateral transects of alluvial rivers have yielded an explanation for the central tendency of channel geometric variables averaged over a length of 100-1000 channel widths (i.e., the reach scale) to grow or decline with discharge on the basis of mutual adjustment in order to pass the typical water and sediment delivered by the catchment. Such variables include slope, bankfull width, bankfull depth, sinuosity, entrenchment ratio, and median particle size. For any river reach there can be large uncertainties in the average values of geometric variables compared to empirical regional expectations. Typical uncertainties for channel width tend to be ~ 30-50% on average, but for individual locations can be much higher. Depth is more certain than width. How accurate geometric specification needs to be will vary depending on what geomorphic processes one seeks to instill, which is beyond the scope of this manual. Users are encouraged to use other software to test whether designed river terrain yield the processes and conditions in their design hypotheses (e.g., Pasternack et al., 2004; Brown et al., 2015).

A synthetic river designed only according to reach-scale metrics of central tendency is called a "***Vanilla River***" (Figure 1). Although vanilla can be a delicious flavor, it is colloquially considered plain, ordinary, without taste or distinction, and generally lacking in desirable variations and complexity. This is apropos, because in fact few river processes are driven by the central tendency of reach-scale river metrics. Instead, they are driven by local patterns of topographic variability. Thus, many natural rivers do not look like synthetic Vanilla Rivers and we must turn to a more sophisticated understanding of topographic complexity to design rivers that reflect their natural patterning.

This software is capable of designing Vanilla Rivers, if that is desired. Vanilla Rivers, including ditches and canals, could be important to design for use in highly managed landscapes. For use in real rivers, we do not recommend stopping with only reach-averaged design metrics.

Figure 1. Vanilla River examples, (a) synthetic, (b) real.

2.3 Channels Within Valleys

Channels exist within valleys and canyons. River Builder software can be used to make nothing more than a simple channel bounded by a wide, flat valley floor if that is all a user wants to have. However, the program was designed for creating entire synthetic river valleys to have maximum utility for a wide range of applications. This was chosen, because free-flowing rivers require lateral as well as longitudinal connectivity (Grill et al., 2019). Thus, a river designer should be mindful of lateral connectivity in their design, regardless of how far out form the centerline they intend to design. There are many examples of real-world scenarios in which a designer would want to overhaul an entire river valley (Figure 2). Such scope of construction is very challenging today due to the fuel cost for earth movement, but in a future society that maximizes its renewable energy potential, especially solar and wind, there would be no limits to energy usage and so massive earth movement for environmental restoration would become readily feasible, especially if performed by autonomous construction drones. Therefore, we need to provide the tools now for the scope of river restoration engineering that society will need when it awakens to address the global ecological collapse that is underway.

To prepare for use of River Builder, consider two ways channels are nested within valleys.

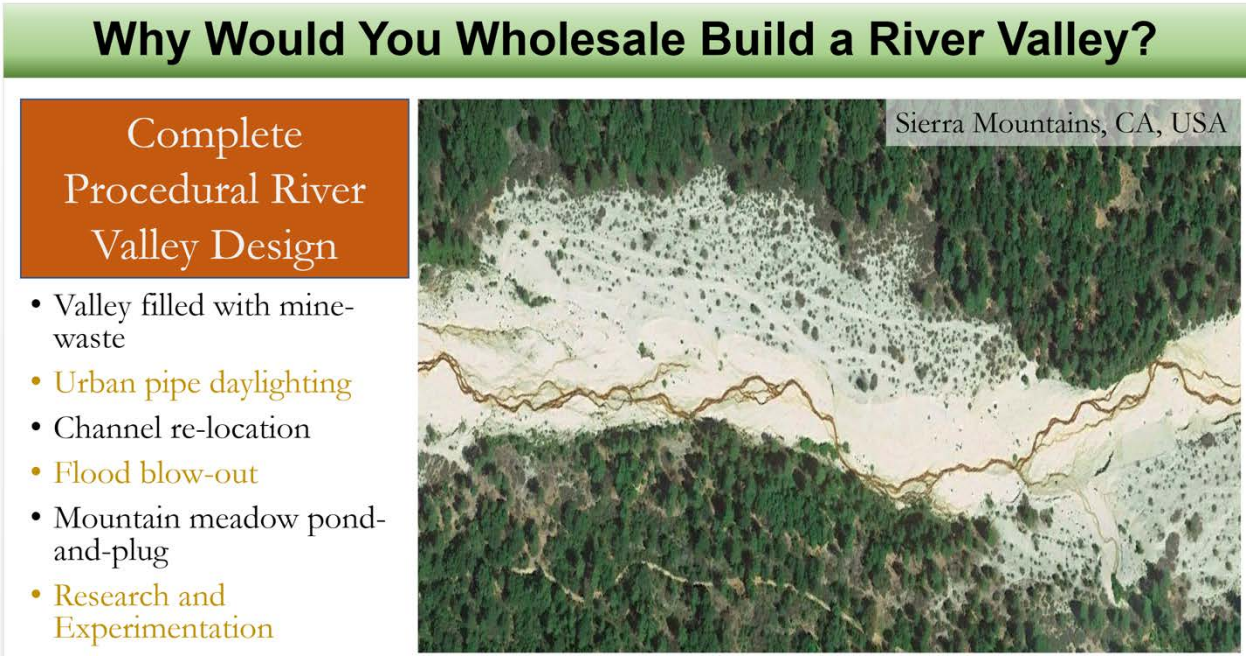


Figure 2. Examples of why a user might design a river valley, not just a channel.

2.3.1 Planform nesting

One aspect in which channels need to be nested into a valley relates to their respective centerlines when view in overhead plan view. A simple scenario would involve a linear channel nested within the center of a linear valley (Figure 1a). However, there is no reason to force this. In the real world, channels meander and so do valleys (Figure 3).

Figure 3. A real sinuous channel nested in a sinuous valley near Novelty, WA, USA. Each has a different sinuosity and the channel's lateral position is not locked to be at the valley's centerline.

2.3.2 Cross-sectional nesting

Sometimes a river corridor is nothing more than a single U-shaped conduit carved by water into a flat valley floor, such as shown in Figure 1a. However, valleys often contain far more cross-sectional features. Features typically have starting and ending positions across the valley's cross-

section. These endpoints are commonly termed "breakpoints". When these points are projected into the third dimension (up and down the river) along whatever path they follow, they are then called "breaklines", even though they are not strictly lines.

In this manual, the term "breakline" will be used to refer to a plan view (aka top view) curvilinear function that denotes the starting and/or ending of a feature that runs down the river corridor.

A simple conceptualization with a bit more complexity might be a U-shaped channel within a floodplain within terraces within steep valley walls or uplands (Figure 4). Within geomorphology, it has long been recognized that valleys may have multiple terraces as a result of distinct periods of higher or lower sediment supply and/or sediment transport capacity (Blum and Tornqvist, 2000). A terrace is a relatively flat surface positioned between floodplains and uplands. More recently, it has become increasingly recognized that channels too often have many distinct flat surfaces within them, which has led to the development of the term "macro channel" to refer to a 'channel-in-channel' form where "a smaller, low flow channel is inset within a larger channel, and separated from the margins of the macrochannel by geomorphic units such as benches, ledges, and various bar types" (Thompson et al., 2016).

In light of this natural range of possibilities, River Builder software aims to allow users unlimited freedom to create as many in-channel and in-valley flat surfaces and associated lateral slope breaks, aka breakpoints (Figure 5). Each lateral slope break is essentially an elevation contour (when projected into the longitudinal direction) herein termed a "breakline". How this works will be explained later, but the basic concept is now established. It is even possible to create negative vertical offsets to obtain special features, such as levees, roads, and even multi-threaded channels (but such "yazoo" type channels may not intersect the main channel).

Because a design can have unlimited in-channel surfaces and lateral slope breaks, it is difficult to converse about some channel cross-sectional features in a universal way. Therefore, we coin the term "inner channel" to refer to the innermost conduit of water in the river valley (Figure 5). In some cases this may be the "bankfull channel", but in other cases it may be a base flow channel or whatever else someone wants to put there with as many nested channel surfaces adjacent to it before reaching the floodplain's overbank domain.

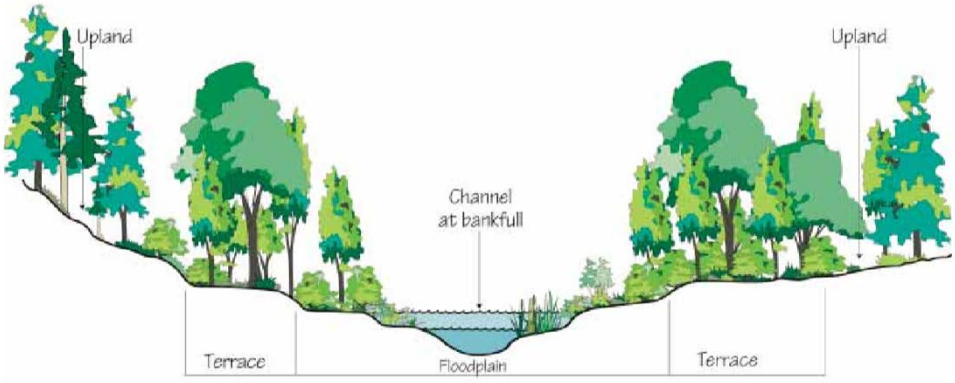


Figure 4. River valley cross-section archetype. Copied from Eubanks (2004).

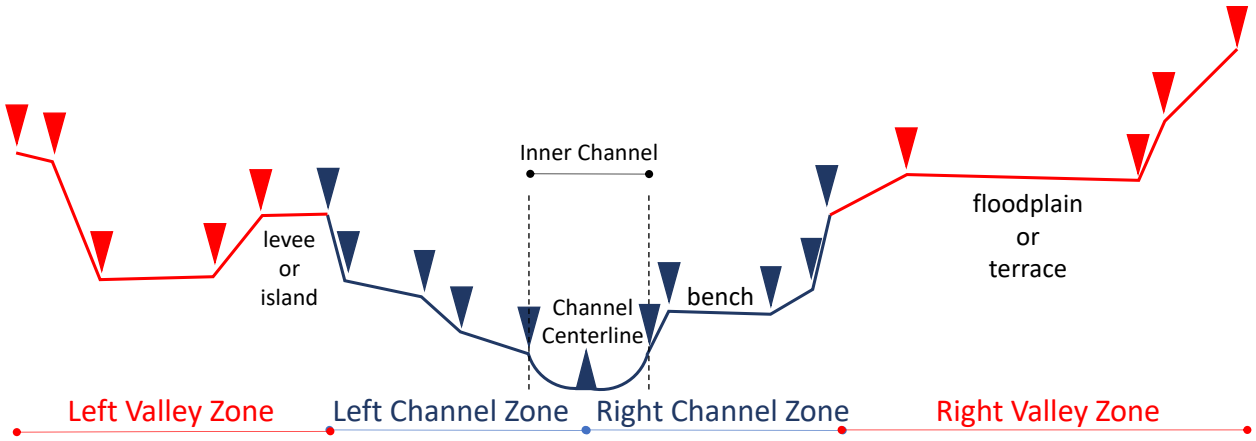


Figure 5. Conceptualization of the different cross-sectional zones (i.e. inner channel, left and right channel zones, and left and right valley zones) in River Builder. Within each zone, a user may specify an unlimited number of slope breaks (downward pointing triangles, aka "breakpoints") to create whatever flat surfaces and side slope angles they want. This is done with user-specified horizontal and vertical offsets. Shown is just one example.

2.4 Geomorphic Covariance Structures

Many measurable variables in geomorphology and allied sciences vary along a pathway, such as a river corridor (Figure 6, top). Variables could be flow-independent measures of topography, sediment attributes, flow-dependent hydraulics, topographic change, and biotic variables. Lane et al. (2017) reported that for a large region of California river variability metrics distinguished channel types better than traditional central tendency river attributes. These variations can contain some random aspects, but to a large degree they are highly organized, interlocked, and readable (Brown and Pasternack, 2014, 2017; Pasternack et al., 2018a,b). Also, river variations can be layered on top of each other, yielding multiple spatial scales of topographic complexity. Both geomorphic processes and ecological functions are more strongly governed by layered scales of spatial variability in topography than the central tendency of a river averaged over scales. In turn, both geomorphic and ecological processes are vital to maintaining spatial diversity across scales.

