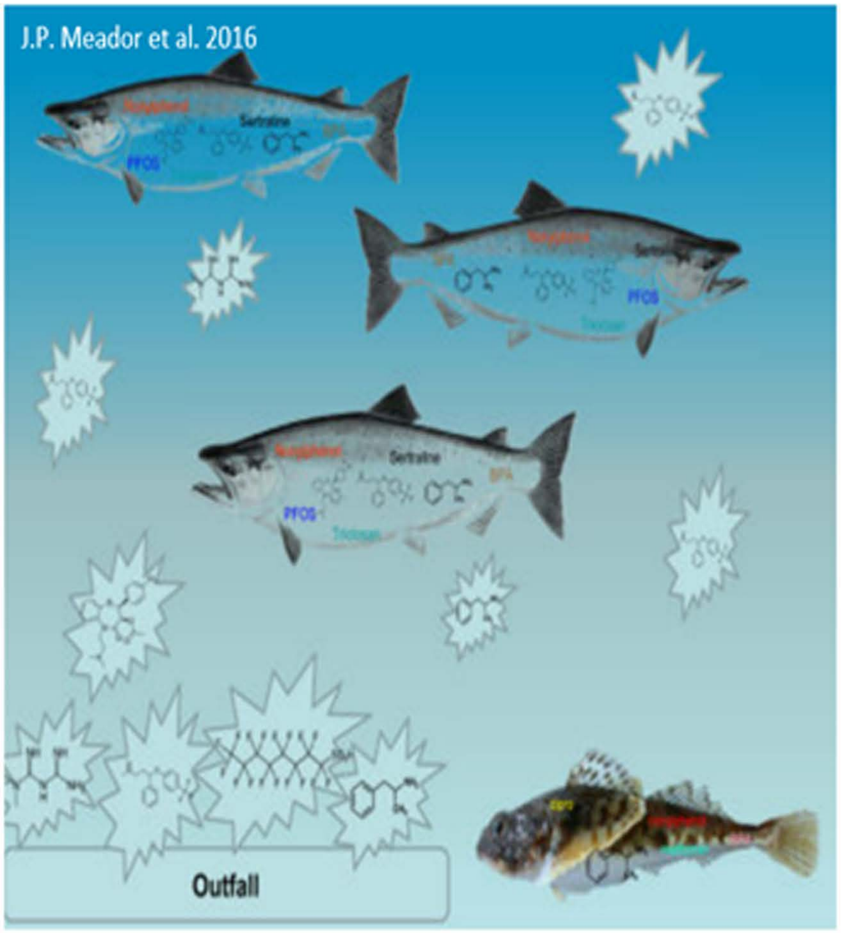
United States Environmental Protection Agency

EPA/600/B-22/024 | March 2022 | www.epa.gov/research

VELMA 2.1 Supplement

VELMA 2.0
User Manual

https://rainbank.info/rainwater-collection-meets-seattle-code/



Office of Research and Development
Center for Public Health and Environmental Assessment
Pacific Ecological Systems Division | Corvallis, Oregon 97333

EPA/600/B-22/024
March 2022
https://www.epa.gov/research

VELMA 2.1 Supplement to VELMA 2.0 User Manual

U.S. Environmental Protection Agency
Office of Research and Development
Center for Public Health and Environmental Assessment
Pacific Ecological Systems Division
Corvallis, Oregon 97333

Authors

Robert B. McKane (mckane.bob@epa.gov)
Allen F. Brookes (brookes.allen@epa.gov)
Jonathan J. Halama (halama.jonathan@epa.gov)
Kevin Djang (djang.kevin@epa.gov)
Bradley L. Barnhart (bradleybarnhart@gmail.com)
Paul B. Pettus (ppettus@gmail.com)
Vivian Phan (phan.vivian@epa.gov)

Acknowledgements

The information in this document has been funded in part by the U.S. Environmental Protection Agency. It has been subjected to the agency's peer and administrative review, and it has been approved for publication as an EPA document. This research was additionally supported in part by NSF grants 0439620, 0436118, and 0922100. We thank our many colleagues who have contributed advice and data during the course of developing VELMA 2.1 and its applications: Heather Golden, Chris Knightes, Don Phillips, Jim Markwiese, Paul Mayer, Joe Ebersole, Nathan Schumaker, Jana Compton, Renee Brooks, Peter Leinenbach, Lisa Chang, Rob Elleman, Angela Adams, Michael Rylko, Catherine Gockel, Dan Brown, Peter Murchie, Rich Fulford, Tom Hollenhorst, Joel Hoffman, John Johnston, Karretta Venable, and many other EPA colleagues; Ed Rastetter, Bonnie Kwiatkowski with the Marine Biological Laboratory; Courtney Zambory with the Oregon Department of Fish and Wildlife; Marc Stieglitz with the Georgia Institute of Technology; Sherri Johnson, Barbara Bond, Mark Harmon, Fred Swanson, Julia Jones, Suzanne Remillard, and other colleagues with the HJ Andrews Long Term Ecological Research (LTER) program; Tarang Khangaonkar with the Pacific Northwest National Laboratory; Ed Kolodziej, Kathy Peter, Joel Baker, Tessa Francis, Nick Georgiadis, Stefano Mazzilli, Guillaume Mauger with the University of Washington; Isaac Kaplan, Hem Nalini Morzaria Luna, Brian Cluer with NOAA; Joe Kane, Justin Hall, Paula Swedeen with the Nisqually Community Forest; David Troutt, Chris Ellings, Sayre Hodgson with the Nisqually Tribe; Cindy Spiry and Matt Baerwalde with the Snoqualmie Tribe; Phillip North, Philip Murphy, Jake McDermott, Michelle Totman with the Tulalip Tribe; Susan Dickerson-Lange and Julia Jay with Natural Systems Design; Mitch Attig and coworkers with the Columbia Land Trust; Jamie Robertson with the Nature Conservancy.

Disclaimer

Neither the United States Environmental Protection Agency (USEPA) nor the Georgia Institute of Technology (GT) makes any warranty or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, product, or process described herein. Reference to any special commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply endorsement, recommendation, or favoring by the USEPA or GT. The views and opinions of the authors do not necessarily reflect those of the USEPA.

Citation

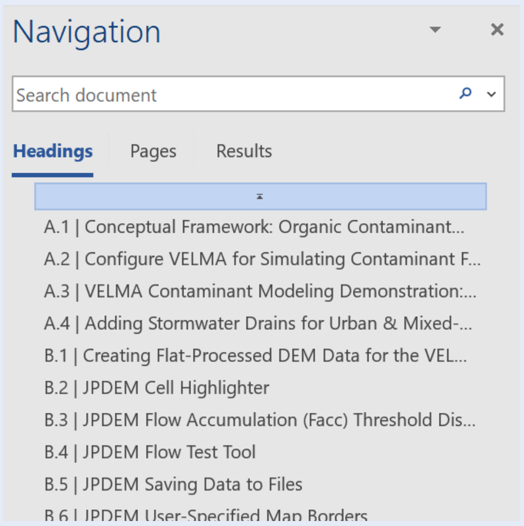
McKane, R.B., A.F. Brookes, J.J. Halama, K.S. Djang, B.L. Barnhart, P.B. Pettus, V. Phan (2022) VELMA 2.1 Supplement to VELMA 2.0 User Manual. U.S. Environmental Protection Agency, EPA/600/B-22/024

EXECUTIVE SUMMARY

This main purpose of this VELMA 2.1 tutorial supplement to the original VELMA 2.0 User Manual (McKane et al. 2014) is to document concepts and methods for modeling effects of green infrastructure and stormwater (gray) infrastructure on contaminant fate and transport in urban and mixed-use watersheds.

This supplement also describes new and updated methods developed since 2014 User Manual that significantly extend VELMA's applicability for a wider range of environmental conditions and decision support needs. For example, VELMA 2.1 procedures are described for explicitly modeling surface water and chemical transport on impervious surfaces to stormwater pipe inlets (drains) and outlets to streams and estuaries. Methods are described for setting up and modeling green stormwater infrastructure (rain gardens, bioswales, riparian buffers, etc.) and its effectiveness for reducing stormwater contaminant runoff through bioretention and enhanced degradation.

We have strived to make the VELMA 2.1 tutorial information in this document as user-friendly as possible. There is a lot of material packed into its nearly 300 pages, so we recommend that users take full advantage of the Table of Contents Navigation feature in Word. Click the View menu and check "Navigation Pane" to see this. It can help you browse the full scope of chapter topics (click on any chapter heading to go directly to that chapter) or search for a topic of interest in the "Search Document" window.



A.1 | Conceptual Framework: Organic Contaminant...
A.2 | Configure VELMA for Simulating Contaminant F...
A.3 | VELMA Contaminant Modeling Demonstration:...
A.4 | Adding Stormwater Drains for Urban & Mixed-...
B.1 | Creating Flat-Processed DEM Data for the VEL...
B.2 | JPDEM Cell Highlighter
B.3 | JPDEM Flow Accumulation (Facc) Threshold Dis...
B.4 | JPDEM Flow Test Tool
B.5 | JPDEM Saving Data to Files
B.6 | JPDEM User-Specified Map Borders

R.B. McKane
12/28/2021

Reference

McKane, R, A Brookes, K Djang, M Stieglitz, A Abdelnour, F Pan, J Halama, P Pettus, D Phillips (2014) Velma 2.0 User Manual and Technical Documentation. U.S. Environmental Protection Agency

Contents

A.1 | Conceptual Framework: Organic Contaminant Modeling Using VELMA

Overview *(Tutorial A.1_HowTo_VELMA Contaminant Modeling Conceptual Framework)*

This document describes the conceptual framework underpinning the use of VELMA 2.1 to model fate and transport of organic contaminants within watersheds. We review how VELMA 2.1 simulates contaminant fate and transport within soils and hillslopes as a function of two processes: (1) the partitioning of the total amount of a contaminant between sorbed (immobile) and aqueous (mobile) phases; and (2) the vertical and lateral transport of the contaminant’s aqueous phase within surface and subsurface waters.

Subsequent sections describe the interplay of contaminant sorption/desorption processes with VELMA’s ecohydrological processes that control vertical and lateral transport within soils, hillslopes and watersheds.

This “HowTo” VELMA document is a companion to the two others listed below. We recommend reading this A.1 concepts document before the other two.

- Tutorial A.2 – Configure VELMA for Simulating Contaminant Fate and Transport
- Tutorial A.3 – VELMA Contaminant Modeling: Longfellow Creek Example

1.0 Introduction

Hundreds of thousands of organic contaminants have been and continue to be produced and released into the environment in a range of quantities. Some are persistent organic pollutants (POPs) resistant to environmental degradation (El-Shahawi et al. 2010). Pesticides, solvents, pharmaceuticals and industrial chemicals are a few examples. Because of their persistence, POPs can bioaccumulate and adversely affect terrestrial and aquatic food webs and human health.

Therefore, it is important to understand the “life cycle” of organic contaminants after they are released into the environment. Where and when is a contaminant deposited and in what quantities? How is it then adsorbed, degraded and transported within natural and built environments, and what is its fate in space and time?

Answers to these questions are elusive, requiring some combination of environmental monitoring (refs), advanced analytical methods (refs), and data synthesis and analysis tools. Computer simulation modeling can potentially play a larger role in data synthesis and analysis, though adequate tools are few.

VELMA 2.0 (McKane et al. 2014) has been enhanced as VELMA 2.1 to simulate the fate and transport of organic contaminants within watersheds, providing an opportunity to facilitate data synthesis and analysis for projects having enough monitoring and ecotoxicological data (refs).

This document provides a guide for using VELMA 2.1 to model fate and transport of organic contaminants. Our intent is to provide a state-of-the-science tool that addresses a specific modeling gap, namely, a spatially-distributed and scalable model for simulating process-level hydrologic and biogeochemical controls on contaminant sorption-desorption, degradation and vertical and lateral transport within soils and watersheds. Other documents in this VELMA 2.1 package illustrate new techniques for simulating the effectiveness of natural and engineered green infrastructure – riparian buffers, rain gardens, bioswales, pervious pavements, green roofs and others – for reducing contaminant loads to surface and groundwaters.

VELMA 2.1 is designed to allow users to model essentially any organic contaminant in the EPA Comptox library for which solubility, partitioning, degradation and other supporting coefficients are available (https://comptox.epa.gov/dashboard/). Comptox provides information for over 760,000 chemical substances, the majority of which are organic contaminants (Williams et al. 2017; McEachran et al. 2017).

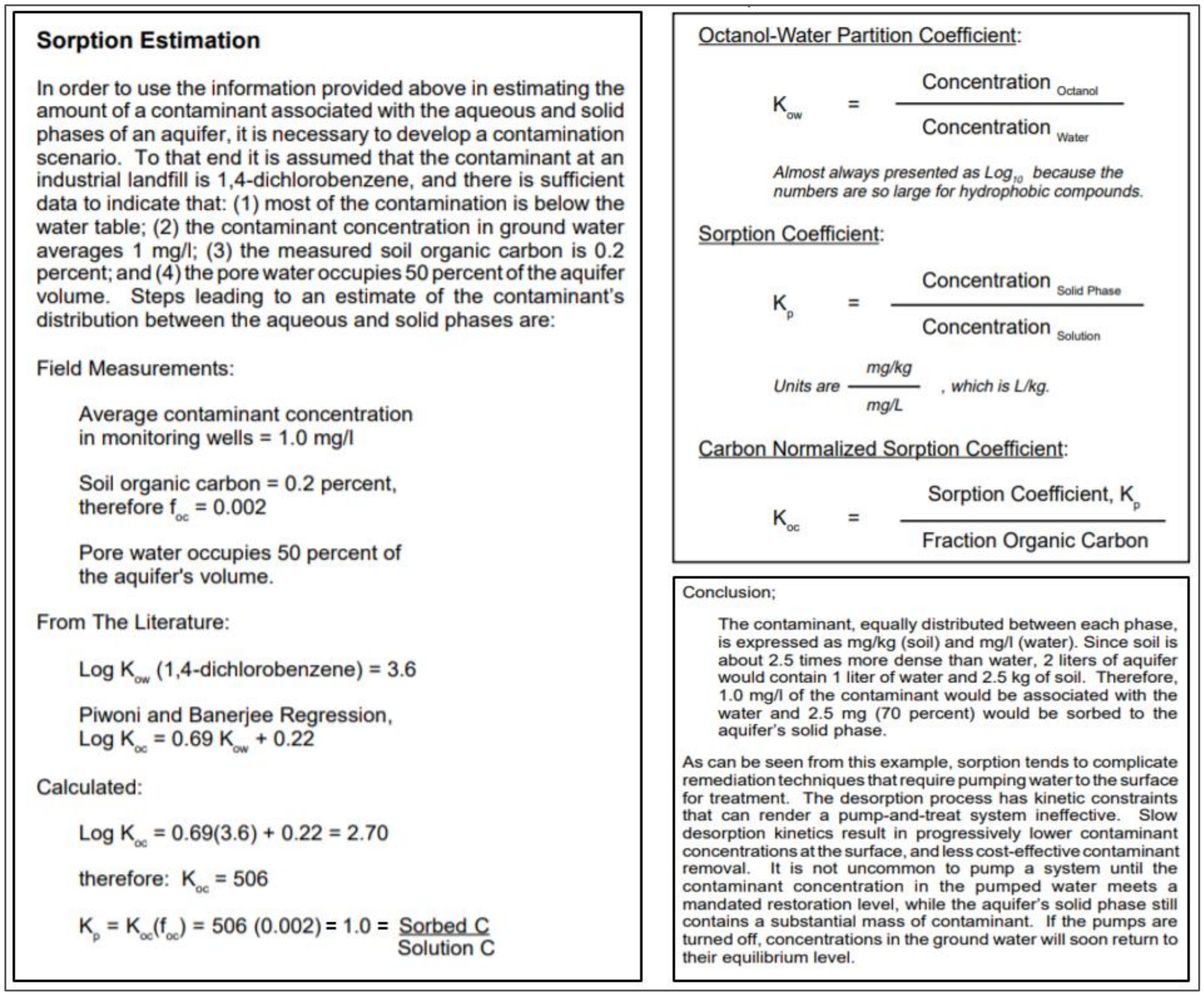
2.0 Conceptual Framework

In VELMA 2.1, contaminant fate and transport within soils and hillslopes is primarily controlled by two processes: (1) the partitioning of the total amount of a contaminant between sorbed (immobile) and aqueous (mobile) phases; and (2) the vertical and lateral transport of the contaminant’s aqueous phase within surface and subsurface waters. Sections 2.1 and 2.2 describe the conceptual basis for the partitioning and transport modeling steps in VELMA.