Brown and Pasternack (2014) coined the term "Geomorphic Covariance Structure" (GCS) to mean the linked bivariate pattern of any two river variables along a pathway. It is not the statistical covariance, which is a single number, but instead a new concept involving the complete bivariate spatial series from which a statistical covariance could be computed if desired (Figure 6, bottom). For example, one GCS could be the spatial series of the product of depth and width, making it a surrogate for cross-sectional area. Note that detrended, standardized bed elevation (Zs) is a surrogate for depth. The GCS between Zs and standardized width (Ws) is the basis for the hydro-morphodynamic mechanism of flow convergence routing (MacWilliams et al., 2006; Pasternack et al., 2018a,b). Another GCS could by the spatial series of the product of channel centerline curvature and width. There are many potential GCSs one can envision. The theory of Geomorphic Covariance Structures (GCSs) is not only useful for assessing the layers of topographic patterning of real rivers (Brown and Pasternack, 2014, 2017) but also for the design of synthetic rivers with more natural landforms that drive the real diversity of physical processes (Brown et al., 2014, 2015).

This software is capable of implementing GCSs to produce rivers with organized, coherent

patterns of variability. This is where the real power of this software lies. However, the software does not tell you what GCSs you need to produce different outcomes. You must have in mind what you want and an understanding of what GCS metrics are required to achieve that vision. Figuring that out is one of the primary open research lines in all of fluvial geomorphology today. It is the most important for advancing process-based river restoration. Take note that if you are highly uncertain if your design specifications will yield the geomorphic processes and ecological functions to seek, then it is important to undertake testing to evaluate design performance relative to your design hypotheses. An increasing number of predictive, mechanistic computer programs test river terrains this way, though they were developed to test real rivers, but nothing precludes their use for testing artificial river designs as well.

Over time we will aim to provide an increasing number of template files with different river archetypes you can use as a starting point, but that is still under development. As archetypes become available, they will be posted within the web site at the link below:

https://riverbuilder.ucdavis.edu/

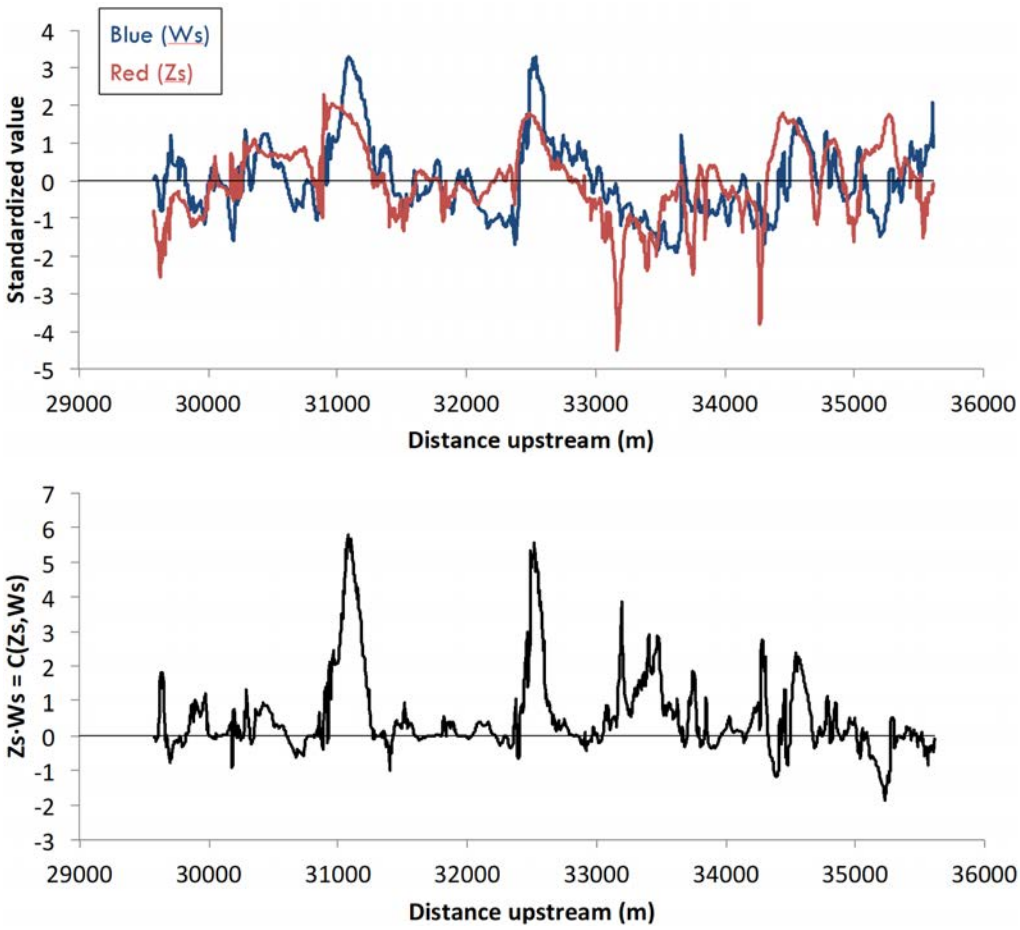


Figure 6. Sample spatial series of detrended, standardized bed elevation (Zs) and standardized width (Ws) (top) and an example geomorphic covariance structure (bottom), in this case the product Ws·Zs. Data taken from Pasternack et al. (2018b).

2.5 Past SRV Implementations

The underlying equations needed to make synthetic river valleys are already known, published, and non-proprietary. How those equations are organized into software is what is at issue here. Prior to the development of this version implemented in Python software, SRV algorithms were produced at UC Davis in three different software platforms for different purposes. First, there was the "RiverSynth" approach using Microsoft Excel®. This approach reproduces the examples in Brown et al. (2014) and has received some subsequent development with a few other features. It is very important to understand that these files can only create valley-orthogonal systems, and the equations are hard-wired with a valley-centric mindset. The formula in individual cells is unaware of and incapable of determining cross-sections that are perpendicular to the channel when the channel is not straight, so any lateral channel metrics for a sinuous centerline are not truly stream-wise.

Second, using an unrestricted donation, the Pasternack group at UC Davis sponsored World Machine, LLC to incorporate SRVs into the World Machine platform on the hopes that this would make this methodology more accessible. World Machine (http://www.world-machine.com) is a native geometric modeling platform for digital terrain development that allows for precise specification of parameters to design diverse landscapes through dialogue boxes and flowcharting. Implementations of World Machine with river design capabilities may not be available for free and may still be developmental, with little to no technical support. Be sure to touch base with World Machine, LLC to find out the current status of SRV tools in that platform as well as what support they offer to users to help learn their implementation of SRV tools and to address any bugs or problems that come up with your projects.

Third, the Pasternack group at UC Davis programmed a new implementation of the Brown et al (2014) equations in R and called it "River Builder". This version has more features and capabilities than the Excel version, but is still using the same math, so it is subjected to the same constraints. This version is available as open-source and free to the public on the CRAN website at this link: https://cran.r-project.org/web/packages/RiverBuilder/index.html.

3 Major Steps in Designing a Digital River

Brown et al. (2014) presented a seven-step method of channel-floodplain design (i.e. synthetic river valley (SRV) design) involving geometric modeling in which multiple scales of continuous equations are specified in each plane (XY, XZ, and YZ) and combined in a digital elevation model (DEM). From that starting point, the framework has further progressed over the years, yielding the current vision show in Figure 7 below that combines and simplifies the steps to portray them in a general workflow.

3.1 River Valley Conceptualization

First, you conceptualize the river corridor in terms of its essential elements and scales on the basis of ecogeomorphic goals. This is where you do your homework to figure out what it is you want to design and what your design aims to achieve. If your design is artistic, then you have vast freedom with the software to pursue your artistic vision. If your design is intended for use in real rivers or for scientific inquiry, the you should establish clear design hypotheses (as defined by

Wheaton et al., 2004) and the capability to translate those design hypotheses into flow-form-function connections (e.g. Lane et al., 2018)

One aspect of conceptualizing the river system you want to design is to specify the model domain, including coordinate systems, units, boundaries, and resolution. River Builder assumes a grid-type projected coordinate system with orthogonal {X, Y, Z} coordinates as the desired input and output. However, internally it transforms data into nested stream-wise and valley-wise coordinate systems so all computations go along the river. In terms of units, it just takes numbers in and produces numbers out without any reference to units. The only exception is an optional routine to estimate bankfull channel depth from bed material grain size assuming that at bankfull discharge the bed material would be right at a state of incipient motion. Therefore, the outputs are infinitely scalable simply by specifying units of whatever scale you want. A width of 60 could be 60 inches, 60 mm, 60 football fields, 60 km, etc. It's all arbitrary to meet your needs.

As part of the conceptualization process, you think through how you want to specify the geometric elements for each plane at all scales of interest that are going to be needed. All of these items are covered in detail later in the manual, but just for example purposes, they can include (but not limited to) channel centerline longitudinal profile, channel bed elevation longitudinal profile along the centerline, XS channel shape, XS valley shape beyond the channel, and the planform profile of every key contour in the river valley you intend to instill. The planform shape of every key contour can be held constant or vary downstream according a function or additive set of functions.

The steps just described are aggregated into the conceptualization box in Figure 7.

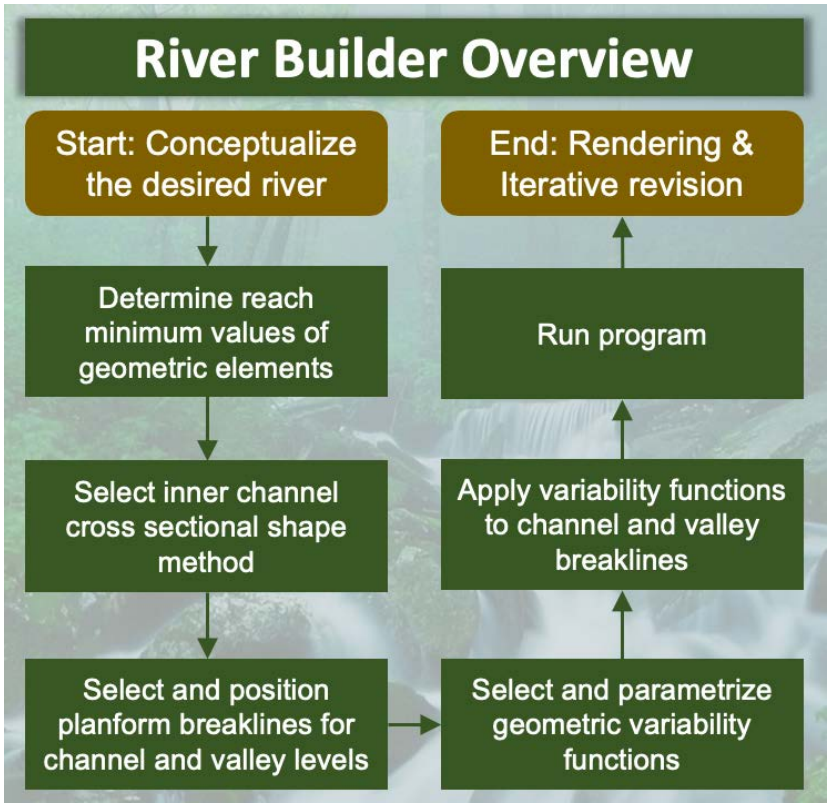


Figure 7. Simplified representation of the workflow for SRV design.

3.2 Initial Reach-Average Values

Second, you determine initial reach-average values for the geometric elements of the straight, valley-orthogonal Vanilla River yoiur design will be based on. You might imagine you could specify your final desired reach-average bankfull or inner channel dimensions at the outset, but this is not possible, because at this early stage in design you have not picked and set up your geometric variability functions, so who knows how these variables will be changing down the river. Thus, you are specifying initial values, and then there will be different options about how to handle the effect of variability later in the design process. Key values to specify include reach-average valley slope, inner channel width, inner channel depth, and then the lateral and vertical offsets for all in-channel and in-valley lateral slope breaks as illustrated in Figure 5. Some variables may be computed from theory and other variables to insure adherence with process concepts, such as computing mean bankfull depth from the Shields equation. Of course, any computation could introduce unit-specific values, so care is needed to insure all values are in the same units throughout all design steps.

3.3 Inner Channel Cross-Sectional Shape

Third, you have to specify the cross-sectional shape you want to use for the inner channel. Cross-sectional shape is specified as one type for the whole length of the river, but of course it will scale in size and proportion as a result of any changes in inner channel width downriver that you specify. There are three primary cross-sectional shapes you can choose from: symmetrical U-shape, engineered geometric shape, or asymmetrical U-shape (Figure 8). The symmetrical U-shape is the basic shape for a natural river that has low sinuosity. The engineered geometric cross-section option allows a user to create a straight-line trapezoidal shape that can be anything from a triangle to a trapezoid to a rectangle. This would be sensible if the user is defining a canal or other such simple engineered channel shape. Intermediate trapezoidal shapes with any side-slope are possible.

The asymmetrical U-shape shifts the deepest part of the cross section back and forth across the river to stay at the outside of the meander bend on the basis of how local channel curvature compares to the global maximum curvature. A challenging issue with the asymmetrical U-shape option is the decision as to what length of channel to use to compute a "local maximum curvature" that theoretically governs a channel's cross-sectional asymmetry. In traditional geomorphic theory and presentation of this concept, the idealized centerline takes on a simple sine function shape, so it is trivially obvious what channel length and curvature calculation method to use to have the deepest part of the cross-section at the outer bank. In that case, local and global maximum curvature are the same and there is no technical problem. However, more realistic centerlines have numerous undulations of varying amplitude, phase, and frequency, so it is extremely difficult to set an objective criterion for what length to use to compute a reference local maximum curvature. Further, River Builder allows for looped meandering, as discussed shortly, so there can be very high curvature, and that can happen at any scale depending on how many nested variability functions a user builds into the complexity of their meandering centerline. The procedure for how River Builder specifies centerline curvature is explained later; for now the important point is to understand that channel asymmetry is governed by that curvature. However, if you are dissatisfied with the longitudinal positioning of your asymmetric pools and riffles, then you may wish to use a lot of care in specifying the centerline function or switch to the symmetrical U shape option.

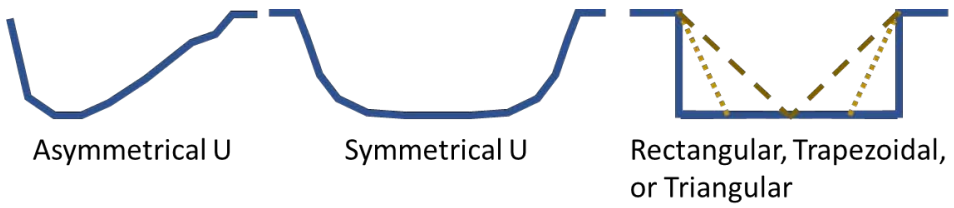


Figure 8. River Builder's inner channel cross-sectional shape options.

3.4 Planform Breaklines

Fourth, you formally select all the lateral slope breaks you want to use in your channel and valley zones (e.g., Figure 5). For each one, you must specify the horizontal and vertical offsets from the previous slope break close to the centerline. Left and right slope breaks are independent, so they do not need to have the same offset values. There is an enormous amount of freedom and capability possible with this aspect of the program. Negative vertical offsets are permitted, but negative horizontal offsets are not permitted. A negative horizontal offset is no different than specifying another contour closer to the centerline, so it does not make sense to allow that; if you want a contour closer to the centerline, then simply specify that before the next one out.

3.5 Geometric Variability Functions

Fifth, you turn the slope break points from your cross-sectional conceptualization into plan-view breaklines (aka contours), including specification of the geometric variability functions for each contour (Figure 11). Brown et al. (2014) only presented additive sinusoidal variability functions, but River Builder now has several more functions available. The functions include linear, sine, cosine, sine squared, cosine squared, Cnoidal, square wave (i.e. flat top and bottom with vertical risers), three-parameter Perlin noise, and gooseneck curve that loop back on themselves (Figure 9).

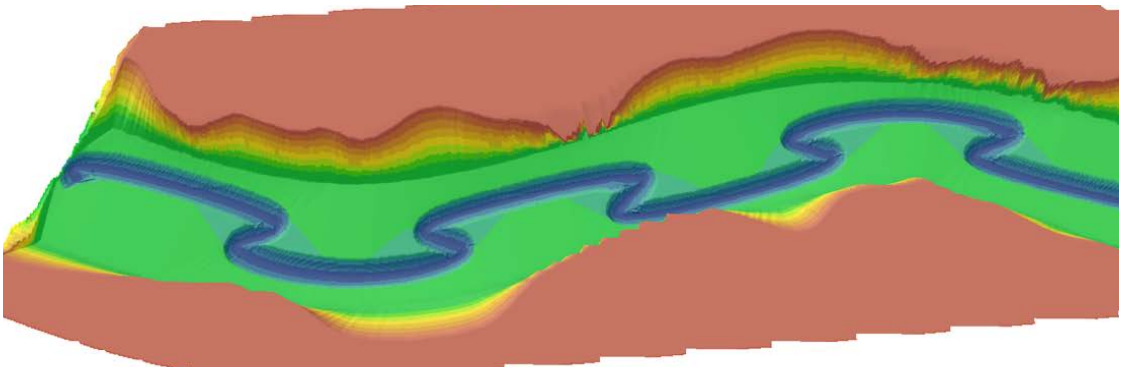


Figure 9. Example of a steep-sided sinuous river valley with an inset gooseneck-meandering channel. Obviously these are tighter looped curves than natural, but they certainly emphasize what is possible with the software.

The decision of which function or combination of functions to use for a given breakline rests on a firm geomorphic understanding of the local reach, its river, and the region. For each breakline you can add together as many individual functions as you want to define a more complex geometric variability function. For example, let's say a river is widening downstream but also oscillating its width. You can design that by adding a line function and a sine function. Or as another example,

let's say a real bed longitudinal profile includes many frequencies of undulation. You could perform a harmonic analysis to obtain the parameters for all the sinusoidal functions necessary to replicate the periodic components of the natural signal. Then, you could mimic the non-periodic residual by adding a Perlin noise function.

River Builder for Python is capable of changing geometric functions at user-designated positions down the valley. This is termed "piecewise varying". This is possible for any breakline. Conceptually, this is done by specifying the length of the river along which each geometric variability function applies. This step involves applying a "mask". How this is done will be explained later.

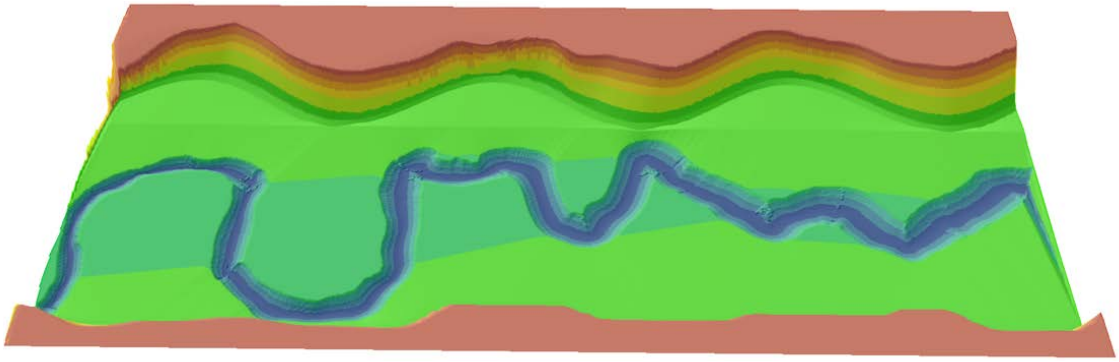


Figure 10. Example of a channel whose centerline function varies at user-specified positions down the valley (i.e. a piecewise varying centerline). From left, it begins with a gooseneck meander function and then switches to other types down valley.

Once you select all your functions, you have to set the parameter values for each one. All functions have parameters. For example, lines have slopes and y-intercepts. Sinusoidal functions have amplitudes, frequencies, and phases. You have to decide what values to use for all the parameters. Further, you can use the same function for multiple breaklines, but if you want to change the parameter values, then that essentially makes it a new instance of the function.

Although parameters for each equation may be specified independently, self-maintenance processes in rivers often produce coherent patterns among multiple geometric elements, including mutual longitudinal variations identified through spectral or wavelet analysis of spatial series or the GCS function between them. Thus, the key decision for parameterization is to decide which two or more variables will be linked to vary coherently. For that to happen the variables have to be described by the same equation, and then parameters governing frequency, amplitude, and phase are specified to yield oscillations that produce the desired landform structure. This is where GCSs come into play. Beyond creating landforms, the deeper goal is to obtain topographically steered, stage-dependent hydraulics that drive a variety of channel maintenance mechanisms when operated on by a natural flow regime. At the same time, there is another goal of providing spatially organized yet heterogeneous aquatic and riparian habitats to serve different species needs in their various life stages.

Remember, geometric functions are defined relative to the stream-wise and valley-wise coordinate systems. Therefore, in your mind's eye you have to think of them that way. It is possible and common that the nesting can get so complicated that contours will cross over. River Builder will erase any such crossovers and insure a smooth topography.

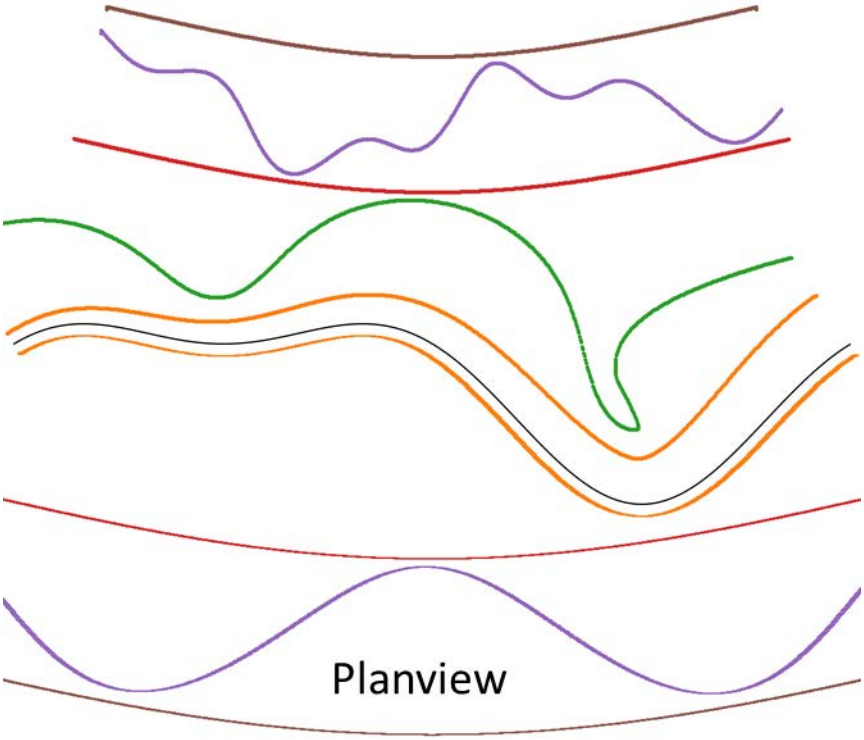


Figure 11. Example of the plan-view result of projecting slope break points into plan-view contours. Black line is centerline. Orange lines are inner channel banks. Green line is an additional slope break that could be viewed as in the channel or on the floodplain depending on the conceptualization a designer has. Red lines are the slope breaks from a macro channel to the valley floor or potentially from the floodplain to adjacent terraces. Purple lines are additional floodplain or terrace slope breaks. Brown lines are the outer-most valley boundaries of the synthetic river valley in this design.

3.6 Apply Functions To Breaklines

Because you may wish to apply the same geometric variability function to multiple channel and valley breaklines, the step to define those functions is independent of the step to apply them to one or more breaklines. As a result the sixth step is to select the functions you want to use for each breakline and write them out. River Builder enables you to add together as many functions as you want for each breakline. Just remember that you may need unique parameter values for different breaklines, so you may have to create a unique set of functions for each breakline if you want that to happen.

3.7 Run River Builder

Finally, you run River Builder to render the 3D terrain and output results. The most important output is a .csv point file that has the coordinates of all the points in the terrain. To help with iterative design and to check design characteristics against expectations, several other outputs are provided, including data tables and 2D plots. River Builder does not have a 3D plotting capability, so you have to import the final .csv point file into another program to visualize it in 3D. Based on

analysis of all outputs, you may choose to iterate the design to refine and improve it until you have the desired outcome. It is recommend you keep your original files for each attempt and use a master log file to document and track all your attempts. This will preserve your design process in case you want to go back later to review what you did or return to an earlier design instance.

4 River Builder 1.0.0 Highlights

As in the past, **no knowledge of coding is required** to use this software. The user interface is as simple as it can be (i.e. a text editing program of your choice and a simple text input file). To run the input file, you do have to have Python3 on your computer along with two packages: python3 numpy and matplotlib; all of these are free. Note that the two packages come with ArcPro, so if your Python3 installation came with ArcPro, then you are all set. Otherwise, just download those packages and you're ready to go. Also, this manual tells you the one and only command you have to run on your computer to make the program evaluate your input file, so that is very accessible for everybody to do.

In considering how to proceed with development of River Builder from the previously existing R version (0.1.1), several factors came into consideration. First, we re-evaluated the future of the R platform relative to the opportunities and challenges we see in the future. A key consideration was the fact that we have also developed a companion "River Architect" software platform to analyze both natural and designed river valleys.

Second, we came to the realization that we needed to revisit and further advance the equations in Brown et al. (2014) to open the possibilities for many of the new features in this version of River Builder. Science is an iterative endeavor. We have learned a lot through time, so it was now necessary to start over from scratch and implement a next-generation code.

Third, we needed to significantly improve the structure of the code to institute best practices in coding.