2.1 Contaminant Partitioning between Sorbed and Aqueous Phases

VELMA 2.1 incorporates the methods of Piwoni and Keeley (1990; https://permanent.access.gpo.gov/lps49292/issue6.pdf) to describe the processes by which organic contaminants are partitioned between sorbed (bound by soil, immobile) and aqueous (dissolved, mobile) phases within soil columns, such as those represented in VELMA. The section on “Sorption Estimation” in Piwoni and Keeley (1990), presented below as Figure 1, provides a concise summary of the processes, parameters and calculations used to estimate partitioning of any given contaminant into its sorbed and aqueous phases. These parameters and methods have been built into VELMA 2.1.

(continued)



Sorption Estimation

In order to use the information provided above in estimating the amount of a contaminant associated with the aqueous and solid phases of an aquifer, it is necessary to develop a contamination scenario. To that end it is assumed that the contaminant at an industrial landfill is 1,4-dichlorobenzene, and there is sufficient data to indicate that: (1) most of the contamination is below the water table; (2) the contaminant concentration in ground water averages 1 mg/l; (3) the measured soil organic carbon is 0.2 percent; and (4) the pore water occupies 50 percent of the aquifer volume. Steps leading to an estimate of the contaminant's distribution between the aqueous and solid phases are:

Field Measurements:

Average contaminant concentration in monitoring wells = 1.0 mg/l

Soil organic carbon = 0.2 percent, therefore $f_{oc} = 0.002$

Pore water occupies 50 percent of the aquifer's volume.

From The Literature:

Log K_{ow} (1,4-dichlorobenzene) = 3.6

Piwoni and Banerjee Regression,
$\text{Log } K_{oc} = 0.69\ K_{ow} + 0.22$

Calculated:

$\text{Log } K_{oc} = 0.69(3.6) + 0.22 = 2.70$

therefore: $K_{oc} = 506$

$K_p = K_{oc}(f_{oc}) = 506\ (0.002) = 1.0 = \dfrac{\text{Sorbed C}}{\text{Solution C}}$

Octanol-Water Partition Coefficient:

$$K_{ow} = \frac{\text{Concentration}_{\text{Octanol}}}{\text{Concentration}_{\text{Water}}}$$

Almost always presented as Log_{10} because the numbers are so large for hydrophobic compounds.

Sorption Coefficient:

$$K_p = \frac{\text{Concentration}_{\text{Solid Phase}}}{\text{Concentration}_{\text{Solution}}}$$

Units are $\dfrac{mg/kg}{mg/L}$ *, which is L/kg.*

Carbon Normalized Sorption Coefficient:

$$K_{oc} = \frac{\text{Sorption Coefficient, } K_p}{\text{Fraction Organic Carbon}}$$

Conclusion;

The contaminant, equally distributed between each phase, is expressed as mg/kg (soil) and mg/l (water). Since soil is about 2.5 times more dense than water, 2 liters of aquifer would contain 1 liter of water and 2.5 kg of soil. Therefore, 1.0 mg/l of the contaminant would be associated with the water and 2.5 mg (70 percent) would be sorbed to the aquifer's solid phase.

As can be seen from this example, sorption tends to complicate remediation techniques that require pumping water to the surface for treatment. The desorption process has kinetic constraints that can render a pump-and-treat system ineffective. Slow desorption kinetics result in progressively lower contaminant concentrations at the surface, and less cost-effective contaminant removal. It is not uncommon to pump a system until the contaminant concentration in the pumped water meets a mandated restoration level, while the aquifer's solid phase still contains a substantial mass of contaminant. If the pumps are turned off, concentrations in the ground water will soon return to their equilibrium level.

Figure 1. Quoted material from Piwoni and Keeley (1990) describing an adaptable set of equations for estimating the partitioning of a contaminant, in this case 1,4-dichlorobenzene, between groundwater (aqueous phase) and soil organic matter (sorbed phase).

Figure 1 describes several contaminant model parameters that require additional comment. K_{OW} is the octanol-water partition coefficient for a specified organic chemical. Octanol is used as a standard organic chemical for calculating hydrophobicity of organic contaminants, particularly lipids. When an organic contaminant of interest – dichlorobenzene in the Figure 1 example – is placed in a well-mixed, octanol-water system, K_{OW} is the ratio of the contaminant's concentration in the octanol phase to its concentration in the aqueous phase. This octanol-water partitioning procedure is determined experimentally.

VELMA 2.1 users can find the published K_{OW} value for contaminants of interest in EPA's Comptox library (https://comptox.epa.gov/dashoard/). Comptox also provides estimates for other contaminant biophysical constants that VELMA requires, for example, solubility in water (mol/L) and biodegradation half-life (days) – See sections 3.0 – 3.4 for details.

As described in the lower left panel of Figure 1, two additional pieces of information – the amounts of soil organic carbon and water within each soil layer – are needed before VELMA can translate the unitless ratio for the octanol/water partition coefficient, K_{OW}, to the more meaningful soil/water sorption coefficient, K_P.

That is, K_P must be calculated to address the question: What portion of the total amount of a contaminant within a soil layer is sorbed by soil (mg contaminant/kg bulk soil) versus that which is dissolved in soil water (mg contaminant/L)?

There are three steps to answering this question.

Step 1: Recall that octanol only provides a proxy, ratio-based measure of a contaminant’s affinity for soil organic matter and, further, that soil organic matter concentrations tend to vary greatly with depth and across soil columns and landscapes. Therefore, to translate K_{OW} to K_P, it is first necessary to obtain estimates of how a specified mass of soil organic matter is distributed within soils, both vertically and laterally. Because VELMA is a grid-based watershed model with each grid cell representing a 4-layer soil column, users must specify the total amount of soil organic matter (carbon g/m^2) within each soil column in the modeled watershed. In VELMA, well-decomposed (unidentifiable) soil organic matter is referred to as “humus”. Typically, this task can be accomplished using SSURGO-quality soil survey data (see VELMA 2.1 How To documentation, “Tutorial D.2 – Soil Data Mapping and Parameter Initialization”). With that information, VELMA internally calculates the vertical distribution of humus carbon for soil layers 1-4. Details on VELMA’s soil carbon submodel and its application can be found in Abdelnour et al. (2011; 2013) and McKane et al. (2014).

Step 2: With the specified survey-based spatial estimates of soil organic carbon (humus), VELMA internally calculates the carbon normalized sorption coefficient, K_{OC}. This requires user input of published empirical regression parameters describing the relationship between K_{OW} and K_{OC}. For example, the lower left panel of Figure 1 uses linear regression parameters ($\log K_{OC} = 0.69 * \log K_{OW} + 0.22$) described by Piwoni and Banerjee (1989).

They found that the same parameters accurately predicted sorption of 1,2-dichlorobenzene and other organic solvents, within the limits of their experimental system. Contaminant-specific regression parameters for calculating K_{OC} can often be applied to chemically similar contaminants, with the caveat that soil organic matter concentrations and mineralogy are within the range of experimental conditions for which the regression was developed (Piwoni and Banerjee 1989; Ho et al. 2005).

VELMA 2.1 allows users to specify parameters for published linear regression equations describing contaminant sorption behavior. While linear regression is perhaps the most common approach for modeling contaminant sorption, it can lack accuracy across broad contaminant and soil organic matter concentration ranges. More robust nonlinear sorption regression options, such as Freundlich, Langmuir and others (Ho et al. 2005), will also be built into VELMA.

Step 3: VELMA then uses the regression-based estimate of K_{OC} (Step 2, above) to internally calculate the soil sorption coefficient, K_P (Figure 1, lower left panel). This third step also requires an estimate of the water content of the soil layer, which VELMA internally calculates for each day of the simulation. The ecohydrological processes by which VELMA simulates soil water content are described in section 2.1.

The preceding information is more concisely summarized in Table 1, below. Table 1 also includes a contaminant solubility parameter that users must enter into VELMA, so that the model can internally calculate the amount (mol/L and g/m^2) contaminant that can be dissolved into solution. Section 3.3.2 provides details.

Table 1. Summary of coefficients for dynamically estimating the amount of a specified contaminant associated with the aqueous and solid (sorbed) phases within soil columns in VELMA.

Contaminant coefficient	Definition and explanation	Units	Data Source
Log K_{OW}	**Octanol-Water Partition Coefficient.** K_{OW} ($\log_{10}$) is the experimentally determined ratio of a contaminant's concentration in the octanol ($C_8H_{17}OH$) phase of a two-phase octanol-water system. K_{OW} is a standard indicator of a contaminant's affinity for adsorption by organic matter. Higher K_{OW} values indicate strongly sorbed, less mobile (hydrophobic) chemicals.	unitless ratio, Log_{10}	User enters K_{OW} value into VELMA based on EPA Comptox website, https://comptox.epa.gov/dashboard. Note that Comptox expresses K_{OW} as K_P, a confusing use of terminology because K_P recalculates K_{OW} to address the effect of variations in soil organic matter on contaminant sorption.
f_{OC}	**Soil Organic Carbon Fraction**. In VELMA, f_{OC} varies with soil type and depth and changes dynamically with climate and management. It is used to convert a contaminant's octanol-water partition coefficient, K_{OW}, to K_{OC} (see below).	Fraction (0-1) = [(g soil humus carbon) / (g bulk soil)]. VELMA calculates f_{OC} per soil layer per grid cell per day.	In VELMA, users specify initial humus (soil organic matter) values (g C/m^2) according to soil survey or other data, for example, NRCS soil survey map data. VELMA then calculates initial f_{OC} values and subsequent daily changes in f_{OC} in response to environmental and management drivers (McKane et al. 2014).
Log K_{OC}	**Organic Carbon Normalized Sorption Coefficient.** K_{OC} is calculated based on published regressions relating K_{OC} to K_{OW} and f_{OC}. For example: log K_{OC} = 0.69*(K_{OW}) + 0.22 for 1,4-dichlorobenzene (Piwoni & Banerjee 1980). K_{OC} is used to calculate K_P (see below).	Unitless, $\log_{10}$	Users need to specify published regression parameters that VELMA uses to calculate K_{OC} as a function of K_{OW} and f_{OC}. See Figure 1 (left panel). Also see Figure 5 for K_{OC} regression coefficient parameter names in VELMA. VELMA uses these coefficients to internally calculate daily changes in Log K_{OC}, which is used in turn to internally calculate K_P.
K_P	**Soil-Water Partition Coefficient.** K_P describes the partitioning of contaminant between soil and water, expressed as the ratio [(mg/kg bulk soil) / (mg/liter water)]. VELMA internally calculates $K_P = K_{OC}$ (f_{OC}), where K_{OC} is the antilog of log K_{OC}. Thus, K_P is conceptually similar to K_{OW}, however, K_P provides a more precise estimate of contaminant sorption by factoring in the dominant effect of soil organic matter on sorption in soils. Higher K_P values indicate less mobile (hydrophobic) chemicals.	Unitless ratio	VELMA internally calculates daily changes in K_P based on methods of Piwoni and Keeley (see Figure 1).
Solubility	**Solubility** of a specified contaminant in water.	mol/L	EPA Comptox website, https://comptox.epa.gov/dashboard

2.2 Ecohydrological Regulation of Contaminant Fate and Transport

VELMA provides a detailed treatment of the hydrological processes that control how water moves within watersheds – vertically within soil columns, and laterally along hillslopes and eventually to streams. As water moves through the watershed it can pick up and transport solubilized contaminants, which can be sorbed, desorbed, or not, at any point along a flow path, depending upon changing hydrologic and biogeochemical conditions.

Figure 2A and Figure 2B conceptually describe the VELMA 2.1 hydrologic submodel, focusing on soil water pools and fluxes affecting transport of dissolved contaminants and nutrients. Details on hydrologic equations parameters and equations can be found in Abdelnour et al. (2011) and McKane et al. (2014).

The most significant VELMA 2.1 updates Figure 2A and Figure 2B include spatially explicit transfers of surface water, S_W (Figure 2A). VELMA 2.1 dynamically simulates daily transfers of surface water and dissolved contaminants and nutrients in and out of each grid cell on a daily time step. Impervious and semi-pervious surfaces can be simulated, including pavement, roofs, rock outcroppings, etc. Geospatially defined curbs, stormwater drains, and pipes can intercept and redirect surface flow to specified outflows (Figure 2B). Curb cuts for directing stormwater runoff to roadside rain gardens can also be simulated. In short, the addition of a spatially-explicit surface layer to VELMA 2.1 makes it possible to better simulate water/contaminant/nutrient fate and transport in urban and mixed-use watersheds to a high degree of geographic specificity (5 or 10-meter grids) where such data are available. Where centralized urban stormwater databases are not available, stormwater drains and pipes can be imputed using road network grid GIS databases and general stormwater system knowledge of municipal stormwater managers (Halama et al. in preparation).

In summary, Figures 2A and 2B need to be considered together to understand the hydrologic processes in VELMA 2.1 affecting contaminant fate and transport in urban watersheds equipped with stormwater infrastructure. Not shown in these figures are the interacting biogeochemical processes and parameters that modify hydrologically-driven contaminant transport processes in VELMA 2.1. These are: (1) Solubilization of contaminants deposited on impervious surfaces (S_W in Figure 2A) or on soil layer 1; (2) Contaminant sorption-desorption kinetics between sorbed and aqueous phases; (3) and contaminant degradation. Modeled biogeochemical processes and parameters for these processes are described in Tutorial A.1 subsection 2.1 (above), and Tutorial A.2. It is the interaction of these hydro-biogeochemical (ecohydrological) processes that control transport of dissolved contaminants within cells, between neighboring cells, and to distant cells and outfalls to stream channel cells via storm drains and pipes.