Based on these considerations, we came to the conclusion that River Builder would have the greatest potential for use on many devices and in many applications if programmed in Python instead of R. Thus, we chose to re-write River Builder from scratch and implement it in Python3. We chose Python3 over Python2 in light of the pending retirement of ArcMap that uses Python2. ArcPro uses Python3. At present River Builder does not use any "arcpy" functionality, but programming in Python3 preserved the option to do so in the future.

The overhaul of the code has led to several major new features and fixes, as summarized next.

4.1 Major New Features

- Different channel and valley centerlines, with channel centerline and associated coordinate system nested inside valley coordinate system (Figure 3; Figure 11).
- Channel now broken into independent left and right sizes, with overbank valley region still having independent left and right sides (Figure 5).
- Dedicated "inner channel" that is assigned a formal XS, using all the same shape options as

before, and they all work correctly now, including asymmetric U (Figure 5).

- Outside the inner channel, outer bank and valley zones each now have an unlimited number of XS slope breaks, with each break point being assigned a vertical and horizontal offset from the next adjacent one closer to the channel centerline (Figure 5).
- Each XS breakpoint may be assigned positive, zero, or negative offsets, which enables users to create levees, roads, and yazoo channels (however, only one channel gets the formal XS shape (Figure 5). All others would have to have their XS shape manually created with slope breaks).
- New geometric functions have been added. These may be used for any channel or valley feature that may use a variability function.

    - The Perlin function now has a higher order functionality with octaves to enable more fractal diversity.
    - The Cnoidal function is now available to allow for creating a periodic function with longer, flatter zones separated by more peaked zones as compared to the sinusoid function. Parameters control how long and flat the flat zones are.
    - The rectangular step function has been added to allow to have flat zones separated by a nearly perfectly straight line on the basis of a representation using many cosine functions
    - The $\sin^2$ and $\cos^2$ functions have been added. Line, sine, and cosine functions remain. The value of the squared functions lies in their ability to produce two positive peaks over the range for which the sine and cosine functions only produce one positive peak.

- Unlimited number of geometric functions may be added/subtracted together to make a single complex function with multiple features with different frequencies.
- A looped meandering function has been added for use with the centerline (Figure 9). This allows for multiple Y values for each X centerline position, which is what is needed to create "gooseneck" meandering.
- New 2D plots to help users evaluate renders, including independent plots for whole valley XS vs just channel XS.
- Initial reach-average values a user specifies only hold true before a user adds geometric function variability. These values are now minimums that variability is added on to. The code now computes the actual final reach-average values of channel slope, width, and depth, so a user can decide whether to go with those values or iterate the design. We also added an optimization routine to allow user to enforce final desired channel-wise reach-average metrics like width and depth by enabling program to change geometric function parameters as needed to end up with the right values.
- Optimization routine so that a user may specify the final desired values for some parameters and then adjust specified variability function parameters to guarantee those final outcomes.
- A mask functionality now applies to all functions. A user can specify at what ranges of distance along the channel or valley centerline a function should turn on, and what ranges of distance the function should turn off. This provides a greater flexibility for users to create a non-periodic curve (Figure 10).

4.2 Major Fixes

- The new code is natively, entirely in stream-wise coordinates, so everything that is done is in line with or perpendicular to the centerline now. This applies to both the channel and valley centerlines.
- The new code has a routine that prevents topographic contours specified with geometric functions from crossing over. When crossovers would occur, the code deletes crossed sections and smoothly interpolates a good surface to avoid any problems.
- The previous equation for asymmetric U cross-section was based on x-y coordinates and calculated global curvature in such a way that the function could break if attempted to generalize for all possible river shapes. In theory the code should use some sort of local curvature for complex centerline meandering. However, local curvature is tricky, because you have to decide what counts as local; if you choose too small a scale, every random fluctuation can become a crazy local maximum and channel asymmetry can go wild with too many undulations back and forth. The new code uses an advance algorithm to compute curvature at each point that is naively stream-wise coordinates. This new calculation is determined only by how the river bends, no longer affected by the position of the river. Specifically, it looks from one point to the next to create a vector between those points. Then it computes the angle between any two such adjacent vectors. The maximum angle sets the global maximum curvature. The degree of cross-section asymmetry is only at this maximum level when a meander bend is as sharp as the maximum curvature location. For bends with lower curvatures, the asymmetry is scaled linearly lower. As long as meandering is well-behaved, then the function will work well, but if extreme high-frequency noise is added along the meandering centerline, then it is possible that the maximum curvature value may not be very meaningful for controlling asymmetry at the smoothed meandering scale where it belongs. In such a situation, there is an advanced feature where the user may specify their own curvature function instead of using the internally computed one. This is explained in section 9.6.
- Addition of crash protection features so that if a user specified impossible values or no values, then the code defaults to set default value and the code still runs. The program outputs a warning message and shows all the values used, so a user may figure out where the problem went wrong, but they still have a working design to study.
- Deletion of all geomorphic covariance structure (GCS) computation for now, because it was being done in the straight valley coordinate system, not the stream-wise system. Will add this back later as a feature addition when we can program it properly.

4.3 Major Features Removed

- The Caamano criterion for two-stage flow convergence routing (aka "velocity reversal") was previously computed for fixed, assumed riffle/pool locations for a few example cases with a straight channel. Thus, the old outputs would be wrong for most cases. This feature has been removed until it can be re-programmed to compute for each sequential riffle-pool couplet.

5 SOFTWARE DOWNLOAD, INSTALLATION, AND SET-UP

5.1 Install Python or Locate Your Existing Copy

Use of River Builder does not require any programing skill, but you do have to install the free software, Python in version 3.x, where x is any version number within the 3 series. Python 3.x is available for free for the Windows, Mac, and Linux operating systems.

If you use ArcPro on Windows, then you already have Python 3.x on your computer and the two packages, python3 numpy and matplotlib. It is recommended that you use this Python program; do not re-install another version elsewhere on your computer. An example of where you might find this program in ArcPro could be something like:

C:\Program Files\ArcGIS\Pro\bin\Python\Scripts\propy.bat

If you do not use ArcPro, then simply go to the Python website and download Python 3.x to your computer using the link below. Follow its instructions to install it.

https://www.python.org/

5.2 Select Your Preferred Text File Editor

River Builder works by having you set up your design in a single text file. Thus, you need a text file editor (aka word processing program), which is available for free on all computer operating systems (e.g., Notepad on Windows and Textedit on MacOS) and from many cloud service providers. You can use any text editor you want- no special functionality is needed.

The input file for River builder uses the ".txt" file type. Be sure that when you finish preparing your text file in your preferred text editor that you save it back to this file type.

5.3 Download River Builder Python Code

River Builder 1.0.0 is publicly available for free as a download from the Github repository at the link below. The Github repository offers the essential Python package, many worked sample files, and any add-on tools we have programmed to help users develop special cases for which designing the variability functions can get very complicated.

https://github.com/RiverBuilder/RiverBuilder

In addition to the Github repository, there is a dedicated website that supports training and research with River Builder at the link below. Over time, training videos, more worked samples, workflows, and other aids that are not suitable for storage on Github will be available here.

https://riverbuilder.ucdavis.edu.

We will do our best to release updates to improve the existing software and build in new capacities over time. When we do, they will be available at these two sites. We do not have a newsletter or anything, as we are just a couple of academics, so you just have to check the websites

periodically.

6 RUNNING THE PROGRAM

We recommend you rename the "river builder" folder to the simpler name "RBpy" just to make it easier to type commands with a shorter name.

Place your text input file into this RBpy folder. It can have whatever name you want to call it. A good nomenclature to use is to add a numerical suffix to the name so that as you iterate the design you casn just increment that number. For the purpose of this manual, we will use the file name "NAME_001.txt" as the example. NAME would be replaced with your preferred design name and then as you try things you would increment form 001 to 002, 003, etc.

You can have as many input files as you want in the folder.

6.1 Windows OS Simple Approach

You can run River Builder with a single instruction that is easy to write and implement. This command is implemented from within the RBpy folder. The command has the following structure:

"A" –m riverbuilder B.txt C "D"

where "A" represents the file location for your Python 3.x executable file, riverbuilder is the River Builder Python package, B.txt represents your text input file name, C represents the name of the new folder within RBpy where you want the program to place your results, and "D" is a simple text comment you can add to the log file, if you want to. C and D is not required. If C is not specified, the output folder will be "riverbuilder_output". Please put the comment D in quotation marks.

An example of such a single command is provided below:

"C:\Program Files\ArcGIS\Pro\bin\Python\Scripts\propy" –m riverbuilder NAME_001.txt NAME_001output "a simple test."

To implement such a command, follow these steps:

Open a Command prompt window by clicking on the search bar and typing "cmd".

In the Command prompt window, navigate through the file directory and into the river builder folder, now called "RBpy".

In the Command prompt window, type your single command to run the program, such as:

"C:\Program Files\ArcGIS\Pro\bin\Python\Scripts\propy" –m riverbuilder NAME_001.txt NAME_001output "test1"

That's it. The code will run and your results will be produced.

6.2 Windows OS Python Integration

To enhance the ability for users to integrate River Builder with other python tools, River Builder can also be used as a regular python package.

Navigate to folder containing the riverbuilder folder andcreate a python script file, for example: "test.py".

Open test.py and write down following code lines:

```
from riverbuilder import river

fname = "INPUT FILE NAME"
outfolder = "FOLDER WANTED TO SAVE OUTPUTS"
log = 'MESSAGE WANTED TO ADD TO LOG'  # or ''
river.buildRiver(fname, outfolder, log)
```

If this test.py file is run, the code will run and results will be produced

7 HOW TO USE RIVER BUILDER

The essence of using River Builder comes down to simply entering your design values into the input text file and then running the one command in the previous section. Its that simple… and yet it is also quite challenging, because you have to know what values you want to put into the input file. The software does not tell you how to specify a river design, but by offering you a set of design options, it certainly focuses your effort on the key components.

The default input text file has pre-defined, acceptable minimum values for the program to function correctly. For full assurance, the user must follow the guidelines specified at the top of the text file.

8 PROGRAM OUTPUTS

As soon as River Builder stops running and the Command prompt window returns to its default ready status, you will have a new folder within RBpy having the name you assigned in your command to run the program. In that folder there are a number of files, which are described next.

Log.txt
Contains a listing of all the parameters and functions used to run the model based on your input file. It also includes "Alert!" messages that tell you how the program handled any missing information, usually by assigning a default value or turning a feature off.

In many instances the "Alert!" message is not an indicator of a problem of any kind; it's just there to encourage you to affirm that the decisions you made were what you intended. However, always read

the messages in case there is something you meant to specify and did not.

SRVcenterline.csv
A comma-limited text file containing the bed slope and planform curvature along the centerline. This can be used for using and plotting these profile data in another program.

SRVcurvature.png
Displays plots of the bed slope and planform curvature series from the SRVcenterline.csv file.

SRVinnerChannelXShape.png
A plot of the cross-section of the inner channel. There will be two plots if an asymmetrical cross-section is used, with one at the curvature closest to zero and one at maximum curvature. If symmetric U or trapezoid, or if no curvature changes, there will be only one plot. The legend shows the X position(s) of the cross-section(s)

SRVlevels_xy.csv
A comma-limited text file containing the planform points for all the specified contours

SRVlevels_xy.png
Planform plot of the river showing the positioning of the river's features.

SRVlevels_xz.csv
A comma-limited text file containing the elevation longitudinal profiles of all the contours, including the thalweg.

SRVlevels_xz.png
Longitudinal plot of the river showing the elevation profiles for all specified contours

SRVmetrics.txt
This file contains statistical analysis results from analyzing the river after it has been rendered with the variability functions. Computed values can be used to iterate the design as needed to get what you want it to be given a complex subreach meandering function. River Builder does have an optimization routine to help with that, but sometimes you may want to adjust different parameters than the ones selected for optimization. The information can be organized into two categories-channel and valley data.

- Slope & Sinuosity: The metrics file tells you the actual channel and valley slopes and sinuosities (along the centerline). Recall that the user enters a straight valley slope in the input file, not the final sinuous valley slope, not nested sinuous channel slope. Only for a straight channel will the valley and channel slopes be the same.
- Average Depth and Width statistics: mean cross-sectional depth (H) and top width (W) values are computed along the channel, perpendicular to the channel. Also, the coefficient of variation of each is computed for the whole reach.

SRVtopo.csv
This is the primary output that is the purpose for River Builder. This file contains all of the comma-delimited XYZ coordinates for the synthetic river valley. Points are provides along the user-specified

contours that define the river corridor as well as along a user-selected number of intervals in the inner channel. The first row of the file contains the header, {X, Y, Z, Label}. Label is a 2-letter designation for what contour a given set of points represents.

SRVvalleyXSshape.png
A cross sectional plot that spans the entire valley width.

9 RIVER BUILDER INPUTS

River Builder works by having the user write all of the information to create an artificial river corridor in a single text file. It cannot get any simpler than this, which is why we chose this approach. The text file is written in plain English that is easily readable, but for the specific input lines you do have to follow the required syntax.

Here are some pointers about parameters and inputs:

- Every line starting with '#' will be ignored.
- All dimensional numbers are in units of meters.
- User-defined functions may be used for sub-reach variability parameters only.
- Every input with an "=" sign should not have any spaces before or after the "=" sign.
    - For example, a line should be "Length=420" not "Length = 420".
- Bankfull depth can either be (A) user-defined or (B) calculated from the Critical Shields Stress and Median Sediment Size.
- Centerline Curvature can either be (A) user-defined or (B) calculate from centerline slope.
- Calculations of banks of channel are based on channel centerline; calculations of levels of valley are based on valley centerline.
- When providing an input involving pi, do so in the form of A*pi, where A is a constant. (EX: 2*pi, pi, pi/6, 3+pi)

9.1 Domain Parameters

- **Datum** – a fixed starting point of operation to measure changes of a specific property. Used for thalweg elevation. Input must be a real number. This datum is located at the downstream end of the synthetic river valley, and then the river goes up from there. Thus, if the datum is set as any number >0, then it is guaranteed that all elevations in the valley will also be > 0. You may set any arbitrary number you wish. If you want to estimate the total elevation range of the thalweg along the river, then simply multiply your Valley Slope input by the Length input and that will give you the overall elevation range of the thalweg. For example, for a 1000 m long channel and a Valley slope of 0.01, then thalweg elevation range is going to span 10 m.

- **Length** – the length of the x-axis of the river valley. Input must be a positive real number. This is only the channel length if the channel and valley are both straight. If the channel is sinuous and valley straight, then this length is the valley length and the actual channel length will depend on the sinuosity. Channel and valley sinuosity are computed by the model and reported in the Data.csv file, so you can compute the final channel length if you want to

know it.

- **X Resolution** – the resolution of equally spaced data points spaced along the X-axis in units of meters. **For now, we request the user keep this at a value of 1.** A value of 1 means 1-m resolution. Input must be a positive number. There is no limit to how fine the point spacing can be, other than computation time for your computer. The smaller this number, the longer the code will take to run. Use your judgement to decide how much detail you need given the scale of your river and the resolution of the sub-reach variability functions you are applying. There is no need for extra detail if the river is easily interpolated with less data.

9.2 Dimensionless Parameters

- **Valley Slope (Sv)** – the slope of the valley floor along the X-Z plane assuming a straight valley. It is used to calculate the overall channel slope with a given sinuosity and govern the incline/decline of the surrounding valley. Input must be a real number. Note that during the process of designing a channel it is often helpful to set this to zero, so you can visualize the "detrended" geometry of the river. Once satisfied, then add the desired slope.

- **Critical Shields stress (t*50)** – This is an optional input. To use it, set the inner channel depth minimum to 0 and then this will be activated. If you want to use the classic equation to establish a bankfull depth to just yield the shear stress needed to mobilize the bed for a specified bed material grain size, then you can use this by making the inner channel take on bankfull dimensions. Common values used for this parameter are 0.03, 0.045, 0.047, or 0.06, depending on what size fraction you want to insure would be mobilized at bankfull flow. The equation that uses this input is Eq. (4) in Brown et al. (2015). Input must be a positive real number. ***This is the one place where the SI unit system matters in River Builder. If you do not use this function, then there really are no units to your numbers for all practical considerations.***

9.3 Channel Parameters

The parameters in this group specify initial values for the width and depth of the inner channel (Figure 5). Because width and depth can be made to vary down the river, it is impossible to pre-specify final average values. A choice has to be made as to how to vary these relative to the start values. In this software, the decision was made to have the user set the **minimum** values for width and depth at this stage. If no variability functions are used, then these are the final values. If variability functions are used, then the software computes the average values along the whole channel and reports that to the user. The user can then decide to manually adjust the initial values or the variability functions to get a different outcome, or they can use the software's optimization routine to force desired average values at the expense of changing variability function parameters. This functionality will be explained later.

- **Inner Channel Width Offset Minimum** – the minimum distance between left and right inner channel bank tops on the X-Y plane, not counting any sub-reach variability functions. Input must be a positive real number. This variable is specified to protect the user from making a mistake with sub-reach variability functions in which the banks would cross over,

yielding an impossible outcome. This insures that the river never has a zero or negative width. If you are confident in what you are doing, then you can set this to zero.

- **Inner Channel Depth Minimum** – the height offset between the bank top and the highest point of thalweg on the X-Z plane. This prevents sub-reach variability functions from making the depth any smaller than this. Depending on channel slope and variability functions, the local inner channel depth can differ from this value. Input must be a positive real number or zero. If zero, then the code calculates an inner channel depth minimum using the critical Shields stress equation in section 9.2. This is intended for creating an inner channel that is the bankfull channel.

- **Median Sediment Size (D50)** – the particle size of the material comprising the river bed. Input must be a positive real number. This is only used if the user has set the inner channel depth minimum to 0 to compute the bankfull depth value associated with the initialization of bed material transport for this specified grain size. This value is used in conjunction with the critical Shields stress value as explained in section 9.2. Otherwise it has no utility in River Builder. Remember, River Builder creates topography, not sediment facies. Sediment facies are too fine scale for use in the current intended applications for River Builder.

9.4 Cross Sectional Shape

This group of parameters controls the shape of the inner channel's cross section on the Y-Z plane.

- **Channel XS Points** – the number of equally spaced points defining the inner channel's cross section. Think of this as the resolution of the cross-sectional shape. This is not applied outside the inner channel, only in it. Beyond the inner channel, River Builder provides you with points along the major breaklines you create, and then you can interpolate the terrain between those on your own later using a TIN approach. Input must be a positive integer greater than 1. For field data collection in a U-shaped channel, people commonly use ~ 15-30 points. An odd number is wise if you want to have a line of output points exactly in the center of the channel.

- **Cross-Sectional Shape** – This is where the user decides whether to have an asymmetrical (AU), a symmetrical U shape (SU), or a symmetrical trapezoid (EN) cross-section.
    - AU sets an asymmetrical cross-section. Once this choice is made, there is no further control over how the asymmetry works. That is dictated by the equations in river builder.
    - SU sets a symmetrical U shape cross-section.
    - EN sets a symmetrical trapezoid cross-section, and then the user further controls the shape of that using the TZ(n) input next. These options are triangular, trapezoidal, or rectangular.
- **TZ(n)** – number of break points placed on the riverbed at the maximum depth across the inner channel's cross-section, not counting the left and right bank tops. For reference, see the far right cross-sectional diagram in Figure 8. This is only required when one uses SU. If these values are present when Cross-Sectional Shape is set to AU, then these values are ignore and have no effect. In this function, *n* is the number of edges that defines the length

of the trapezoidal base such that $0 \leq n \leq$ (Y-Z Increments).

- When TZ(n)=1, there is only one breakpoint at the centerline, so that yields a triangular cross-sectional shape
- When TZ(n)= **Channel XS Points**, then all of the breakpoints are along the bed right up to the banks, so that means the shape of the cross-section is rectangular
- When TZ(n)= any other number between 1 and Channel XS Points, then the shape is going to be trapezoidal The higher the number, the steeper the banks.

9.5 User-Defined Functions

This group of parameters consists of the mathematical functions a user may then apply to any feature (i.e., breakline) later on. It is common that a user might want to re-use the same function to control multiple features, so that is why these are specified independently from the river features.

Take note that if you do NOT want to have any sub-reach variability, then you can leave all of these inputs blank and the code will move forward using only straight lines for every breakline you set up.

By far, this is the most challenging part of river design, because River Builder does not tell you what functions with what parameter values to use- you have to decide that before getting started on the basis of your expertise as a river scientists and/or engineer. Further, it is especially difficult, because as of today very little of river science has looked into the structured patterning of sub-reach variability functions, except for riverbed undulation. Thanks to the common practice of mapping thalweg profiles, riverbed undulations are relatively well described in the literature. However, how those undulations relate to other sub-reach variability functions (i.e., their GCSs) has hardly been investigated yet. Further, there is relatively little about width variability and almost nothing about sub-reach variability functions for other breaklines outside the inner channel. If anything, this software serves as a call for more of such research to be done. However, as a user, you have to make do with what you can achieve right now and try your best to specify the sub-reach variability functions you think need to be there or which match real observations from reference reaches.

For each type of function, you first specify a text identifier for that type of function and then you specify a sequential numbering after that for each additional function of that type that you add. For example, the first sine function (SIN1) could have an amplitude of 2 and the second sine function (SIN2) could have an amplitude of 4. There is no limit to how many of each type you have, but each of the same type should have a different number. We recommend that users number functions upward sequentially, but there is no limitation to how you order and/or group numbering as long as each function has its own number. There is also no limit to how many different types you create, up to all the types that exist in the software at this time.

Each function has some number of parameters to make it work and the user must specify the values for all required parameters for a given function.

9.5.1 Types of functions

At present there are 9 different kinds of functions available for you to use to create organized sub-reach variability.

9.5.1.1 Periodic functions

Seven types of functions (Sine, Cosine, Sine^2, Cosine^2, Cnoidal, Rectangular Step, and Looped) provide periodic oscillations. As simple as those are individually, remember that you can add as many of them as you want together into one overall function using addition. Subtraction can be easily achieved by using a negative amplitude. As shown in Brown et al. (2014), it is possible to get some very interesting and meaningful variability patterns with 2-4 SIN() functions added together, using different values for frequency, amplitude, and phase. One way to ascertain what those parameter value could be would be to perform harmonic analysis on real series of river variables, such as bed elevation, width, meander centerline, etc. You can extract all the periodic functions you want from that kind of analysis and use those in your design to whatever degree you wish.

- The sine and cosine functions are widely known and have been used to conceptualize river designs for generations. Itis important to understand that while it is easy to conceptualize a river with these functions, few rivers actually vary exactly according to their simple geometry. By using multiple frequencies of these functions, you can get a better representation of reality, but River Builder does not limit you to that. In any case, these functions are a good starting place to consider.
- The sin^2 and cos^2 functions are valuable because they produce two positive peaks over the range for which the sine and cosine functions only produce one positive peak. That could be used to enforce that something repeats for each location of max/min position along a sine wave, not only at the maximum or only at the minimum.

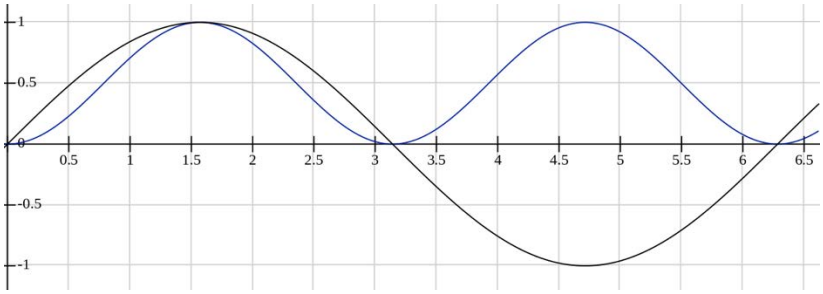


Figure 12. Sine^2 function (blue) compared with sine function.

- The Cnoidal function allows for creating a periodic function with longer, flatter zones separated by more peaked zones as compared to the sinusoid function. Parameters control how long and flat the flat zones are. When taken to the extreme and applied to the inner channel, the Cnoidal function can be used to make sharp-crested weirs for the longitudinal bed profile and/or plan view jetties.

Figure 13. Cnoidal function example.

- The rectangular step function allows for creation of flat "treads" separated by perfectly straight "risers". In a thalweg profile this could be used to have a broad-crested weir. In a plan view breakline it could be used to make a bedrock outcrop or man-made bank feature, such an artificial peninsula.

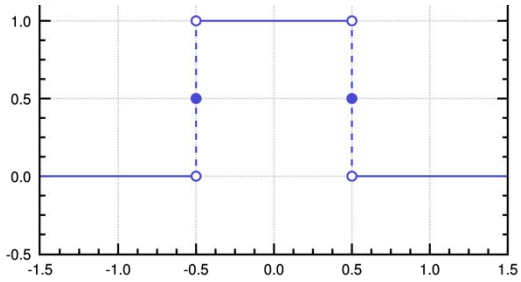


Figure 14. Rectangular step function example.

9.5.1.2 Line function

Because rivers often do exhibit linear trends in variables, such as width and depth, River Builder enables you to apply the line function for use to create that effect.

9.5.1.3 Non-periodic, deterministic oscillations

Sometimes you want to have a function that is deterministic and oscillating, but non-periodic. One application for this type of function would be for the meander centerline, because channel alignment is not going to be truly periodic. To achieve this, River Builder includes the three-parameter Perlin noise function. To find out what sinuosity you get from a particular set of Perlin parameters, you have to run River Builder and look at the SRVmetrics.txt file. That's a bit clunky, but it works. Alternately, you can use an online Perlin noise function calculator to fine-tune what you want and then simply copy the parameters form that calculator into your River Builder input file.