VELMA 2.1 Ecohydrology Model Conceptual Diagram

Drivers & Specified Parameters

P = daily total precipitation
T = daily mean temperature
Φ_i = soil porosity in layer i
z_i = thickness of soil layer i
DTB = depth to bedrock

Calibration Parameters

Ks_i = saturated hydraulic conductivity, soil layer i
Ksv_i = vertical saturated hydraulic conductivity, soil layer i
Ksl_i = lateral saturated hydraulic conductivity, soil layer i

Response Variables

m = snow melt
S_{SWE} = snow water equivalent
S_w = water on surface (above soil)
I = infiltration of S_w into soil layer 1
s_i = soil water content in layer i
D_i = vertical drainage from soil layer i to next layer
Q_s = surface runoff in or out (above soil)
Q_i = subsurface runoff in or out of layer i
ET_i = evapotranspiration from layer i

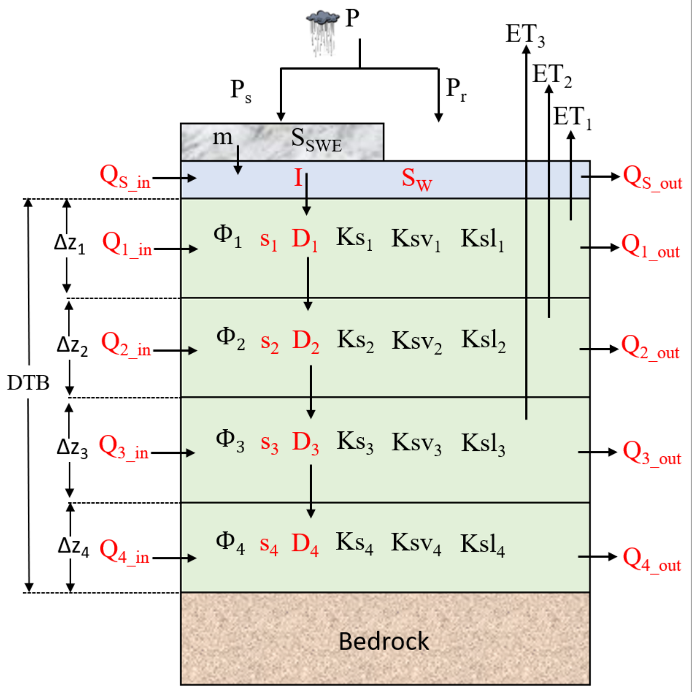


Figure 2A. Hydrologic processes simulated in VELMA 2.1 soil columns. Contaminant transport is directly affected by red highlighted pools and fluxes.

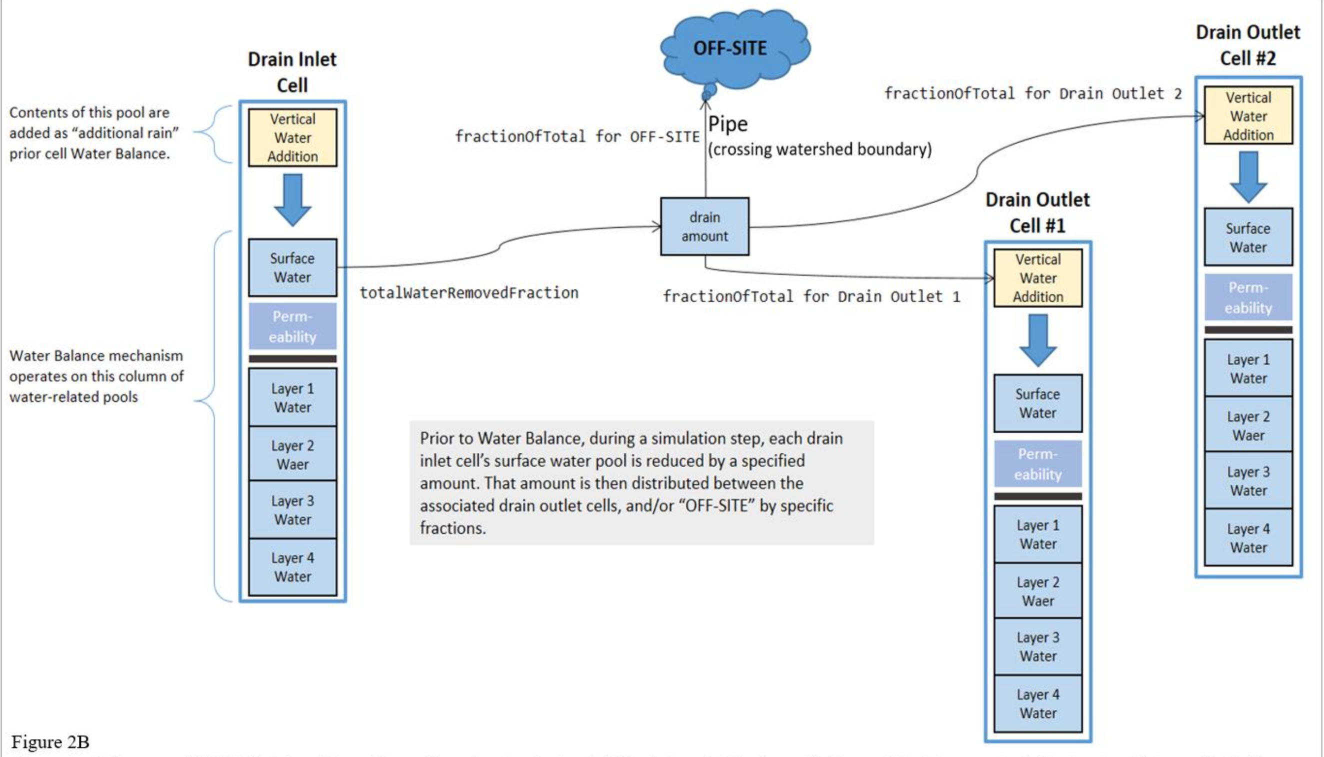

Drain Inlet Cell
Contents of this pool are added as “additional rain” prior cell Water Balance.
Vertical Water Addition
Surface Water
Perm-eability
Layer 1 Water
Layer 2 Waer
Layer 3 Water
Layer 4 Water
Water Balance mechanism operates on this column of water-related pools
OFF-SITE
fractionOfTotal for OFF-SITE
Pipe (crossing watershed boundary)
drain amount
totalWaterRemovedFraction
fractionOfTotal for Drain Outlet 1
fractionOfTotal for Drain Outlet 2
Drain Outlet Cell #1
Drain Outlet Cell #2
Prior to Water Balance, during a simulation step, each drain inlet cell’s surface water pool is reduced by a specified amount. That amount is then distributed between the associated drain outlet cells, and/or “OFF-SITE” by specific fractions.

Figure 2B
Conceptual diagram of VELMA 2.1 cell-to-cell transfers of water (and solubilized chemicals) via explicitly modeled stormwater infrastructure features, including curbs, drains, and pipes. VELMA’s Water Drain Disturbance routine transfers water and chemicals between non-adjacent cells or removes it from the simulation watershed. It provides a way to redistribute or remove water within a watershed at specific steps of a VELMA Simulation run. Users specify one or more drain inlet cell locations for a given Water Drain Disturbance, along with parameters that determine when, and under what conditions, those drain inlet cells are drained. Users also specify whether the water drained from inlet cells is transferred to drain outlet cells in the same watershed or is removed from the watershed.

2.3 Other Processes Affecting Contaminant Fate and Transport

Organic contaminants added to a VELMA simulation configuration can also undergo degradation. Contaminant degradation in VELMA is based on generalized organic matter decay equations and applied using microbial efficiency and decay parameters (Potter et. al 1993; see McKane et al. 2014).
No other soil and plant processes currently occur for contaminants. Plant uptake of contaminants is not currently simulated but is functionally possible with VELMA’s existing plant-soil submodel (McKane et al. 2014) and could be added later to address phytoremediation actions.

2.4 Summary of Processes Controlling Contaminant Fate and Transport in VELMA

The following bullets summarize fate and transport concepts described in the preceding sections, in the context of steps outlined in subsequent sections for configuring VELMA 2.1 to simulate contaminant fate and transport in watersheds for which it has been parameterized.

<u>Steps</u>

1. No contaminant processing occurs unless contaminants are initialized into soil columns at time zero of a simulation, or are configured to deposit onto the surface during a simulation run. Section 5.0 provides details for instructing VELMA on how much, when and where to deposit a contaminant.
2. Contaminant amounts deposited onto the surface are transported laterally (to other surface cells) and vertically (infiltrating to the top soil layer) based on water movement calculated by VELMA’s water-

balance mechanism, coupled with the contaminant's specified solubility (section 3, below). Yellow highlighting in Figure 2 (section 2.2) illustrate the pertinent infiltration (I) and runoff inflows and outflows (Q_{in}, Q_{out}) that VELMA internally calculates each day.

3. Contaminant amounts within soil layers undergo partitioning into aqueous and sorbed amounts. Sorbed amounts are a function of soil organic matter (humus) and soil water amounts (sections 2.1 and 2.2).
4. The aqueous (dissolved) portions of the contaminant within the soil layers undergo lateral and vertical transport based on VELMA's water-balance mechanism (Figure 2 highlighted fluxes), coupled with the contaminant's specified partition coefficients controlling sorbed/aqueous ratios (sections 2.1 and 2.2).
5. Sorbed contaminant amounts occurring within soil layers decompose using the same Potter equation employed for VELMA's core chemistry calculations (McKane et al. 2014) (section 3.3.3).
6. Steps 2 through 5 occur for any cell containing contaminant amounts for every time step of the simulation run.

3.0 How to Configure VELMA to Simulate Contaminant Fate and Transport

VELMA 2.1 supports the addition of one or more organic contaminants to a simulation configuration.
As described in the following sections, users need to add each contaminant with its own set of parameters in order to simulate fate and transport of specific organic contaminants (e.g., DDT) during a simulation run.

Configuring one or more contaminant parameterizations requires configuration of an equal number (or more) of VELMA disturbance parameterizations to introduce specified amounts of the contaminant into the simulation state. VELMA disturbances can be configured to introduce specified amounts of contaminant to specific locations (grid cells) of the simulated watershed and at specified days during the simulation run.

Note that unlike VELMA's atmospheric nitrogen deposition parameter, "nin", there is no designated parameter in the VELMA 2.1 that users can use to automate uniform, watershed-scale daily additions of a contaminant. However, an alternate method for implementing uniform daily amounts of contaminant deposition can be used to create a contaminant deposition map and supporting parameters to deposit spatially and temporally uniform amounts of a contaminant to all cells within the watershed. Users have option of creating deposition maps that are uniform or variable in time and space across a watershed.

VELMA 2.1 is programmed to automate the process of generating Spatial Data Writers that produce grid-based data arrays (ascii grid files) describing spatial and temporal changes in contaminant pools and fluxes during a simulation. VELMA saves these ascii grid files (.asc) to the VELMA results folder. At the end of a simulation, users can load these ascii grid files into ArcGIS or other GIS-based visualization tool to develop spatial and temporal visualizations and animated movies showing contaminant fate and transport within the modeled watershed. When a user sets up a contaminant simulation run, VELMA automatically sets up a generic Spatial Data Writer prefix name that cannot be modified but to which the user can add a suffix name that defines specific contaminants and pools to be saved as ascii grids in the VELMA results folder. Section 23.0 in the VELMA 2.0 User Manual (McKane et al. 2014) provides details for setting up Spatial Data Writers.

3.1 Optional: How to Configure VELMA for Simulating Contaminant Fate and Transport in Parallel Mode

This optional section provides instructions for setting up a contaminant fate and transport simulation for one or more contaminants, but while running VELMA in parallel mode – that is, for simultaneously running separate simulations in parallel for the multiple subwatersheds that make up a larger basin. The option of running VELMA in parallel mode can save time for simulating large watersheds. However, this is unnecessary if the simulated watershed is less than 10 km^2.

Running in parallel model, VELMA will merge hydrologic and biogeochemical outputs from each subwatershed’s pour point to the appropriate downstream reaches, such that spatial and temporal model outputs are entirely consistent with outputs when VELMA is run for the entire watershed. The only difference between running VELMA in parallel mode versus whole-watershed model is that parallel mode takes significantly less time.

To set up a parallel mode simulation, start with a “proven”, VELMA simulation configuration – i.e., a simulation configuration .xml file that is already calibrated and known to run reliably and accurately for the study site.

Load that simulation configuration .xml file into JVelma – for details, see McKane et al. (2014) – and immediately change its “Simulation Run Name”, then save it. This creates a copy of the original .xml that you can modify with contaminant additions, while saving the original .xml for single-process use.

With the new copy of the proven simulation configuration loaded into JVelma, adding one or more contaminants involves the following steps for each contaminant you wish to add:

1. **Use JPDEM to discover and note the subwatershed outlets of interest for running VELMA in parallel model.** For instructions see the tutorial named “Tutorial B.9 – JPDEM Sub-Reach Delineations”.
2. **Parameterize a VELMA 2.1 simulation configuration for MULTI (parallel) model runs.** For instructions see the tutorial named “Tutorial C.2 – How to Run VELMA Parallel Mode”.
3. **Add a new Contaminant parameterization group to the MULTI simulation configuration.** For instructions see the tutorial named “How-to” documentation named “Tutorial A.2 – Configure VELMA to Add Organic Contaminants”.
4. **Add one or more Disturbance parameterization groups to the simulation configuration file (.xml) to add specified amounts (g/m^2) at specific times and locations during the simulation.** For instructions see the tutorial named “Tutorial A.3 – VELMA Contaminant Modeling: Longfellow Creek Example”
5. **(Optional) Add one or more Spatial Data Writer parameterization groups to produce spatially-explicit map results for the contaminant.** For instructions see the tutorial named “Tutorial D.8 – Cell Data Writer Configuration”.
   Note: The DailyResults and any configuration Cell Data Writer .csv files automatically include columns for any contaminants added to simulation configuration. Only Spatial Data Writers must be explicitly parameterized.

Save the changes to the simulation configuration .xml file. It is now ready for use in VELMA Parallel Mode.