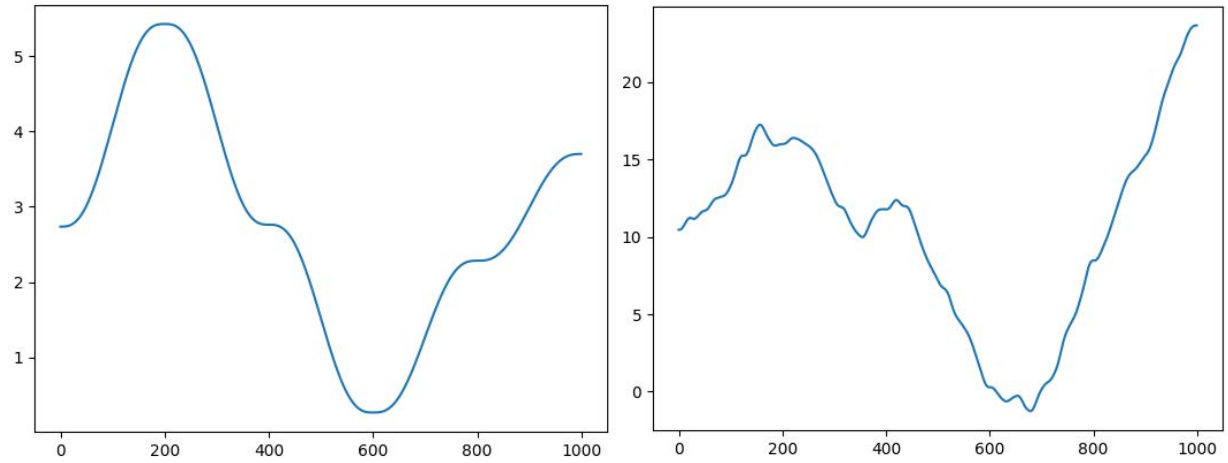


Figure 15. Perlin function examples. Both use the same amplitude and wavelength, but the one on the left only has 1 octave while the one on the right has 4 octaves. More octaves

means the function has more rugosity.

9.5.1.4 Piecewise functionality

River Builder is generally intended to serve as a reach-scale river valley design tool. As a result, if you want to create a longer river segment composed of multiple reaches, then it makes the most sense to create a unique design for each reach. However there are a few time when breaking a long channel into multiple independent files just does not make sense. The easiest example of this for a natural river is a step-pool channel type. Nobody would want to make one reach design for each section between drops and then one reach design for the riser of each step. What if you wanted to make a river design of Niagara Falls. The second example of where this comes into play involves practical river design wherein there exist non-adjustable fixed points that serve as lateral turning points for the centerline or as vertical change points, like when one might want to daylight a stream at a certain elevation. After thoroughly studying the math of continuous functions with discontinuities, we decided that the best solution to the problem of discontinuities was to add a piecewise functionality to River Builder (Figure 10).

The way this work is that every geometric variability function has a fourth parameter called "MASK". MASK is an entire function until itself with its own novel parameterization. Just like with the variability functions, you create a numbered MASK function in sequential order, MASK1, MASK2, MASK3, etc. By default, every input file must include MASK0, which states that the geometric variability function applies to the whole length of the river. Consequently, by default every function can begin with MASK0 as its fourth parameter.

MASK functions include just two parameters, but they can repeat many times in sequence in the function. The first parameter is a position along the centerline. The second parameter states whether the MASK is on or off beginning at the position in the first parameter. After that, the user writes each change position and whether the mask is turning on or off for the length of the river.

For example, imagine a 1-km reach in which the first 250-m section was engineered straight and then after that it meandered. There is no need to specify on/off at position 0. You would create a line function with a mask that turns it off at position 250. The nomenclature for this would me "MASK#=(250,off)", where # is the MASK function number you assign to this. You would also create a sine function with a mask that is off at 0 and then turns on at 250. The nomenclature for this would me "MASK#=(250,on)", where # is the MASK function number you assign to this. For a single change/turning point, it is that simple.

Note that each section of a river can have as many geometric variability functions as you want to specify for it, and you can simplify the process by re-using the same MASK function for all the geometric variability functions that will be turned on and off at the same positions.

The problem with this nomenclature is that if you want to make a complex river, then it can get quite involved to specify all the mask functions. It is possible, but time-consuming.

For now, this is how River Builder works.

A future feature that is under development is a set of "function generators" that have a much-simpler way for you to specify typical types of piecewise channels. For example, there is going to be

a step function generator that will allow you to create sequences of step-pool units very easily. This tool will be a separate Python3 script that will take your inputs and produce the exact geometric variability and MASK functions you need for River Builder to get what you want, and then you paste them into your River Builder input file. For now, this separate tool is not available, but keep your eye on the main branch of the River Builder Github for when it arrives.

9.5.2 Common function parameters

Many functions tend to use the same types of parameters, even if they mean different things for different functions. All parameter values are in absolute units, but not referenced to any specific unit system, so it does not matter if you are working in SI, American customary units, or anything else. How a function parameter works with its absolute units depends on which channel breakline it is being applied to , and that is explained in detail in the list of parameters next. Special for periodic functions, the whole reach is treated as one period cycle, which is 2 π. This even holds for $\sin^2$ and $\cos^2$ functions, whose period is π. Different period cycles can be obtained by manipulating the frequency parameter. If this becomes confusing, the solution to learn how it works for yourself is to play with a simple design using a single geometric variability function applied to one channel breakline in an input file to see how it works.

Here is a list of the common parameters

- **Amplitude (a)** – The range that the function's magnitude will span in absolute units. The amplitude is defined in an absolute scale for the variable the function is being used for. How it works will depend on the native range of the function it is being used in. For example, if used for thalweg elevation, then a value of 1 changes the elevation by 1, regardless of the function. However, as another example, if you want to have a 50-m undulation using specifically the sine function, then you would set the amplitude to 25, remembering that the sine function goes from -1 to 1. If it was the $\sin^2$ function, then that only goes from 0 to 1, so a with undulation of 50 would require an amplitude of 50. The amplitude applies to all functions that call for it as a parameter, but you have to be mindful of the native range of the function you are using.

- **Frequency (f)** – How many times the function will repeat itself over a length scale. In this software, the length scale depends on what this function is being used for. A design begins with a valley length and centerline, so the scale of that centerline is the length of the valley. After that, if the centerline has a variability function, then anything dependent on that function will be referenced to the length of the centerline. For example, undulations in inner channel width will go along the meandering centerline scaled by the length of the meandering centerline. Inner and outer bank breaklines will be scaled by the channel centerline's length, while valley level breaklines will be scaled by the valley's centerline. No matter the length of the centerline a breakline's undulations is based counting on, the number of periods will always equal the number of frequency

- **Phase shift (ps)** – The position along the function where the function will begin. This is referenced relative to the start of the periodic cycle.

- **Wavelength (w)** – Some functions use wavelength instead of frequency. This is defined as

the distance over which the function's shape repeats.

- **Octave** – The number of times that the Perlin function is iterated at smaller spatial scales. The higher this number, the greater the rugosity a function will have (Figure 15).

- **Slope (slope)** – the rise or fall per unit length. Positive value is oriented such that the river flows downstream and downslope. Positive stands for increasing the distance along meandering stream from upstream to downstream.

- **Y-intercept (y-intercept)** – The offset of a linear function when X equals zero.

- **Elliptic parameter** (**m**) – used in Cnoidal and Step functions. m should have a value between 0 to 1. When m = 0, it becomes cosine function; when m ~ 1, it has sharp peaks or steps.

- **Sinuosity parameter** (**p**) – used in high curvature function. It should be a positive number. Typically, with p > 2, functions show in high curvature feature.

- **Mask position** (**DIST**) – position along the centerline governing the breakline of interest where a mask is turned on or off.

- **Mask on/off** – a text identifier indicating whether a mask is turning "on" or "off" at the specified centerline position.

9.5.3 Available Variability Functions

For each function listed below, a brief definition is provided. Indented bullets show the exact notation to be used in the user-defined functions section of the input file. The parameter symbols match those in the previous section.

- **MASK** function. A special kind of function, MASK function, is integrated with other functions as a parameter. A recognizable mask function should have the following format:

  MASK#=(DIST1, on/off, DIST2, on/off, DIST3, on/off, ...)

  - DIST means starting from what distance we want to turn on/turn off the function.
  - DIST should always increases.
  - DIST should not exceed the maximum distance of the river. (Exceeding parts will be ignored)
  - DIST should always be integers.
  - Every DIST should pair with an on/off switch.
  - A default MASK function MUST always be defined: ***MASK0=(ALL)***. It means the function will always be turned on for the whole length of the river.

  An example mask with a 200-long reach could be:

  MASK1=(0, off, 30, on, 75, off, 135, on) *long-form version
  MASK1=(30, on, 75, off, 135, on) *short-form version

(both of these yield the exact same outcome, as by default River Builder understands that if you turn on a mask at a certain point then it must have been off before that)

- Repeated for clarity and emphasis: ***MASK0=(ALL)*** is always required in your input file.

- **Sine** and **cosine** function. Standard trigonometry as you expect. Requires specification of amplitude, frequency, and phase
  - SIN#=(a, f, ps, MASK)
  - COS#=(a, f, ps, MASK)

- **Sine^2** and **cosine^2** function. Standard trigonometry as you expect. Requires specification of amplitude, frequency, and phase
  - SINSQ#=(a, f, ps, MASK)
  - COSSQ#=(a, f, ps, MASK)

- **Linear** function. Standard line. Note that the riverbed already has a built in slope function, so you do not have to add this to the thalweg to have a sloped riverbed. Could be used to create width expansions or constrictions, as one example.
  - LINE#=(slope, y-intercept, MASK)

- **Perlin** function – A smoothed, interpolated noise function that resembles a natural fractal curve. The function first generates a series of random numbers and then fits a curve to the set. The amplitude parameter controls the magnitude range of the random points, while the wavelength parameter controls the spacing between points. Next, the process is repeated for incrementally changed values of amplitude and frequency to obtain a set of curves. Finally, the curves are summed to yield a single function with many scales of fluctuations, similar to a fractal function. The persistence parameter controls how amplitude and wavelength change for a given set of curves to obtain different patterns of complex curvature. This function is especially suitability for meandering the centerline to avoid a pure sinusoidal shape.
  - PERLIN#=(a, w, octave, MASK)

- **Cnoidal** function – Periodic wave with one side flatter and one side sharper. If this was used for the riverbed, then the pools would be longer and flatter, while the riffles would be shorter and more peaked- or vice versa. See. Figure 10. If m=0 it will be a cosine function.
  - CNOIDAL#=(a, f, ps, m, MASK)

- **Rectangular** step function – Allows for creation of flat "treads" separated by perfectly straight "risers". If m=0 it will be a cosine function.
  - STEP#=(a, f, ps, m, MASK)

- **Looped** function – Gooseneck meandering.
  - HIGHCURV#=(a, f, ps, p, MASK)

9.5.4 Example List of Functions

One you select which functions you want to use and specify the parameter values for each instance of each function, then the next step is to write out your list of functions.

The list goes in the section of the input file names "user-defined functions".

A list of functions might be something like this:

MASK0=(ALL)
MASK1=(0, off, 500, on)

SIN1=(1, 4, 0, MASK0)
SIN2=(0.25, 5, 0, MASK0)
COS1=(0.5, 4, 0, MASK0)
COS2=(0.5, 2, 0, MASK0)

SIN3=(0.3, 4, 0, MASK0)
SIN4=(0.1, 2, 0, MASK0)
COS3=(0.2, 4, 0, MASK0)
COS4=(0.1, 2, 0, MASK0)

SIN5=(175, 0.5, 0, MASK0)
SIN6=(50, 2, 0, MASK0)
COS5=(75, 4, pi, MASK0)
COS6=(25, 10, 0, MASK0)

SIN7=(100, 0.2, 0, MASK0)
SIN8=(50, 1, 0, MASK0)
COS7=(75, 2, pi, MASK0)
COS8=(10, 7, 0, MASK0)

SIN9=(100, 0.2, 0, MASK0)
SIN10=(50, 1, pi, MASK0)
COS9=(75, 2, 0, MASK0)
COS10=(10, 7, 0, MASK0)

SIN11=(150, 1, 0, MASK0)

Notice that in this example list (taken from a variant of our "S10" archetype) there are four sets of pairs of sine and cosine functions used, plus an additional sine function at the end. This is done to organize subsets that will go together for sets of river breaklines that will share the same sub-reach variability functions. It is not required to do it this way. You can list them in any order you want. Whatever works for you.

9.6 Channel Sub-Reach Variability Parameters

This is where you get down to business with specifying attributes of each channel breakline in your design.