3.2 How to Add a Contaminant Parameterization to a VELMA Configuration

With a proven VELMA simulation configuration loaded into JVelma,
click the Edit→Contaminants→“Add a New Contaminant” menu item, as shown in **Figure 3**, below:

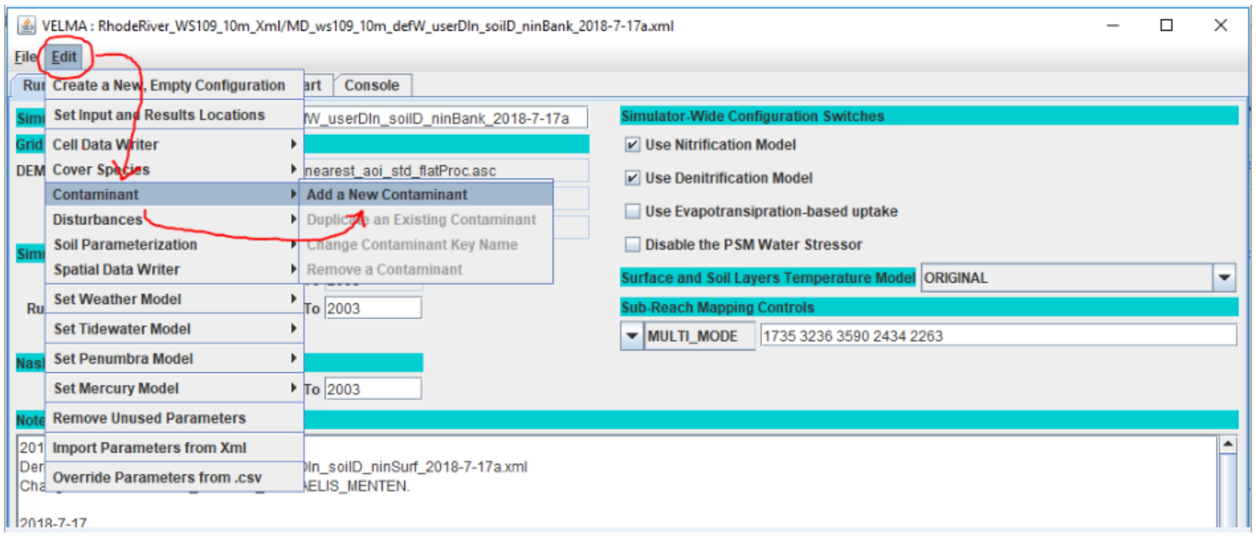
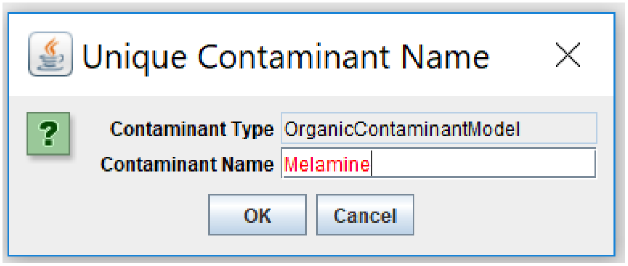

VELMA : RhodeRiver_WS109_10m_Xml/MD_ws109_10m_defW_userDln_soilD_ninBank_2018-7-17a.xml
File
Edit
Create a New, Empty Configuration
Set Input and Results Locations
Cell Data Writer
Cover Species
Contaminant
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Set Penumbra Model
Set Mercury Model
Remove Unused Parameters
Import Parameters from Xml
Override Parameters from .csv
Add a New Contaminant
Duplicate an Existing Contaminant
Change Contaminant Key Name
Remove a Contaminant
Console
Simulator-Wide Configuration Switches
Use Nitrification Model
Use Denitrification Model
Use Evapotranspiration-based uptake
Disable the PSM Water Stressor
Surface and Soil Layers Temperature Model
ORIGINAL
Sub-Reach Mapping Controls
MULTI_MODE
1735 3236 3590 2434 2263

Figure 3.

In the "Unique Contaminant Name" pop-up dialog (Figure 4**)** that opens next, provide a name for the contaminant. The name should begin with an alphabet character and may consist of only alpha-numeric and underscore ("_") characters and must remain unique among all contaminants in the configuration.


Unique Contaminant Name

Contaminant Type: OrganicContaminantModel

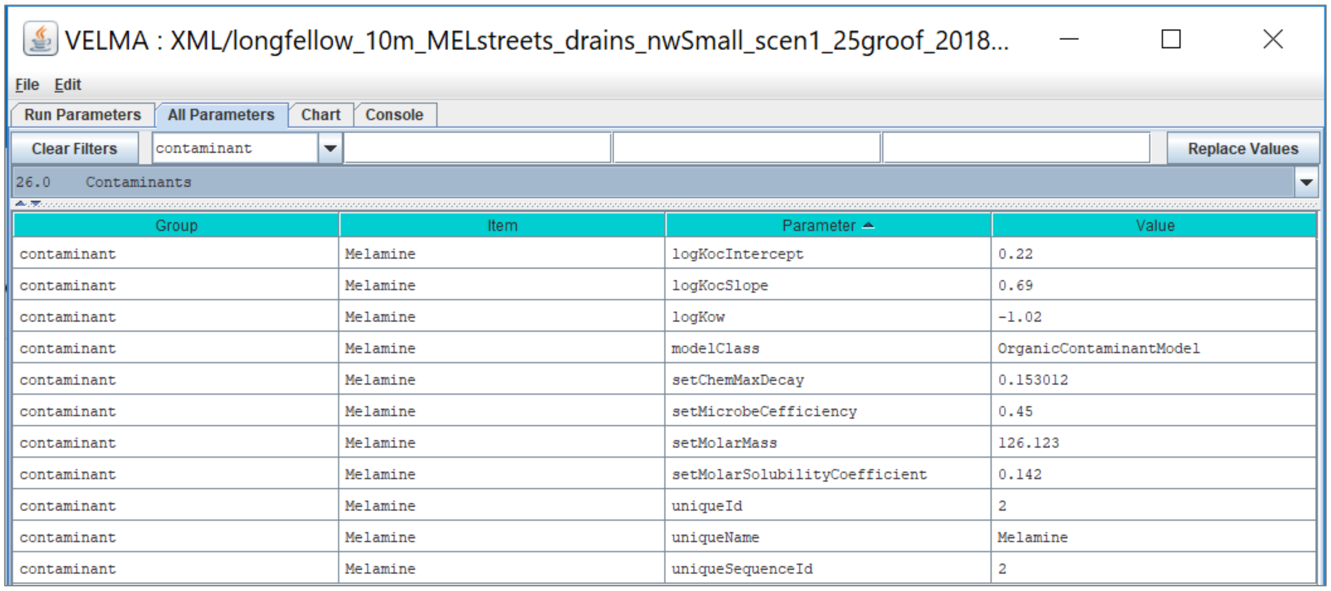
Contaminant Name: Melamine

Figure 4.

Clicking OK in the "Unique Contaminant Name" dialog adds a new parameterization group to the simulation configuration, and changes JVelma's display to the "All Parameters" tab, with the Item-level filter set to only display the parameters for the newly-added contaminant.

In the display below, click the "Parameter" column header to sort the parameters by name.

(continued)



Group	Item	Parameter	Value
contaminant	Melamine	logKocIntercept	0.22
contaminant	Melamine	logKocSlope	0.69
contaminant	Melamine	logKow	-1.02
contaminant	Melamine	modelClass	OrganicContaminantModel
contaminant	Melamine	setChemMaxDecay	0.153012
contaminant	Melamine	setMicrobeCefficiency	0.45
contaminant	Melamine	setMolarMass	126.123
contaminant	Melamine	setMolarSolubilityCoefficient	0.142
contaminant	Melamine	uniqueId	2
contaminant	Melamine	uniqueName	Melamine
contaminant	Melamine	uniqueSequenceId	2

Figure 5.

Note that when you first establish a contaminant Group/Item, all adjustable parameter values are initially highlighted in yellow, indicating that they are currently unset and need to have Values entered. In Figure 5, the user has left no adjustable parameters unset. Preset parameter values that are never highlighted and should never be adjusted. Preset parameters include "modelClass" and "uniqueName". In Figure 5, the user preset values for these parameters as "OrganicContaminantModel" and "Melamine", respectively, when using the "Edit/Contaminant/Add a New Contaminant" menu.

3.3 How to Set Contaminant Parameter Values

VELMA users can set contaminant parameter values using the graphical user interface shown in Figure 5. Sections 3.3.1 – 3.3.5 break out several categories of contaminant parameters shown in Figure 5. In these sections, VELMA users must enter a value for parameter names that are initially highlighted in yellow. Note that Parameter names are also presented in camelCase (https://en.wikipedia.org/wiki/Camel_case). Definitions, units and data sources are provided for each parameter. Users can find values for most contaminant parameters through searching publicly available sources, such as EPA's CompTox library or publications. If data are not available for a contaminant, users should consult with an expert to discover unpublished data, or to seek advice on the feasibility of using data for a biochemically similar contaminant.

3.3.1 Parameters for Sorbed-Aqueous Contaminant Partitioning

VELMA 2.1 users need to enter values for just three parameters that affect partitioning of a contaminant between sorbed (bound by soil) and aqueous (dissolved) phases within soil. These are listed in Figure 5 under the Parameter column: logKow, logKocIntercept and logKocSlope. Column 4 of Figure 5 shows where users enter values for these parameters for the contaminant specified. The following notes describe data sources for these parameters.

logK_{OW} = octanol-water partition coefficient, expressed as the logarithm of the experimentally determined ratio for chemical concentration in octanol phase / chemical concentration in aqueous phase. See Section 2.1 for additional details. Log K_{OW} values for most contaminants can be found in EPA's

Comptox library (https://comptox.epa.gov/dashboard/). **CAUTION:** Comptox generally refers to K_{OW} as K_P, an unfortunate break from established nomenclature. In VELMA, we follow established nomenclature, such as found in Piwoni and Keeley (1990), whereby K_{OW} is reserved for octanol-water partitioning, and K_P is reserved to address the important effect of soil organic matter content (f_{OC}) on contaminant sorption for a particular soil. See Figure 1 for additional details. Therefore, for a given contaminant, VELMA users will need to set the logKow parameter value (see Figure 5) to the K_P value in Comptox.

logKocIntercept = unitless slope intercept parameter for an empirically-derived linear regression describing the correlation of Log K_{OC} to K_{OW} for the contaminant of interest. Users need to search the literature for regression equations representative of the contaminants and soil characteristics of interest. See Figure 1 in Section 2.1 and Piwoni and Keeley (1990) for examples and further explanation.

logKocSlope = unitless slope parameter for an empirically-derived linear regression describing the correlation of Log K_{OC} to Log K_{OW} for the contaminant of interest. See Figure 1 and Piwoni and Keeley (1990) for an example. Depending upon the contaminant to be simulated, users may need to search the literature for a representative regression equation.

Reminder: VELMA internally recalculates all other partitioning parameters (f_{OC}, Log K_{OC}, K_P) described in Table 1, as a function of the preceding parameter values and simulated daily changes in soil water and humus content.

3.3.2 Contaminant Solubility Parameter

setMolarSolubilityCoefficient = solubility (mol/L) of a contaminant in water. Contaminant solubility values can be found in EPA's Comptox library (https://comptox.epa.gov/dashboard/).

setMolarMass = contaminant molecular weight (g/mol). Comptox provides the molecular weight for all contaminants in its library. Users need to enter this so that VELMA can calculate solubility

When the user provides these two parameters, VELMA calculates a *gramSolubilityCoefficent* value "behind the scenes" as:

*gramSolubilityCoefficient = molarMass * molSolubilityCoefficient*,
with units, g/L = g/mol * mol/L

All of which leads to the behind the scenes calculation for surface contaminant solubility and vertical transfer:

*totalChemTransferAmount = Math.min(totalWaterFlow * gramSolubilityCoefficient, totalChemAmount)*

3.3.3 Contaminant Decay Parameters

setMicrobeCefficiency = the unitless microbial efficiency (range [0.0, 1.0]) of degradation of organic contaminants in soil, that is, the ratio of carbon assimilated/total carbon consumed, whereby the difference is that respired as carbon dioxide. Typical microbial efficiencies range from 0.45 to 0.55.

setChemMaxDecay = the maximum daily decay rate constant (fraction per day) of a specified contaminant. The Comptox library (https://comptox.epa.gov/dashboard/) lists the half-life for most organic contaminants, from which the daily maximum decay rate constant can be calculated and specified for this parameter. Users can enter the specified half-life value into an online calculator to calculate the decay constant. For example, see https://www.calculator.net/half-life-calculator.html?type=2&chalflife=4&cmeanlifetime=&cdecayconstant=&x=35&y=16. Note that decomposed contaminant amounts are not tracked by the simulation and not reported in its results. This is because contaminants almost always make up an insignificant fraction of the total amount of organic matter within soils.

3.3.5 Parameters That Can Be Left Alone (purposely not yellow highlighted)

uniqueName = the parameter name that you specified while adding the contaminant parameterization to the configuration. After that initial specification, we recommend never changing this parameter's value, because it is used as part of the name for the contaminant's spatial data pools. If you do change a uniqueName's value, any disturbances referencing its spatial data pools must be updated to reference those pool names as well.

modelClass = parameter set by VELMA when you add the contaminant to the simulation configuration. Do Not Change this parameter's value. Ever.

uniqueId = parameter that VELMA generates automatically, and should be left as-is.

uniqueSequenceId = determines the sequence in which the contaminants are processed during each VELMA simulation step. It is provided for future, possible enhancements to the contaminants code that would let one contaminant interact with another. However, that functionality is currently unavailable, and the uniqueSequenceId values that JVelma generates for you can be left as-is.

4.0 Specifying the Names of a Contaminant's Spatial Data Pools

Spatial Data Writers are how a VELMA reports spatially-explicit simulation results. Users have the option of setting up Spatial Data Writers to report a simulation result for an output variable for every cell in the simulation grid. This feature is especially useful for displaying 2D or 3D maps in VISTAS (http://blogs.evergreen.edu/vistas/vistas-software/), ArcGIS, or other software equipped to display model output in a landscape-oriented format. For details, see Section 23, page 96, in McKane et al. (2014).

A given contaminant parameterization is associated with 2 types of spatial data pools:

- A single-layer Surface Pool
- Multi-layered Soil Pools (4 total)

Both pools store per-cell (and per-cell, per-layer for the layered pool) quantities of the contaminant itself.

Unlike VELMA's core chemistry pools, contaminant spatial data pool names are constructed using a common prefix and a unique suffix. To provide proper key-name for disturbances or spatial data writers parameterized to reference or modify contaminant pool values, you must know how to construct the specific pool name for a specific contaminant pool.

The contaminant's surface pool prefix is "CONTAMINANT_SURFACE_" and the suffix is the contaminant's uniqueName parameter value shifted to all-uppercase.

Example:
If uniqueName = "DDT", the surface pool key-name is: CONTAMINANT_SURFACE_DDT.

Similarly, the contaminant's layered pool prefix is "CONTAMINANT_LAYERS_" and the suffix is the uniqueName parameter value, again, all-uppercase.

Example:
With uniqueName = Glyphosate, the layered pool key-name is: CONTAMINANT_LAYERS_GLYPHOSATE.

5.0 Contaminant Deposition

Adding one or more OrganicContaminantModel parameterizations to a VELMA simulation configuration does not add any actual chemical amounts to the simulation run. To introduce amounts of the contaminant(s) into the contaminant pools, you’ll need to parameterize one or more disturbance events.

You can use Surface Deposition disturbances, or Set Spatial Data By Map disturbances (or one or more instances of both) to introduce contaminant amounts into the contaminant pools.

For general information on how to configure these disturbances in VELMA, see the following.