Take note that if you want a design with no sub-reach variability and no breaklines outside the inner channel, then you can leave this whole section blank and you will get a straight vanilla channel,

such as shown in Figure 1b.

Note: **Do not modify any of the names of the parameters in the input text files** (doing so will result in failure to use the functions or values correctly).

9.6.1 Channel breaklines

This section defines the geomorphic aspects of the river that variability functions may be used to control. The channel features are broken into two groups, those that describe the inner channel and those that describe the outer channel.

9.6.1.1 Inner channel properties

There are five optional river features that control sub-reach variability of the inner channel in plan view. Each of these features can have a sub-reach variability function.

- **Meandering Centerline** – governs the shape of the river's tortuous flow path (i.e. meandering or sinuosity) on the X-Y plane. Note that the thalweg does not necessary occur on the centerline.
    - Meandering Centerline Function=

- **Centerline Curvature** – Normally the sub-reach variability function for centerline curvature would be set equal to the computed curvature function governing the meandering centerline. However, one application of the curvature value is with controlling the asymmetry of a channel's cross-sectional shape. To provide greater flexibility with how a river's XS shape changes along the river, we have created a decoupled centerline curvature variable as an option. Normally one would not use this, but for an advanced user it might be considered. This variable has no effect on the channel's meandering. It only affects how channel cross-sectional asymmetry varies down the river along the meandering. For example, let's say that you did not want the deepest part of the cross-section to be at the outer bend of a meander, then you could use this to put the deepest part anywhere you want governed by the available variability functions. That is what this is for. This is a highly complex tool to control compared to others, so only use it if you really know what you are doing with your river design concept.
    - Centerline Curvature=

- **Thalweg Elevation** – governs downstream undulations in the bed elevation along the thalweg on the X-Z plane. May be used to create simple riffle-pool couplets and/or more complex sets of morphological units.
    - Thalweg Elevation Function=

- **Inner Bank** – The next river feature is actually a pair of breaklines used to specify inner channel width variability- **left inner bank** and **right inner bank**. Left Inner Bank governs the downstream variability in the river-left bank top breakline on the X-Y plane. There is only one left inner bank. Right Inner Bank governs the downstream variability in the river-left bank top breakline on the X-Y plane. There is only one right inner bank. To provide maximum flexibility, the user specifies each bank independently with as many user-defined functions as desired, but of course the two banks can be assigned the same function(s) if you

want them to behave the same way and have a symmetrical outcome. Together, the left and right inner bank functions define inner channel width. Recall that the "Inner Channel Width Offset Minimum" parameter will keep the two inner bank functions from crossing over as long as it is not set to zero. Note that the two inner bank inputs both go under this one heading, so you may want to separate them with an empty line but that is optional.

- Left Inner Bank Function=
- Right Inner Bank Function=

Ideally, one might expect that for maximum freedom between the two banks, the user ought to be able to not only specify different plan view sub-reach variability functions, but also different bank top elevations. The software has the capability with a workaround, but not using the inner channel specifications. The reason why it should not be attempted in coding the inner channel, is because we also must have the river's cross-section specified by a function and it would get extremely more complex if that function had to be cut in half to yield different bank tops a the outsides while still meeting exactly together at the middle. That's too much unnecessary coding to figure out compared to the solution we have implemented instead.

The simpler solution is to lock the inner channel left and right bank top elevations to a single value (governed by the specified the Inner Channel Depth Minimum parameter) and then let the user specify additional "outer bank" breaklines, each with their own vertical and lateral offsets to create whatever asymmetric banks is desired. This will become cleared after explaining the next two inner channel properties.

9.6.1.2 Outer channel properties

The outer channel is the zone between the bank top and the inner edge of the valley zone (Figure 5). There are two of these zones- left and right. "Outer banks" are individual breaklines that are within an outer channel zone. They may be used to represent asymmetric bank heights for the inner channel (by simply creating another bank breakline close to the inner bank but at whatever additional positive height offset is desired) and/or represent additional lateral slope breaks and channel features for a multi-level nested channel system. In dry climates it is common to have lower flow channels carved inside higher flow channels. Australian "macro-channels" with multiple nested channels within channels are the extreme of this in reality.

River Builder allows you to have as many outer bank breaklines as you wish, and there are many novel ways to apply them to get different outcomes. The simplest examples is to have nested channels within channels. However, mindful use of multiple positive and negative height offsets combined with high-amplitude sub-reach variability functions could yield braided, anastomosing, an yazoo channel scenarios. There can only be one main channel with thalweg undulation at this time, but conceptually having multiple threads or complex braided threads is plausible, though we have not attempted to create these archetypes for ourselves as of yet.

Specifying outer banks work a bit differently from the inner banks in two ways. First, there is an infinite number permitted, so it is necessary to have a numbering system. This handled by using a one-letter designator (L or R) for which side of the river one is addressing followed by a sequential number. For example, for 3 breaklines on the left side one would use the designators L1, L2, and L3. The designators for left and ride sizes are independent and do not have to match in height

offset, so L3 and R3 can be at totally different elevations (or there can be an R3 with no L3 at all).

Second, the lateral and vertical positioning of these breaklines must be specified. For the inner banks, lateral and vertical positioning were handled by the width and depth values specified in the Channel Parameters section. For each outer bank breakline, you have to specify its offsets from the preceding breakline. An offset is the incremental distance from the previous breakline to the new one. If this is the first outer bank, then its offset is relative to the inner bank on that size. The notation for the lateral offset is "Outer Bank Offset Minimum". For example, "L1 Outer Bank Offset Minimum=2". The lateral offset has to be a minimum, because one can apply a sub-reach variability function to it. The notation for the height offset is "Outer Bank Height". For example, "L1 Outer Bank Height=10". There are no vertical sub-reach variability functions in River Builder for anything other than the thalweg.

REMEMBER, offsets are NOT absolute lateral positions or elevations, they are incremental distances from the next closest feature towards channel center! Thinking in offsets is much easier than thinking in absolute positions. If you want to know absolutely positions, simply sum up all the offsets closer to channel center plus the new offset for a given feature and then you'll know where that feature is positioned in absolute units.

Finally, outer bank breaklines can be assigned sub-reach variability functions using the same approach as for inner channel features, as previously discussed.

- **Left Outer Bank** – Breakline in the left outer channel zone.
    - L1 Outer Bank Function=
    - L1 Outer Bank Lateral Offset Minimum=
    - L1 Outer Bank Height Offset=
    - Increment as L1, L2, L3, etc. as far as you wish to go, every numbering must be used only once and must be in increasing order with distance away from channel center.
    - After the equal sign for the two offset functions you write the offset value in units of meters. For the variability function, see section 9.6.2

- **Right Outer Bank** – Breakline in the right outer channel zone.
    - R1 Outer Bank Function=
    - R1 Outer Bank Lateral Offset Minimum=
    - R1 Outer Bank Height Offset=
    - Increment as R1, R2, R3, etc. as far as you wish to go, every numbering must be used only once and must be in increasing order with distance away from channel center.
    - After the equal sign for the two offset functions you write the offset value in units of meters. For the variability function, see section 9.6.2

9.6.2 Assigning Functions To Channel Sub-Reach Variability Parameters

The way to assign a function to a sub-reach variability parameter is to simply write the name of the function right after the equals sign on each line. For example,

Right Inner Bank Function=SIN3

If you want to add multiple user-defined functions together, simply list all the functions on

subsequent lines. For example,

Right Inner Bank Function=SIN3
Right Inner Bank Function=SIN4
Right Inner Bank Function=COS3
Right Inner Bank Function=COS4

As another example, here is a list with offsets and variability functions:

L1 Outer Bank Lateral Offset Minimum=5
L1 Outer Bank Height Offset=15
L1 Outer Bank Function=SIN7
L1 Outer Bank Function=SIN8
L1 Outer Bank Function=COS7
L1 Outer Bank Function=COS8

9.7 Valley Sub-Reach Variability Parameters

One of the cool features of River Builder is the capability to nest the channel within a valley that has all of its own independent features. Figure 3 illustrates a real river nested in a sinuous valley. River Builder can replicate that by assigning a sub-reach variability function to a valley centerline that is independent of the channel centerline. In addition, the valley can have as many plan view breaklines as you want to implement to create features such as floodplain levels, terraces, roads, levees, yazoo channels, etc.

The way the valley works is very similar to how the outer bank zone works. Specifically,

- There are independent left and right valley zones
- Each zone has a L and R designator
- Each breakline in a valley zone gets a sequential number increasing with distance away from the channel.
- Each breakline has three attributes: a lateral offset, a height offset, and (optionally) a sub-reach variability function.

9.7.1 Valley Breaklines

There are five valley breaklines. Each gets a lateral offset, a height offset, and (optionally) a sub-reach variability function. Offsets are required because without them there is no new breakline.

Beyond the lateral extent of all user defined valley breaklines, there isd one final pair that represents the outer limits of the terrain model. These "Valley "boundary" brealines are always parallel to the valley centerline and thus no functions will apply to them. Only offsets need be specified. If the user forgoes specifying boundary offset values, the program automatically assigns default values of 10 for the lateral offset and 20 for the height offset.

Here is the list of valley breaklines:

- **Valley Centerline** Function – governs the shape of the river's tortuous flow path (i.e. meandering or sinuosity) on the X-Y plane. The channel's centerline is nested within the valley's streamwise coordinate system defined by this function.
    - Valley Centerline Function=

- **Left Valley Level** – Breakline in the river-left valley zone.
    - L1 Valley Level Lateral Offset Minimum=
    - L1 Valley Level Height Offset=
    - L1 Valley Level Function=
    - Increment as L1, L2, L3, etc. as far as you wish to go. Numbering must be in increasing order with distance away from valley center.
    - After the equal sign for the two offset functions you write the offset value in units of meters. For the variability function, see section 9.6.2

- **Right Valley Level** – Breakline in the river-right valley zone.
    - R1 Valley Level Lateral Offset Minimum=
    - R1 Valley Level Height Offset=
    - R1 Valley Level Function=
    - Increment as R1, R2, R3, etc. as far as you wish to go. Numbering must be in increasing order with distance away from valley center.
    - After the equal sign for the two offset functions you write the offset value in units of meters. For the variability function, see section 9.6.2

- Left Valley Boundary – The final, outermost breakline in the river-left valley zone. There is nothing beyond this one, so it is the outer boundary of the terrain model. Because it serves mostly to provide a friendly boundary for 3-D rendering software like GIS, it is designed that user cannot set functions to it.
    - Left Valley Boundary Lateral Offset Minimum=
    - Left Valley Boundary Height Offset=

- Right Valley Boundary – The final, outermost breakline in the river-left valley zone. There is nothing beyond this one, so it is the outer boundary of the terrain model. Because it serves mostly to provide a friendly boundary for 3-D rendering software like GIS, it is designed that user cannot set functions to it.
    - Right Valley Boundary Lateral Offset Minimum=
    - Right Valley Boundary Height Offset=

9.7.2 Assigning Functions To Valley Sub-Reach Variability Parameters

The procedure for assigning sub-reach variability functions for the valley breaklines is identical to that used for the channel, so refer to section 9.6.2.

Here is an example:

L1 Valley Level Lateral Offset Minimum=5
L1 Valley Level Height Offset=15
L1 Valley Level Function=SIN7

L1 Valley Level Function=SIN8

10 Optimization Routine

All parameters require starting values to run the program. Conceptually you might expect that these are the reach-average values, but of course one you apply any one or set of geometric variability functions at channel and valley scales, then it becomes extremely difficult to state in advance what the reach-average values are in the end. If you had to know that at the outset, you would need such a complex calculator that it would defeat the purpose of this software.

The goal of River Builder is to make your job easier and more complete, not much worse. To help you arrive at the final reach-average metrics you want, River Builder include an optimization routine that allows you to identify some reach average values as the final goal. Once these reach average values are defined, the program will automatically iterate with reasonable adjustments on some parameters until a halting condition is met. Of course, if you try to over-specify requirements, then it might become impossible to solve. The simpler the channel, the more you can confidently control and optimize for. If you make the variability functions highly complex, do not expect everything else to be easily forced to some set values.

There are 3 halting conditions to the optimization routine:

1. The calculated reach average value meets the required reach average value.
2. Number of iteration exceeds 100.
3. Unable to satisfy the reach average value requirements even with extreme parameters. For example, if calculated reach average value is greater than required reach average value when lateral offset is set to 0, program halts because lateral offset can't go to negative.

The resolution of reach average value is the same as user defined. For example, if a user defines river slope to be 0.1 (and this value is reasonable with given parameters), then the final calculated river will have a river slope ROUNDED to 0.1. (That is to say, from range 0.05 to 0.14.) If it is set to 0.10, then the final calculated river will have a river slope ROUNDED to 0.10 (range from 0.095 to 0.104).