- Surface Deposition:
  Tutorial E.3 – Surface Chemical Deposition

- Set Spatial Data by Map:
  Tutorial D.3 – Initialize a Spatial Data Pool Using an Ascii Grid

The important difference between using the above disturbances with VELMA’s core chemistry pools and with contaminants is that you must provide the prefix+suffix contaminant pool names to the disturbance, as described in Section 4.0, “Specifying the Names of a Contaminant’s Spatial Data Pools”, rather than simply looking up the key-names in a table as you would for NO3-N, for example.

6.0 Simulation Results for Contaminants

When a VELMA simulation configuration contains one or more contaminants, an additional daily results file is written as part of the simulation’s results. The file is named DailyContaminantResults.csv, and contains daily-averaged values for each contaminant’s surface and layered pools as well as surface and layered watershed loss amounts (the amount calculated to have left the watershed via transfer from non-channel to channel cells).

Additionally, Cell Data Writers specified for a VELMA simulation configuration contain rows reporting the surface and layer amounts (g/m^2), and the surface and layer lateral-outflow amounts ($g\ m^{-2}\ d^{-1}$) for each contaminant in the simulation configuration.

Column names for contaminant results mimic (with slight variations) the prefix-suffix rules outlined in Section 4.0, “Specifying the Names of a Contaminant’s Spatial Data Pools”. Any contaminant results data column will begin with the keyword “CONTAMINANT_” (or “Contaminant_” in some cases). Surface-data columns follow CONTAMINANT_ with “SURFACE”, and finally, after these keywords, the uniqueName value for the specific CONTAMINANT appears.

7.0 Demonstration: VELMA Simulation of PCB and Melamine Fate and Transport in a Seattle Watershed

This section describes a VELMA simulation beta test that illustrates how multiple factors – hydrologic processes, contaminant solubility and partition coefficients (log Kow, an indicator of hydrophobicity) collectively affect fate and transport of two organic contaminants – melamine and PCB – within a subwatershed of Longfellow Creek in West Seattle, WA.

Melamine is present in tire residues and is a known constituent of urban stormwater runoff and possible contributor to prespawn mortality of coho salmon in Seattle area streams (Peter et al., 2018).

PCBs are persistent in some Seattle area brownfields and Superfund sites, as well as in the Puget Sound food web, including apex predators such as Chinook salmon and killer whales (Washington Department of Ecology, 2018).

Table 2. Biochemical properties that differentiate fate and transport behavior of melamine and PCB. Source of values shown: https://comptox.epa.gov/dashboard.

Contaminant	Solubility (mol/L)	log K_{OW} (unitless)	Maximum Decay Rate (fraction/day)
Melamine	0.142	-1.02	0.15
PCB 026	0.0000012	5.48	0.05

In summary, Table 2 shows that in comparison to melamine, PCB is about 10^6 times less soluble, and decays about 3 times more slowly.

Figure 6 illustrates how these biochemical properties differentiate the fate and transport behavior of melamine and PCB within a subwatershed of the Longfellow Creek watershed in West Seattle, WA.

(continued)

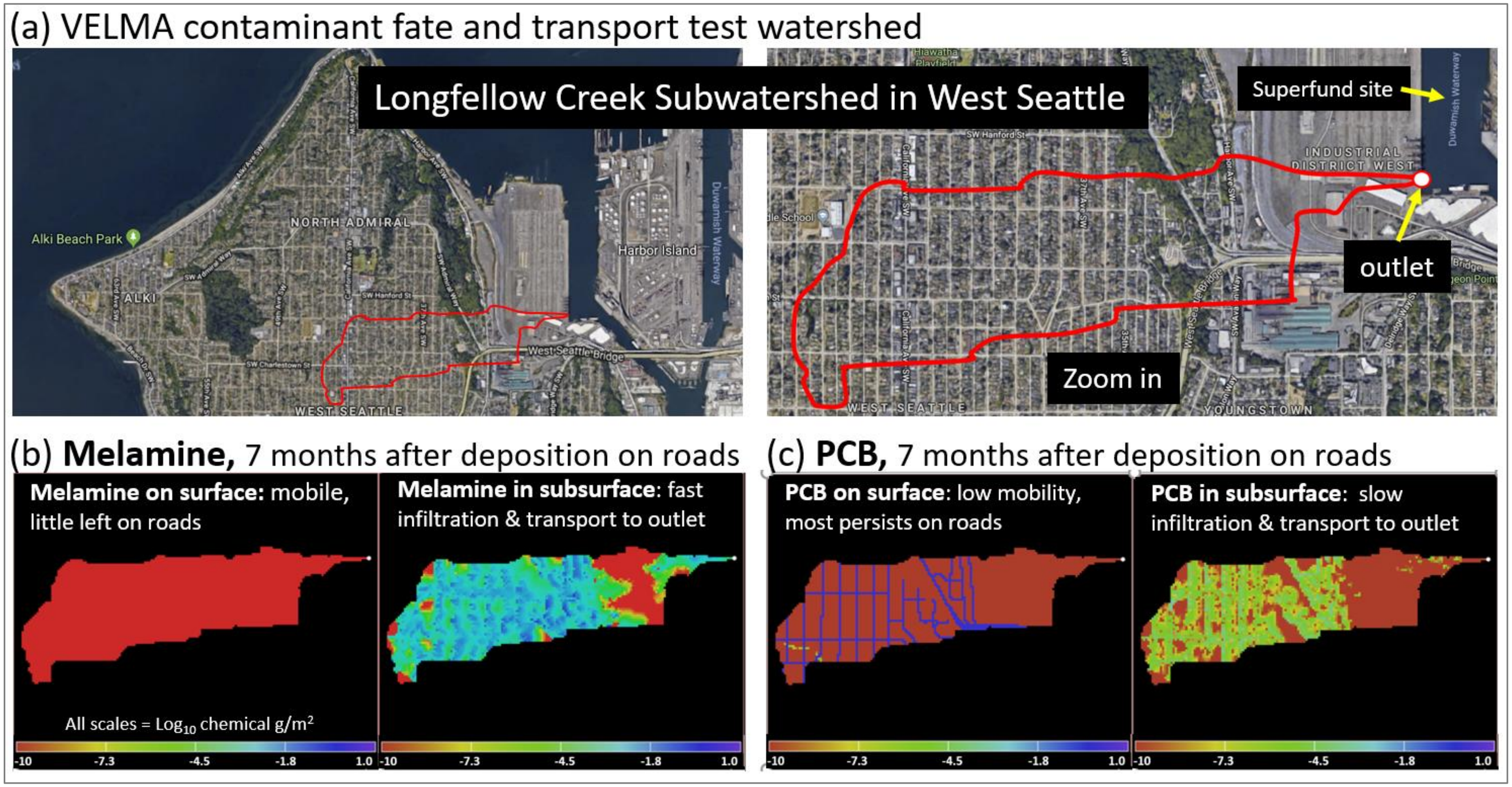

(a) VELMA contaminant fate and transport test watershed
Longfellow Creek Subwatershed in West Seattle
Superfund site
outlet
Zoom in
Alki Beach Park
NORTH ADMIRAL
ALKI
Harbor Island
Duwamish Waterway
West Seattle Bridge
WEST SEATTLE
INDUSTRIAL DISTRICT WEST
YOUNGSTOWN
(b) Melamine, 7 months after deposition on roads
Melamine on surface: mobile, little left on roads
Melamine in subsurface: fast infiltration & transport to outlet
All scales = Log10 chemical g/m2
(c) PCB, 7 months after deposition on roads
PCB on surface: low mobility, most persists on roads
PCB in subsurface: slow infiltration & transport to outlet
-10
-7.3
-4.5
-1.8
1.0

Figure 6. We conducted a beta test of VELMA 2.1 or a subwatershed of Longfellow Creek in West Seattle. This preliminary demonstration compares fate and transport of melamine and PCB. (a) The test subwatershed includes residential and industrial districts that drain to the outlet (white dot, upper right frame) in the Lower Duwamish Waterway Superfund site. This hypothetical simulation compares fate and transport of a one-time (June 1, 2011) deposition of 1 g/m^2 each of melamine and PCB onto all road surface VELMA cells (10 m^2) within the subwatershed boundary (red outline in frames (a) and (b)). Melamine is present in tire residues, and is a relatively soluble and hydrophilic (low K_{OW} and K_P sorption coefficients) constituent of urban stormwater runoff and possible contributor to prespawn mortality of coho in Seattle area streams (Peter et al. 2018). In contrast, PCBs are relatively insoluble, hydrophobic (high sorption coefficients) and persistent within soils, marine sediments and the Puget Sound food web, including endangered Chinook salmon and killer whales. Our VELMA test simulations results for December 31, 2011 (7 months after PCB and melamine deposition on road surfaces) illustrate how these contrasting chemical properties lead to (b) melamine's rapid vertical and lateral (west to east downslope gradient) transport toward to the outlet, leaving almost no residual chemical on the source roadways; and (c) PCB's persistence on road surfaces with up to six orders of magnitude less PCB in near-surface soil layers compared to melamine. For this demonstration, soil organic matter content did not vary across the watershed, and storm sewer drains and pipes were not simulated pending acquisition of these data sets.

8.0 References

Abdelnour, A., Stieglitz, M., Pan, F., & McKane, R. (2011). Catchment hydrological responses to forest harvest amount and spatial pattern. *Water Resources Research*, *47*(9).

Abdelnour, A., McKane, R. B., Stieglitz, M., Pan, F., & Cheng, Y. (2013). Effects of harvest on carbon and nitrogen dynamics in a Pacific Northwest forest catchment. *Water Resources Research*, *49*(3), 1292-1313.

Bahadur, R., Amstutz, D. E., & Samuels, W. B. (2013). Water contamination modeling—a review of the state of the science. *Journal of Water Resource and Protection*, *5*(02), 142.

El-Shahawi, M. S., Hamza, A., Bashammakh, A. S., & Al-Saggaf, W. T. (2010). An overview on the accumulation, distribution, transformations, toxicity and analytical methods for the monitoring of persistent organic pollutants. *Talanta*, *80*(5), 1587-1597.

Ho, Y. S., Chiu, W. T., & Wang, C. C. (2005). Regression analysis for the sorption isotherms of basic dyes on sugarcane dust. *Bioresource technology*, *96*(11), 1285-1291. https://pdfs.semanticscholar.org/982f/3099249a9e0cb60a46c3675110d28f56d42b.pdf.

McEachran, A. D., Sobus, J. R., & Williams, A. J. (2017). Identifying known unknowns using the US EPA’s CompTox Chemistry Dashboard. *Analytical and bioanalytical chemistry*, *409*(7), 1729-1735.

McKane, R.B., A. Brookes, K. Djang, M. Stieglitz, A. Abdelnour, F. Pan (2014). VELMA 2.0 User Manual and Technical Documentation. USEPA/ORD Report ORD-010080, Safe and Sustainable Waters Research Program, September 30, 2014. (Downloadable here: https://www.epa.gov/water-research/visualizing-ecosystem-land-management-assessments-velma-model-20.

Peter, K. T., Tian, Z., Wu, C., Lin, P., White, S., Du, B., ... & Kolodziej, E. P. (2018). Using High-Resolution Mass Spectrometry to Identify Organic Contaminants Linked to Urban Stormwater Mortality Syndrome in Coho Salmon. *Environmental science & technology*, *52*(18), 10317-10327.

Piwoni, M. D. and Banerjee, P. (1989). Sorption of volatile organic solvents from aqueous solution onto subsurface solids. *Journal of Contaminant Hydrology*, *4*(2), 163-179.

Piwoni, M. D., & Keeley, J. W. (1990). Basic concepts of contaminant sorption at hazardous waste sites. US Environmental Protection Agency, Superfund Technology Support Center for Ground Water, Robert S. Kerr Environmental Research Laboratory, Ada, OK. Report Number: EPA/540/4-90/053. https://nepis.epa.gov/Exe/ZyPDF.cgi/2000L079.PDF?Dockey=2000L079.PDF.

Potter, C. S., Randerson, J. T., Field, C. B., Matson, P. A., Vitousek, P. M., Mooney, H. A., & Klooster, S. A. (1993). Terrestrial ecosystem production: a process model based on global satellite and surface data. *Global Biogeochemical Cycles*, 7(4), 811-841.

Washington Department of Ecology. 2018. Toxic chemicals in Puget Sound. Link: https://ecology.wa.gov/Water-Shorelines/Puget-Sound/Issues-problems/Toxic-chemicals. AND, University of Washington Tacoma Puget Sound Institute, https://www.eopugetsound.org/terms/407.

Williams, A. J., Grulke, C. M., Edwards, J., McEachran, A. D., Mansouri, K., Baker, N. C., ... & Richard, A. M. (2017). The CompTox Chemistry Dashboard: a community data resource for environmental chemistry. *Journal of cheminformatics*, *9*(1), 61.

A.2 | Configure VELMA for Simulating Contaminant Fate and Transport

Overview *(Tutorial A.2 – Configure VELMA for Simulating Contaminant Fate and Transport)*

This document is a broad overview of how to configure VELMA 2.1 to model fate and transport of organic contaminants within watersheds. Summaries and instructions are provided for

- Configuration Steps for A VELMA Contaminant Simulation Run
- Adding a Contaminant Parameterization to a VELMA Configuration
- Parameters That Should Be Left Alone
- Decay Parameters
- Parameters for Sorbed/Aqueous Partitioning
- Specifying the Names of a Contaminant’s Spatial Data Pools
- Contaminant Deposition
- Simulation Results for Contaminants

This Tutorial document is a companion to the two listed below:

- Tutorial A.1_HowTo_VELMA Contaminant Modeling Conceptual Framework
- Tutorial A.3 – VELMA Contaminant Modeling: Longfellow Creek Example

1.0 Introduction

VELMA supports the addition of one or more Organic Contaminants to a simulation configuration. Each contaminant is added as its own set of parameters, and models one specific organic contaminant (e.g., DDT) during a simulation run.

Organic contaminants added to a VELMA simulation configuration undergo degradation and lateral and vertical transport during each simulation step. No other processes occur for contaminants.

Currently, only one type of degradation is available, based on equations developed by C.S. Potter, et. al, and parameterized by microbe efficiency and a maximum decay factor.

Unlike VELMA’s core chemistry pools (NH4-N, NO3-N, etc.), there is no global, initial amount of contaminant added to the simulation state at simulation start.

Configuring one or more contaminant parameterizations requires configuration of an equal number (or more) of VELMA disturbance parameterizations to introduce actual amounts of the contaminant into the simulation state. VELMA disturbances can be configured to introduce specific amounts of contaminant to specific locations of the simulation watershed and arbitrary steps during the simulation run.

Each parameterized contaminant adds its own surface and layered spatial data pools, and loss-tracking temporal data arrays to the simulation state at runtime. Spatial Data Writers may be configured for contaminant pools, but the pool key-name provided to the Spatial Data Writer parameterization is derived from a common root and the contaminant-specific parameterization pool names – this is unlike any other spatial data pool naming currently employed by the VELMA Simulation Application.

2.0 Overview: Configuration Steps for **a** VELMA Contaminant Simulation Run

Start with a proven VELMA simulation configuration – i.e., a simulation configuration .xml file that is already calibrated and known to run reliably and accurately for the study site.

Load that simulation configuration .xml file into JVelma, and immediately change its "Simulation Run Name" (i.e. the run_index parameter's value), then save it. This creates a copy of the original .xml that you can modify with contaminant additions, while saving the original .xml for single-process use.

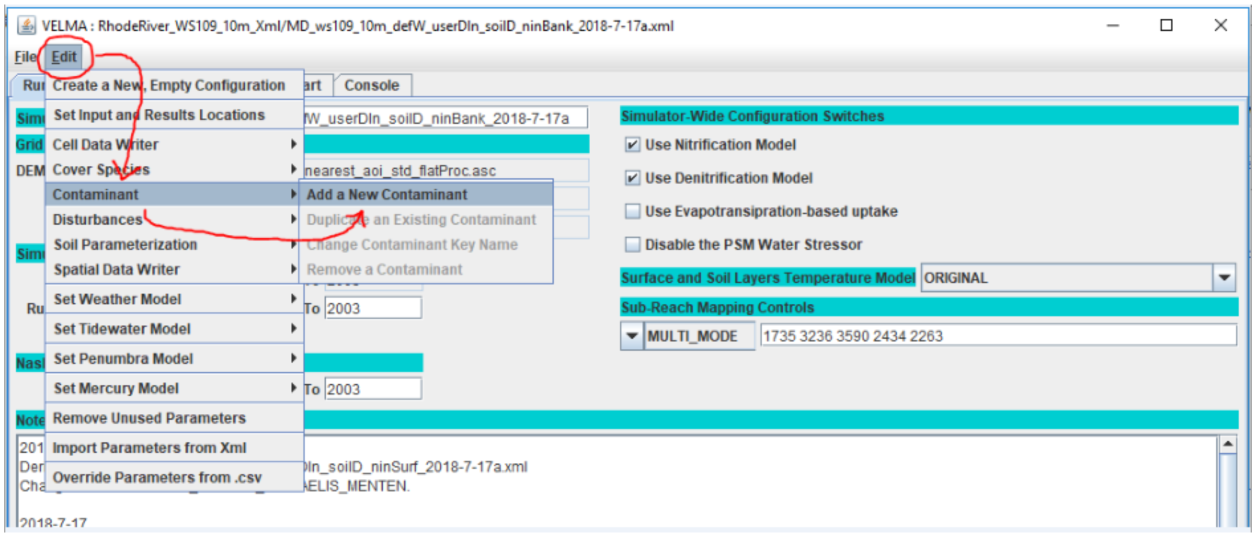
With the new copy of the proven simulation configuration loaded into JVelma, adding one or more contaminants involves the following steps for each contaminant you wish to add:

6. Add a new Contaminant parameterization group to the simulation configuration.
7. Add one or more Disturbance parameterization groups to the simulation configuration to introduce amounts of the contaminant to the simulation run and specific times and locations.
8. (Optional) Add one or more Spatial Data Writer parameterization groups to produce spatially-explicit map results for the contaminant.
   Note: The DailyResults and any configuration-specified Cell Data Writer .csv files automatically include columns for any contaminants added to simulation configuration. Only Spatial Data Writers must be explicitly parameterized.

Save the changes – the simulation configuration .xml is now ready for use with JVelma.

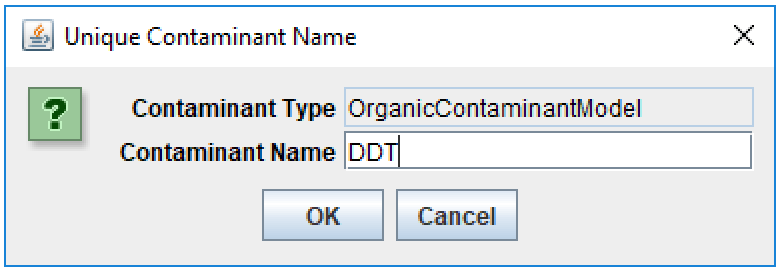
3.0 Adding a Contaminant Parameterization to a VELMA Configuration

With a proven VELMA simulation configuration loaded into JVelma,
click the Edit -> Contaminants -> "Add a New Contaminant" menu item, as shown below:


VELMA : RhodeRiver_WS109_10m_Xml/MD_ws109_10m_defW_userDIn_soilD_ninBank_2018-7-17a.xml

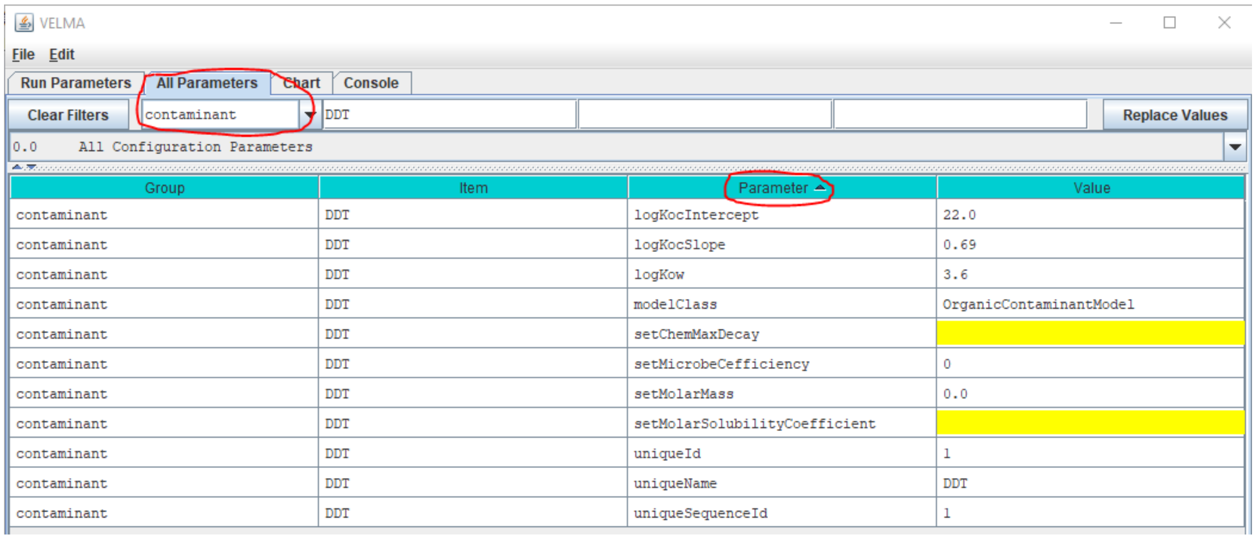
File
Edit
Create a New, Empty Configuration
Set Input and Results Locations
Cell Data Writer
Cover Species
Contaminant
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Set Penumbra Model
Set Mercury Model
Remove Unused Parameters
Import Parameters from Xml
Override Parameters from .csv
Add a New Contaminant
Duplicate an Existing Contaminant
Change Contaminant Key Name
Remove a Contaminant
Console
Simulator-Wide Configuration Switches
Use Nitrification Model
Use Denitrification Model
Use Evapotranspiration-based uptake
Disable the PSM Water Stressor
Surface and Soil Layers Temperature Model
ORIGINAL
Sub-Reach Mapping Controls
MULTI_MODE
1735 3236 3590 2434 2263
To 2003
To 2003

In the “Unique Contaminant Name” pop-up dialog that opens next, provide a name for the contaminant. The name should begin with an alphabet character, and may consist of only alpha-numeric and underscore (“_”) characters, and must remain unique among all contaminants in the configuration.



Contaminant Type	OrganicContaminantModel
Contaminant Name	DDT

Click OK in the “Unique Contaminant Name” dialog adds a new parameterization group to the simulation configuration, and changes JVelma’s display to the “All Parameters” tab, with the Item-level filter set to only display the parameters for the newly-added contaminant.
(Click the “Parameter” column header to sort the parameters by name)



Group	Item	Parameter	Value
contaminant	DDT	logKocIntercept	22.0
contaminant	DDT	logKocSlope	0.69
contaminant	DDT	logKow	3.6
contaminant	DDT	modelClass	OrganicContaminantModel
contaminant	DDT	setChemMaxDecay	
contaminant	DDT	setMicrobeCefficiency	0
contaminant	DDT	setMolarMass	0.0
contaminant	DDT	setMolarSolubilityCoefficient	
contaminant	DDT	uniqueId	1
contaminant	DDT	uniqueName	DDT
contaminant	DDT	uniqueSequenceId	1

Some of the parameters have default values already set for them (although it is likely you will want to change them for the specific contaminant you are adding), some are set and should not be changed, and some are highlighted yellow, indicating that they are currently unset, and need to have values provided for them.

4.0 Parameters That Must Be Left Alone

The modelClass parameter is set by JVelma when you add the contaminant to the simulation configuration. Do *Not* Change this parameter’s value. *Ever*.

The uniqueName parameter is set to the name you specified while adding the contaminant parameterization to the configuration. After that initial specification, we recommend never changing this parameter’s value, because it is used as part of the name for the contaminant’s spatial data pools.
If you do change uniqueName’s value, any disturbances referencing its spatial data pools must update those pool names as well.

The uniqueId parameter was generated automatically and should be left as-is.

The uniqueSequenceId determines the sequence in which the contaminants are processed during each VELMA simulation step. It is provided for future, possible enhancements to the contaminants code that would let one contaminant interact with another. However, that functionality is currently unavailable, and the uniqueSequenceId values that JVelma generates for you can be left as-is.

5.0 Decay Parameters

The setMicrobeCefficiency parameter is the fractional (range [0.0, 1.0]) efficiency of microbe activity upon this organic contaminant.

setChemMaxDecay is the maximum daily decay rate constant (fraction per day) of this contaminant. Note: Comptox (https://comptox.epa.gov) et. al list the half-life for most organic contaminants, and the value for the setChemMaxDecay parameter can be calculated from the half-life. Various calculators are available on the web, for example: https://www.calculator.net/half-life-calculator.html

6.0 Parameters for Sorbed/Aqueous Partitioning

logKocIntercept (unitless) is the intercept parameter for the linear-regression equation describing the correlation of LogKoc to LogKow for this contaminant.

logKocSlope (unitless) is the slope parameter for the linear-regression equation describing the correlation of LogKox to LogKow for this contaminant.

logKow (units of log10 Kow) is the Octanol-water partition coefficient for this contaminant.

setMolarMass is this contaminant's mass, in grams per mol (g/mol).

setMolarSolubilityCoefficient is the solubility coefficient of this chemical in mol per liter (mol/L).

7.0 Specifying the Names of a Contaminant's Spatial Data Pools

A given contaminant parameterization is associated with 2 spatial data pools:

- A single-layer Surface Pool
- A multiple-layer "Layered" Pool

Both pools store per-cell (and per-cell, per-layer for the layered pool) quantities of the contaminant itself.

Unlike VELMA's core chemistry pools, contaminant spatial data pool names are constructed using a common prefix and a unique suffix. To provide proper key-name for disturbances or spatial data writers parameterized to reference or modify contaminant pool values, you must know how to construct the specific pool name for a specific contaminant pool.

The contaminant's surface pool prefix is "CONTAMINANT_SURFACE_" and the suffix is the contaminant's uniqueName parameter value shifted to all-uppercase.
Example:
If uniqueName = "Glysophate", the surface pool key-name is:
CONTAMINANT_SURFACE_GLYSOPHATE.

Similarly, the contaminant's layered pool prefix is "CONTAMINANT_LAYERS_" and the suffix is the uniqueName parameter value, again, all-uppercase.

Example:
With uniqueName=DDT, the layered pool key-name is: CONTAMINANT_LAYERS_DDT.

8.0 Contaminant Deposition

Adding one or more OrganicContaminantModel parameterizations to a VELMA simulation configuration does not add any actual chemical amounts to the simulation run. To introduce amounts of the contaminant(s) into the contaminant pools, you'll need to parameterize one or more disturbance events.

You can use Surface Deposition disturbances or Set Spatial Data By Map disturbances (or one or more instances of both) to introduce contaminant amounts into the contaminant pools.

For general information on how to configure these disturbances, see the following

- Surface Deposition:
  Tutorial E.3 – Surface Chemical Deposition

- Set Spatial Data by Map:
  Tutorial D.4 – Create Spatial Soil and Plant Chemistry Pools

The important difference between using the above disturbances with VELMA's core chemistry pools and with contaminants is that you must provide the prefix+suffix contaminant pool names to the disturbance, as described in the section "Specifying the Names of a Contaminant's Spatial Data Pools" above, rather than simply looking up the key-names in a table as you would for (example) NO3.

9.0 Simulation Results for Contaminants

When a VELMA simulation configuration contains one or more contaminants, an additional daily results file is written as part of the simulation's results. The file is named DailyContaminantResults.csv and contains daily-averaged values for each contaminant's surface and layered pools as well as surface and layered watershed loss amounts (the amount calculated to have left the watershed via transfer from non-channel to channel cells).

Additionally, Cell Data Writers specified for a VELMA simulation configuration contain rows reporting the surface and layer amounts, and the surface and layer lateral-outflow amounts for each contaminant in the simulation configuration.

Column names for contaminant results mimic (with slight variations) the prefix-suffix rules outlined in the "Specifying the Names of a Contaminant's Spatial Data Pools" section earlier in this document:
Any contaminant results data column will begin with the keyword "CONTAMINANT_" (or "Contaminant_" in some cases). Surface-data columns follow CONTAMINANT_ with "SURFACE", and finally, after these keywords, the uniqueName value for the specific CONTAMINANT appears.

A.3 | VELMA Contaminant Modeling Demonstration: Longfellow Creek Example

Overview *(Tutorial A.3 – VELMA Contaminant Modeling Demo: Longfellow Creek Example)*

This document is a detailed step-by-step tutorial for configuring VELMA 2.1 to simulate contaminant fate and transport within a watershed. The tutorial example used is a subwatershed of Longfellow Creek, a mixed-use urban watershed draining in Seattle, WA.

This A.3 contaminant modeling tutorial is a companion to the two listed below:

- Tutorial A.1_HowTo_VELMA Contaminant Modeling Conceptual Framework
- Tutorial A.2 – Configure VELMA for Simulating Contaminant Fate and Transport

This tutorial provides instructions for running a specific example scenario using VELMA 2.1's graphical user interface (GUI).

Through that process scenario it introduces VELMA 2.1 and some of its capabilities. It is *not* a comprehensive guide to VELMA 2.1. Refer to the documentation (McKane et al. 2014) and the VELMA 2.1 tutorials enclosed here for further information.

Requirements and Preliminaries

Section I Topics:

- Hardware Requirements
- Running PowerShell
- Checking for Java 8
- Installing Java 8
- VELMA .jar Files

(If you are already familiar with these topics and confident your computer meets their requirements, you may skip ahead to Section II.)