Only inner channel parameters can have pre-defined reach average values, because outer banks and valley breaklines are not matched in pairs by definition. The following 3 reach average values can be pre-defined for optimization:

- Inner Channel Average Bankfull Width
- Inner Channel Average Bankfull Depth
- River Slope

Here is an example:

Inner Channel Average Bankfull Width=30
Inner Channel Average Bankfull Depth=10.5
River Slope=0.001

Optimization control commands are specified on lines at the end of the River Builder input text file.

11 Generating 3D Renderings of River Builder Outputs

After a user runs River Builder in Python3, then they are going to want to visualize the output and then likely go through an iterative process of design refinement.

By design, River Builder is intended as a creation program, not an analysis program. It does not have any 3D rendering capability within it.

There are so many 3D rendering programs available already, both free and commercial that it has not seemed worth the cost and effort to take on writing our own code from scratch. Meanwhile, if we choose to implement a commercial code for this, then we limit the user base if they do not have a license for that code. Therefore, we leave it to the user to take the simple output file with the topographic points and use it to render the terrain in whatever 2D mapping and/or 3D rendering program they want.

The key file to import into rendering software is **SRVtopo.csv**.

12 Acknowledgements

The underlying research used to develop River Builder came from a variety of sources, including UC Davis, the USDA National Institute of Food and Agriculture [Hatch project number #CA-D-LAW-7034-H], the Ticho Foundation (unrestricted donation), US Army Corps of Engineers, and Yuba County Water Agency (Award #201016094). Dr. Rocko Brown and Dr. Greg Pasternack also contributed their own resources in developing different aspects of the underlying research. We thank postdoctoral scholars Dr. Colin Byrne and Dr. Sebastian Schwindt for conversations about various aspects of this version of the software as we were developing it. Other members of the Pasternack lab group gave input from time to time during 2019 and 2020.

13 References

Blum M. D., Tornqvist, T. E. 2000. Fluvial responses to climate and sea-level change: a review and look forward. Sedimentology 47: 2-48.

Brown, R. A., Pasternack, G. B. 2014. Hydrologic and topographic variability modulate channel change in mountain rivers. Journal of Hydrology 510: 551-564.

Brown, R. A., Pasternack, G. B. 2017. Bed and width oscillations form coherent patterns in a partially confined, regulated gravel–cobble-bedded river adjusting to anthropogenic disturbances, Earth Surface Dynamics, 5, 1-20, doi:10.5194/esurf-5-1-2017.

Brown, R. A. Pasternack, G. B. 2019. How to build a digital river. Earth-Science Reviews. DOI: 10.1016/j.earscirev.2019.04.028.

Brown, R. A., Pasternack, G. B., Lin, T. 2015. The topographic design of river channels for form-process linkages for river restoration. Environmental Management, 57 (4): 929-942. doi: 10.1007/s00267-015-0648-0

Brown, R. A., Pasternack, G. B., Wallender, W. W. 2014. Synthetic river valleys: creating prescribed topography for form-process inquiry and river rehabilitation design. Geomorphology 214: 40-55. 10.1016/j.geomorph.2014.02.025.

Caamaño, D., Goodwin, P., Buffington, J. M.; Liou, J. C. P.; Daley-Laursen, S. 2009. Unifying criterion for the velocity reversal hypothesis in gravel-bed rivers. Journal of Hydraulic Engineering. 135(1): 66-70.

Eubanks, C. E. 2004. Riparian restoration. US Department of Agriculture, Forest Service, Technology & Development Program.

Grill, G., B. Lehner, M. Thieme, B. Geenen, D. Tickner, F. Antonelli, S. Babu, P. Borrelli, L. Cheng, H. Crochetiere, H. Ehalt Macedo, R. Filgueiras, M. Goichot, J. Higgins, Z. Hogan, B. Lip, M. E. McClain, J. Meng, M. Mulligan, C. Nilsson, J. D. Olden, J. J. Opperman, P. Petry, C. Reidy Liermann, L. Sáenz, S. Salinas-Rodríguez, P. Schelle, R. J. P. Schmitt, J. Snider, F. Tan, K. Tockner, P. H. Valdujo, A. van Soesbergen, and C. Zarfl. 2019. 'Mapping the world's free-flowing rivers', Nature, 569: 215-21.

Jackson, J. R., Pasternack, G. B., Wheaton, J. M. 2015. Virtual manipulation of topography to test potential pool-riffle maintenance mechanisms. Geomorphology 228: 617-627. http://dx.doi.org/10.1016/j.geomorph.2014.10.016.

Lane, B. A., Pasternack, G. B., Sandoval-Solis, S. 2018. Integrated analysis of flow, form, and function for river management and design testing. Ecohydrology. DOI: 10.1002/eco.1969.

Lane, B.A., Pasternack, G.B., Dahlke, H.E., Sandoval-Solis, S. 2017. The role of topographic variability in river channel classification. Physical Progress in Geography. 18 doi:10.1177/0309133317718133.

MacWilliams, M. L., Wheaton, J. M., Pasternack, G. B., Kitanidis, P. K., Street, R. L. 2006. The Flow Convergence-Routing Hypothesis for Pool-Riffle Maintenance in Alluvial Rivers. Water Resources Research 42, W10427, doi:10.1029/2005WR004391.

Pasternack, G. B., Wang, C. L., and Merz, J. 2004. Application of a 2D hydrodynamic model to reach-scale spawning gravel replenishment on the lower Mokelumne River, California. River Research and Applications 20:2:205-225.

Pasternack, G. B., Baig, D., Weber, M., Brown, R. 2018a. Hierarchically nested river landform sequences. Part 1: Theory. Earth Surface Processes and Landforms. DOI: 10.1002/esp.4411.

Pasternack, G. B., Baig, D., Weber, M., Brown, R. 2018b. Hierarchically nested river landform sequences. Part 2: Bankfull channel morphodynamics governed by valley nesting structure. Earth Surface Processes and Landforms. DOI: 10.1002/esp.4410.

Schwindt, S., Larrieu, K. G., Pasternack G. B., Rabone, G. 2020. River Architect. SoftwareX 11: 100438. DOI: 10.1016/j.softx.2020.100438.

Thompson, C.J., Croke, J., Fryirs, K. and Grove, J.R., 2016. A channel evolution model for subtropical macrochannel systems. Catena, 139, pp.199-213.

Wheaton, J. M., Pasternack, G. B., and Merz, J. E. 2004. Spawning Habitat Rehabilitation - 2. Using hypothesis development and testing in design, Mokelumne River, California, U.S.A. International Journal of River Basin Management 2:1:21-37.

14 Change Logs

With each update to River Builder there is a list of changes that have been implemented to improve it. As this is the first of version 1.0, there are no changes to report.

15 Mathematical Equation Appendix

This appendix contains a complete list of the mathematical functions used in River Builder to try to have some transparency in plain written text. Obviously, the code is open source, so everybody can study the equations and their implementation in the code directly, but it helps to have a list like this as well.

15.1 Geometric Variability Function Equations

- **Sin and cos functions**

$$f(a) = amplitude * sin(frequency * a + shift)$$

Where,

$$a = 2\pi * \frac{x_i}{x_{max}}$$

- **Sin^2 and cos^2 functions**

$$f(a) = amplitude * sin^2(frequency * a + shift)$$

Where,

$$a = 2\pi * \frac{x_i}{x_{max}}$$

- **Cnoidal function**

  Set:

  $$\boldsymbol{m} \in [\mathbf{0}, \mathbf{1}]$$

  $a = 2\pi * \frac{x_i}{x_{max}} + shift$,

  then:

  $$\boldsymbol{k(m)} = \frac{\mathbf{1}}{\sqrt{\mathbf{1} - \boldsymbol{m} * \boldsymbol{sin(a)}}}$$

  $$\boldsymbol{y_i} = \boldsymbol{amplitude} * (\frac{\boldsymbol{cos(a)}}{\mathbf{2}} * \boldsymbol{k(m)})^{\mathbf{2}}$$

  $\boldsymbol{y_i} = \boldsymbol{y_{max}} - \boldsymbol{y_i}$

- **Perlin noise function**

  Each octave will decrease wavelength and amplitude into half.

  Within each octave:

  $$y_{\lambda*(i+1)} = y_{\lambda*i} + r$$

  λ:wave length

  r: random number between [-amplitude, +amplitude]

  Within each wave:

  $\alpha = \frac{x_i - x_{left}}{x_{right} - x_{left}}$

  $\alpha = 6 * \alpha^5 - 15 * \alpha^4 + 10 * \alpha^3$

  $y_i = (1 - \alpha) * y_{left} + \alpha * y_{right}$

  left: point at the beginning of the current wave

  right: point at the end of the current wave

- **Step function**

  Set:

  $$\boldsymbol{m} \in [\mathbf{0}, \mathbf{1}]$$

  $a = 2\pi * \frac{x_i}{x_{max}} + shift$,

  then:

  $$\boldsymbol{k(m)} = \frac{\mathbf{1}}{\sqrt{\mathbf{1} - \boldsymbol{m} * \boldsymbol{sin(a)}}}$$

  $$\boldsymbol{y_i} = \boldsymbol{amplitude} * (\frac{\boldsymbol{cos(a)}}{\mathbf{2}} * \boldsymbol{k(m)})$$

- **High curvature function**

  p = sinuosity

  s_i - point along meandering line.

  $$\omega = 2.2 * \sqrt{\frac{p-1}{p}}$$

  $$\theta = \omega * cos(s_i)$$

  $$x_{i+1} = x_i + cos(\theta)$$

  $$y_{i+1} = y_i + sin(\theta)$$

15.2 River Bank/Valley Level Related Function Equations

- **Width offset function**

  Left side:

  $$offset_{yi} = y_i + fun(x_i) + min_{offset} - min(fun(x))$$

  Right side:

  $$offset_{yi} = y_i - fun(x_i) - min_{offset} + min(fun(x))$$

- **Depth offset function**

  $$\boldsymbol{z_{i+1} = z_i + offset}$$

15.3 Centerline Related Function equations

- **Centerline calculation**

  $$y = fun_1(x) + fun_2(x) + fun_3(x) + \ldots$$

- **Shields equation**

  $$\underline{H_{BF}} = \frac{(\gamma_S - \gamma_W)\underline{\underline{D_{50}\tau_C}}}{\gamma_W \underline{S}}$$

- **SN-to-XY coordinate system transformation**

  $$x = -\frac{\Delta y}{\sqrt{\Delta x^2 + \Delta y^2}} n + x_1$$
  $$y = \frac{\Delta x}{\sqrt{\Delta x^2 + \Delta y^2}} n + y_1$$

- **XY-to-SN coordinate system transformation**

  Left: $n = \sqrt{\Delta x^2 + \Delta y^2}$; right:$n = -\sqrt{\Delta x^2 + \Delta y^2}$

- **Dynamic Curvature**

  $$radians = arccos(\frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{|\boldsymbol{v}_1| * |\boldsymbol{v}_2|})$$

- **XShape points calculation**

  $$B = \frac{1}{2} * (1 - \left|\frac{cur}{cur_{max}}\right|)$$
  $$L = -log_B 2$$

$$Y = \frac{\frac{w_{BF}(s_i)}{2} - n}{w_{BF}(s_i)}$$

$$z_n(s_i) = 4 * h_{BF}(s_i) * Y^L * (1 - Y^L) \ \ if\ cur \leq 0$$

$$z_n(s_i) = 4 * h_{BF}(s_i) * (1 - Y)^L * (1 - (1 - Y)^L) \ \ if\ cur > 0$$

16 Troubleshooting

Inevitably, River Builder software will require further debugging and improvements as users create interesting scenarios that break the code in unexpected ways. Please email us your problems and we'll do our best to fix them. Obviously, this software is free and we have no consistent funding for software development, so we can nly do what we can in our free time.

16.1 I set a mask function with some ons and offs, but the result is all on. Why doesn't the mask function work?

A recognizable mask function should have the following format:
MASK#=(DIST1, on/off, DIST2, on/off, DIST3, on/off, ...)
DIST means starting from what distance we want to turn on/turn off the function.
DIST should always increases.
DIST should not exceed the maximum distance of the river. (Exceeding parts will be ignored)
DIST should always be integers.
Every dist should pair with a on/off switch.
An example mask with a 200-long reach could be:
MASK1=(0, off, 30, on, 75, off, 135, on)

Check the log to check whether the target mask function appeared in "User defined functions"; if not, it probably means that the format of target mask function is not correct or not defined. Check the spell and number of target mask function.

If the mask function appeared in "User defined functions", then check if it has the correct format as stated above.

16.2 Why is my final 'Average height of Inner Channel' a negative number?

Please check if both 'Valley Slope' and 'Inner Channel Depth Minimum' are set to 0. When 'Inner Channel Depth Minimum' is set to 0, depth will be calculated based on Shield's Equation, and Shield's Equation has to have a positive slope input. Thus we recommend to manually set 'Inner Channel Depth Minimum' if user wants to set slope to 0.