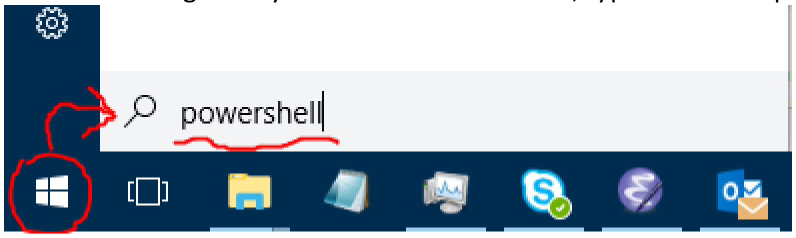
Hardware Requirements: You will need a computer with the following

- Windows 10 as its operating system.
- 4 Gigabytes ("GB") of RAM at a minimum (8 GB is preferred, more than that even better).
- Approximately 20 GB of disk space available for used.
  (A little less may still work, and more cannot hurt.)

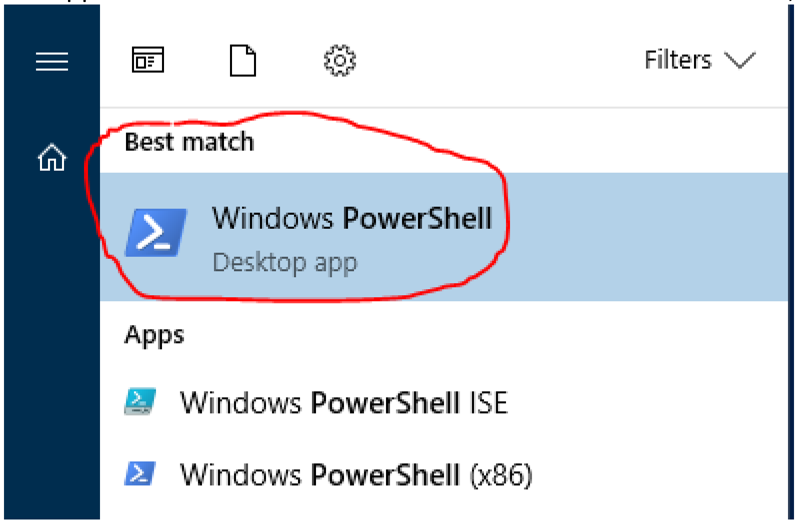
Running PowerShell: You must be able to access and run a Windows PowerShell

1. Click the Windows Start Menu button ⊞ to open the Start Menu.

2. Without clicking on any items in the Start Menu, type the word "powershell".

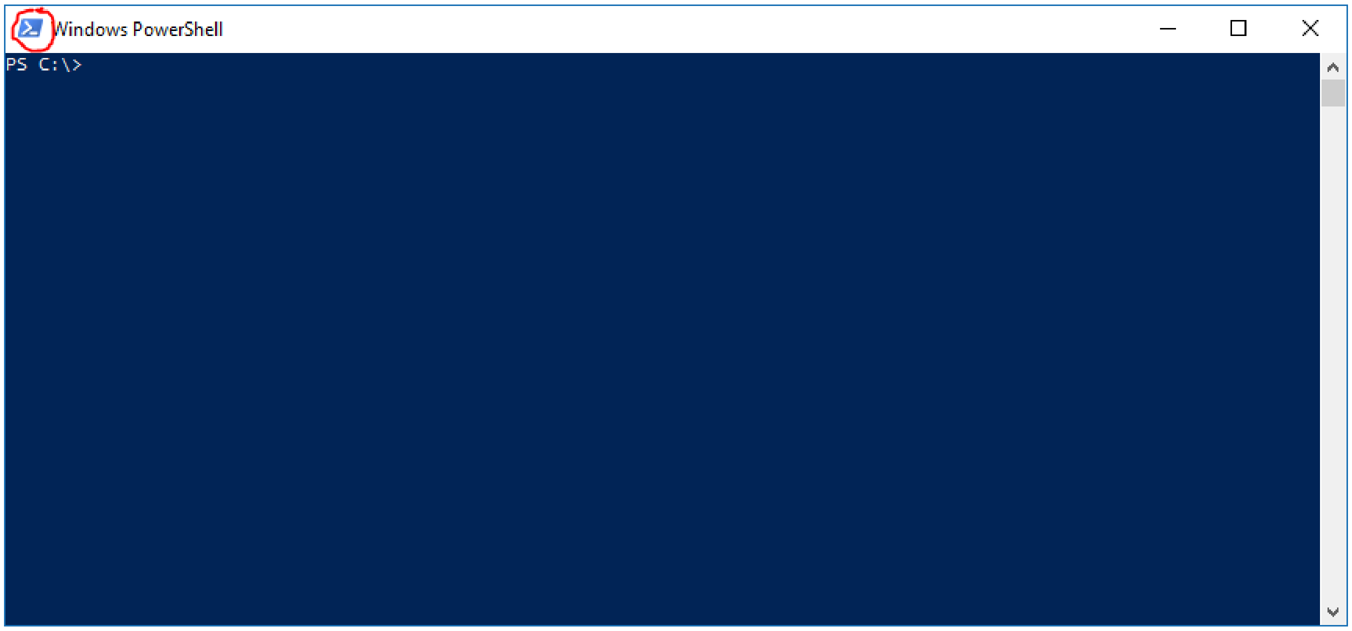


3. As you type, the Start Menu should automatically search for powershell-related items on your computer. If Powershell is available (and it should be on any standard Windows 10 computer) then it will appear as one of the items the Start Menu's search results list, and likely as the "Best match" item:



(If no item named "Windows PowerShell" or "Windows PowerShell (x86)" appears in the Start Menu search results, you do not have access to PowerShell. You will need to install it or gain access to it before proceeding. However, Windows PowerShell is part of the standard Windows 10 distribution, and *should* be available.)

4. Start a Windows PowerShell console by clicking on the "Windows PowerShell" item in the Start Menu search results list.
A console window should open like this:

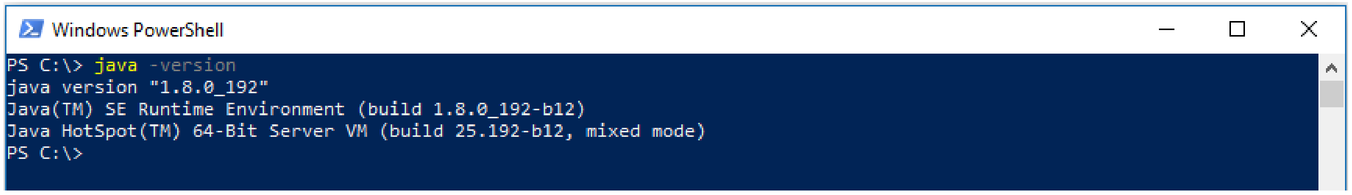


```
PS C:\>
```

(Your console window may have different dimensions, colors, and fonts than the example above. You can customize all of these settings by right-clicking the PowerShell icon – circled in red above.)

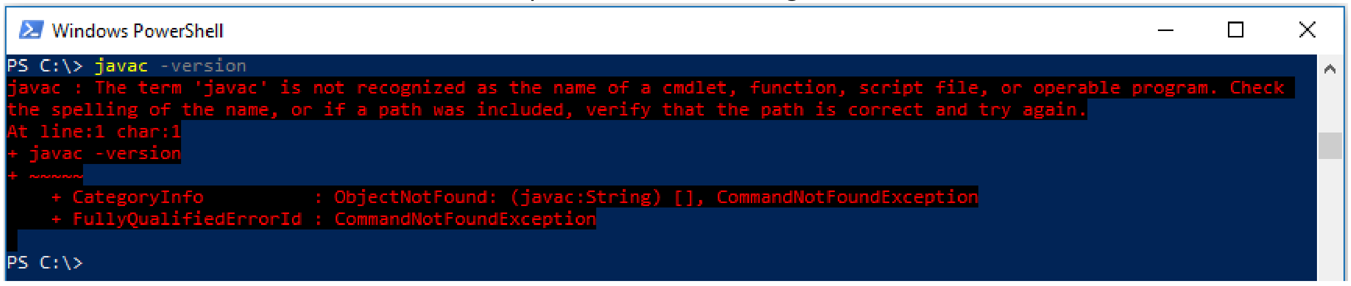
Checking for Java 8: You must have a Java JRE, version 8 or later, installed on the computer

1. Verify whether JRE installation and version using the PowerShell window you started above. (Or, open a new PowerShell window if you have already closed it.)
   At the PowerShell window's prompt, type the following command: java -version
2. If you have a Java JRE installed, you should see something like this:



```
PS C:\> java -version
java version "1.8.0_192"
Java(TM) SE Runtime Environment (build 1.8.0_192-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.192-b12, mixed mode)
PS C:\>
```

   If the version begins with "1.8" or a higher number (e.g. "1.9") you have an acceptable Java JRE installed and available.
   However, if the version begins with "1.7" or a lower number, you will need to install a newer, more-recent Java JRE.
3. If no Java JRE is installed or accessible, then you will see something like this:

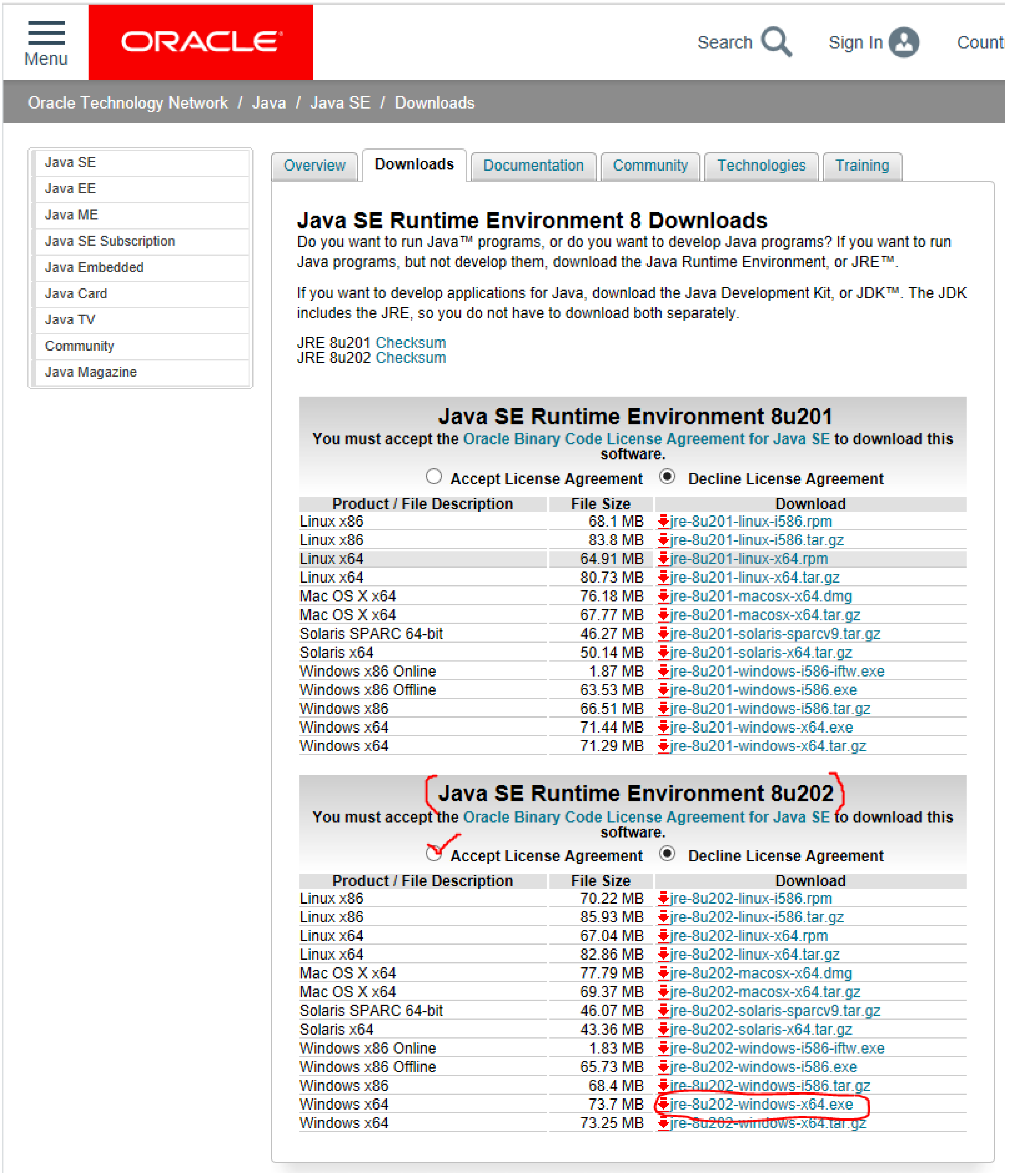


```
PS C:\> javac -version
javac : The term 'javac' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ javac -version
+ ~~~~~
    + CategoryInfo          : ObjectNotFound: (javac:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\>
```

(Note: the example above used "javac" to trigger a "not found" message on a computer that actually has a Java JRE installed. On a computer without a Java JRE, typing "java" would have had the same-looking result.)

Installing Java 8: How to install it onto your computer

- Go to the following Oracle company webpage: https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html
  The page looks like this: (see next page)



Java SE Runtime Environment 8 Downloads

Do you want to run Java™ programs, or do you want to develop Java programs? If you want to run Java programs, but not develop them, download the Java Runtime Environment, or JRE™.

If you want to develop applications for Java, download the Java Development Kit, or JDK™. The JDK includes the JRE, so you do not have to download both separately.

JRE 8u201 Checksum
JRE 8u202 Checksum

Java SE Runtime Environment 8u201

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ Accept License Agreement ● Decline License Agreement

Product / File Description	File Size	Download
Linux x86	68.1 MB	jre-8u201-linux-i586.rpm
Linux x86	83.8 MB	jre-8u201-linux-i586.tar.gz
Linux x64	64.91 MB	jre-8u201-linux-x64.rpm
Linux x64	80.73 MB	jre-8u201-linux-x64.tar.gz
Mac OS X x64	76.18 MB	jre-8u201-macosx-x64.dmg
Mac OS X x64	67.77 MB	jre-8u201-macosx-x64.tar.gz
Solaris SPARC 64-bit	46.27 MB	jre-8u201-solaris-sparcv9.tar.gz
Solaris x64	50.14 MB	jre-8u201-solaris-x64.tar.gz
Windows x86 Online	1.87 MB	jre-8u201-windows-i586-iftw.exe
Windows x86 Offline	63.53 MB	jre-8u201-windows-i586.exe
Windows x86	66.51 MB	jre-8u201-windows-i586.tar.gz
Windows x64	71.44 MB	jre-8u201-windows-x64.exe
Windows x64	71.29 MB	jre-8u201-windows-x64.tar.gz

Java SE Runtime Environment 8u202

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

[✓ Accept License Agreement] ● Decline License Agreement

Product / File Description	File Size	Download
Linux x86	70.22 MB	jre-8u202-linux-i586.rpm
Linux x86	85.93 MB	jre-8u202-linux-i586.tar.gz
Linux x64	67.04 MB	jre-8u202-linux-x64.rpm
Linux x64	82.86 MB	jre-8u202-linux-x64.tar.gz
Mac OS X x64	77.79 MB	jre-8u202-macosx-x64.dmg
Mac OS X x64	69.37 MB	jre-8u202-macosx-x64.tar.gz
Solaris SPARC 64-bit	46.07 MB	jre-8u202-solaris-sparcv9.tar.gz
Solaris x64	43.36 MB	jre-8u202-solaris-x64.tar.gz
Windows x86 Online	1.83 MB	jre-8u202-windows-i586-iftw.exe
Windows x86 Offline	65.73 MB	jre-8u202-windows-i586.exe
Windows x86	68.4 MB	jre-8u202-windows-i586.tar.gz
Windows x64	73.7 MB	jre-8u202-windows-x64.exe
Windows x64	73.25 MB	jre-8u202-windows-x64.tar.gz

In the “Java SE Runtime Environment 8u202” section of the page, click “Accept License Agreement”, then click the Windows x64 installer link: “jre-8u202-windows-x64.exe”.
After downloading the .exe file to your computer, double-click it to run the installer, and follow the installer instructions.

Note: You must have VELMA’s .jar files installed and available on your computer at this point.

Starting the VELMA GUI

Section II Topics:

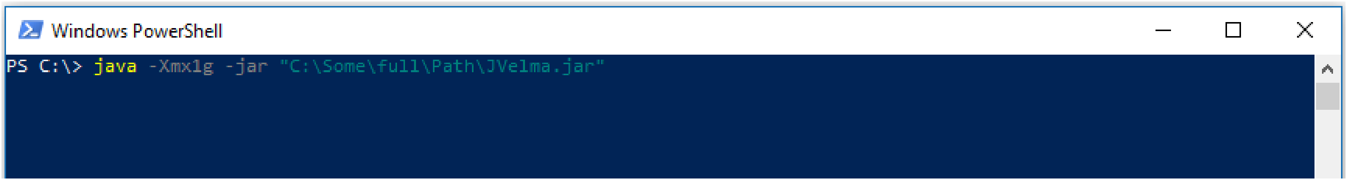
- Starting the VELMA GUI
- Overview the VELMA GUI's Layout

The VELMA GUI is named JVelma.jar. You can start the GUI by double-clicking the JVelma.jar file from a File Explorer window, but we do not recommend this method because VELMA will likely run out of memory during a simulation. Double-click starting the GUI allocates a default (and relatively small) amount of memory to the process – it's usually not enough. To explicitly specify the amount of memory the GUI runs with, follow the steps below:

Open a PowerShell window (as described in part I of this guide) and type the following startup command at the window's prompt:

```
java -Xmx1g -jar “C:\Some\full\Path\JVelma.jar”
```

Here's an example:



```
PS C:\> java -Xmx1g -jar "C:\Some\full\Path\JVelma.jar"
```

(If –but only if-- your path contains NO whitespace, you can omit the double-quotes" – see Appendix 1 for further discussion as well as other tips.)

Here's what each part of the command means:

1. `java` invokes the Java JRE, which in turn will start and run the GUI.

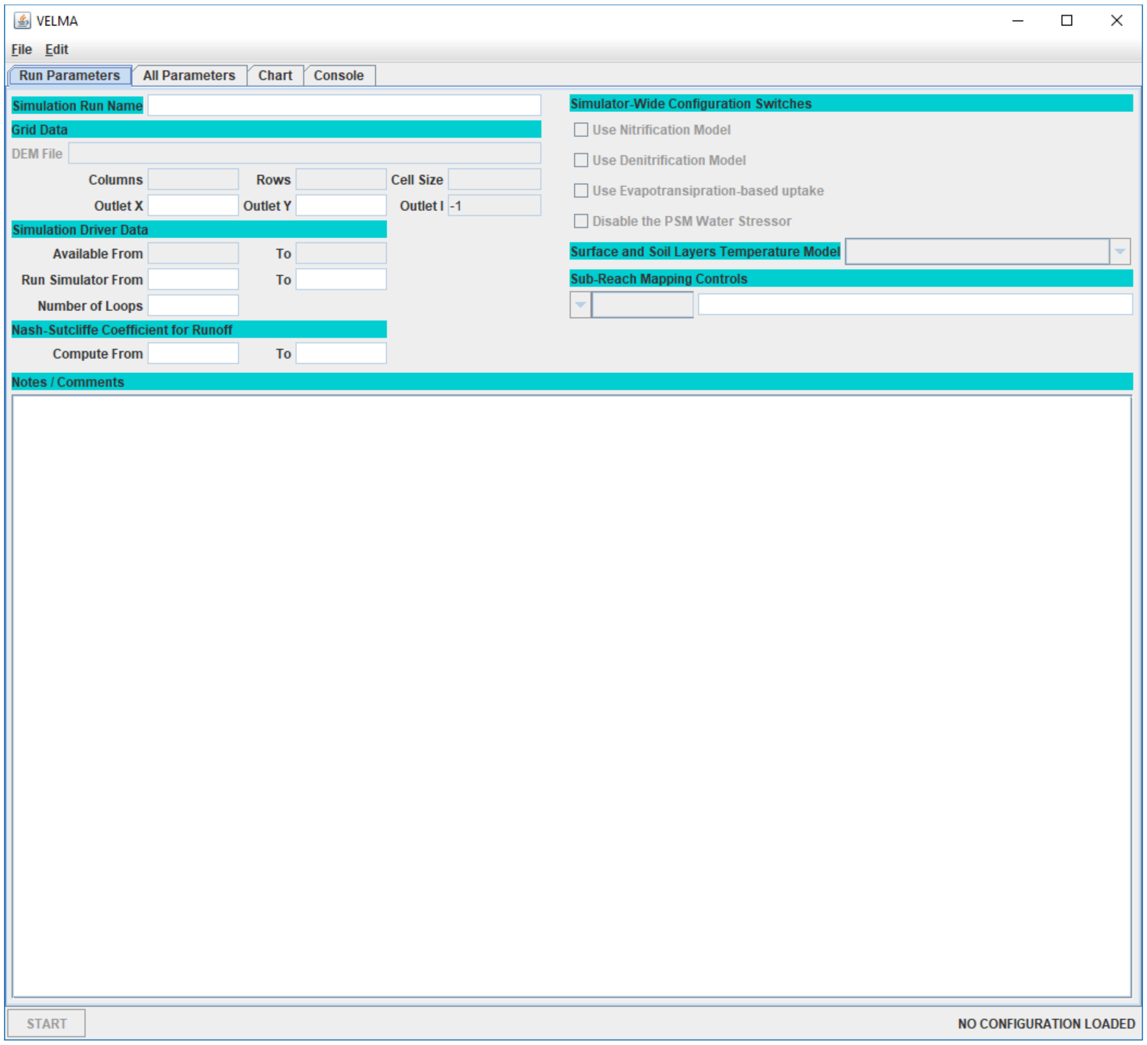
2. -Xmx1g allocates 1 Gigabyte of memory to the GUI/simulator.
   The "-Xmx" part indicates "memory allocation" do the Java JRE, the "1" is how much, and the "g" means "in Gigabytes". You can allocate more memory by increasing the numeral (e.g. from 1 to 4) but if you allocate more memory than your computer can make available, JVelma will fail to start properly.

3. `-jar` indicates to the java JRE that the next argument on the command line is the .jar file to run (and the directory path of its location).

4. `C:\Some\full\Path\JVelma.jar` is the "fully qualified" directory path and name of the JVelma.jar file – i.e. the VELMA GUI.
   Of course, this example path should be replaced by the actual, fully qualified path to the location of JVelma.jar on your computer.
   "Fully qualified" means that the path starts at a disk drive letter, and includes all directory names

(separated by backslashes) down to the JVelma.jar file itself.

If the JVelma GUI successfully loads and starts, you will see the screen below:



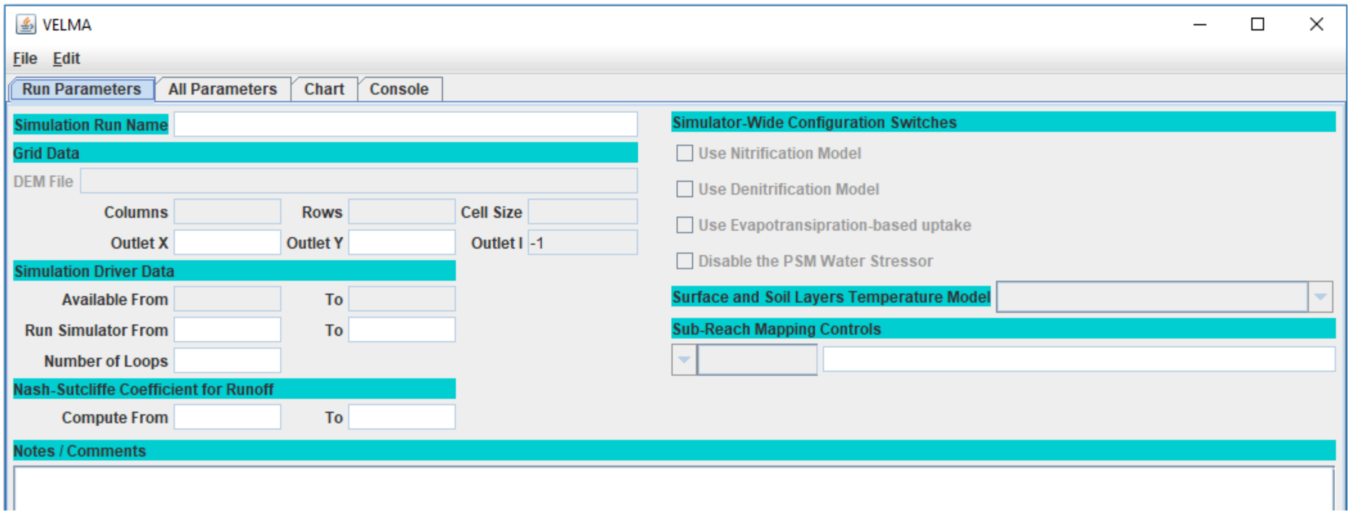
Note the "NO CONFIGURATION LOADED" message in the lower-right corner of the GUI's panel.

This means you are ready to load a VELMA simulation configuration and run it.

Here is a brief overview of the GUIs layout and features:

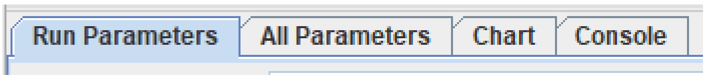
(If you closed the GUI after performing the steps above, re-open it by following those steps.)

When VELMA starts, you see the blank GUI and the menus and tabs used for configuring and running simulations. The image below shows the top portion of the main GUI panel:



Key Features of the main GUI panel:

- Two drop-down menus:

   1. The "File" menu contains items for loading and saving simulation scenarios.
      (It also contains an "About" item which reports VELMA's version info.)
   2. The "Edit" menu contains various items for creating and modifying a simulation scenario's configuration / parameters.
- Four Tabs: 

each tab switches the GUI to display a different panel:
   1. **Run Parameters**: a high-level summary of the scenario.
      Some core parameters and their values are accessible here for convenience.
      The Notes / Comments section displays any notes about the scenario you wish to record.
      The Run Parameters tab is initially visible when the VELMA GUI is started.
   2. **All Parameters**: the complete list of all the scenario's parameters and their values, along with a set of filter controls that can be used to narrow the set of visible parameters.
   3. **Chart**: during simulation runs, this panel displays user-selectable graphs of the simulator's runtime behavior.
      When the GUI isn't running a simulation, this panel displays a "Currently Unavailable" message.
   4. **Console**: during simulation runs, this panel displays status messages that the simulation engine would have otherwise written to the PowerShell window. Use the messages in this panel to

troubleshoot model crashes.
When the GUI isn't running a simulation, this panel is mostly blank.

Loading and Running an Example VELMA Simulation

Section III Topics:

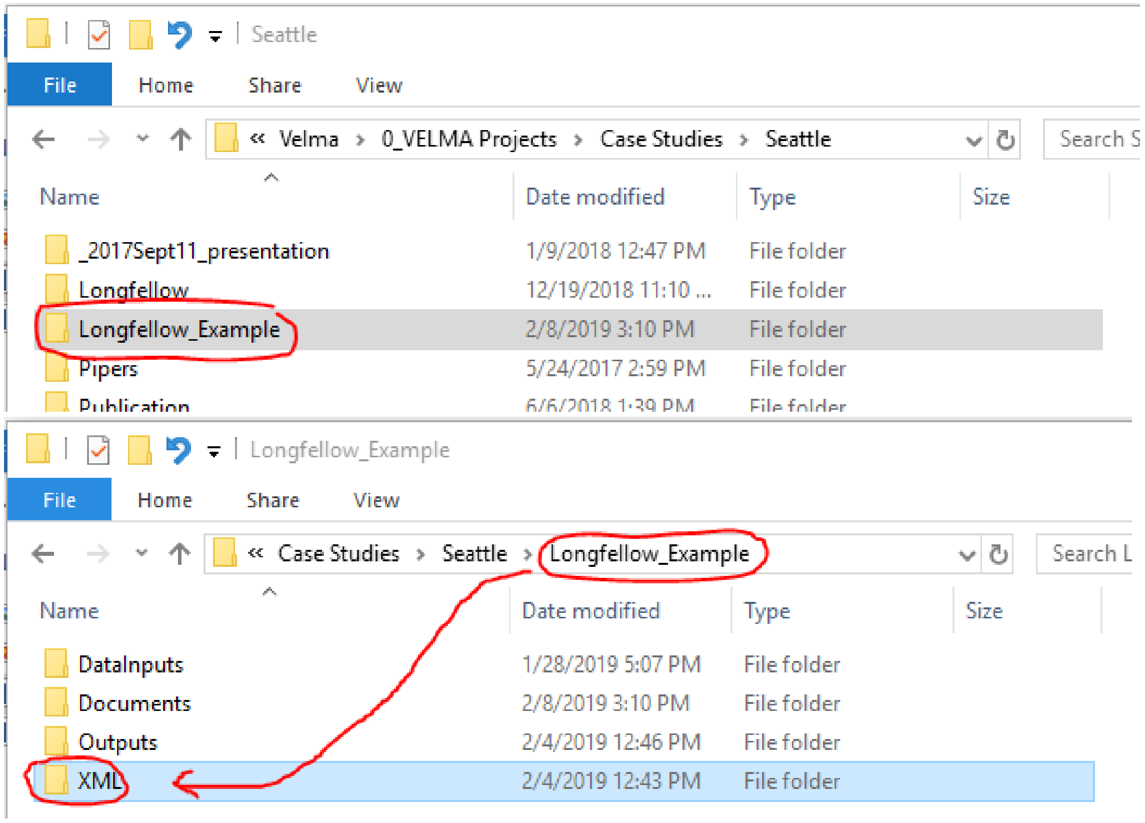
- Running a VELMA Simulation
- Monitoring a running VELMA Simulation

The Longfellow watershed example data contains several pre-configured example VELMA simulation scenarios. You can load and run them using the VELMA GUI.

Running a VELMA simulation from the GUI

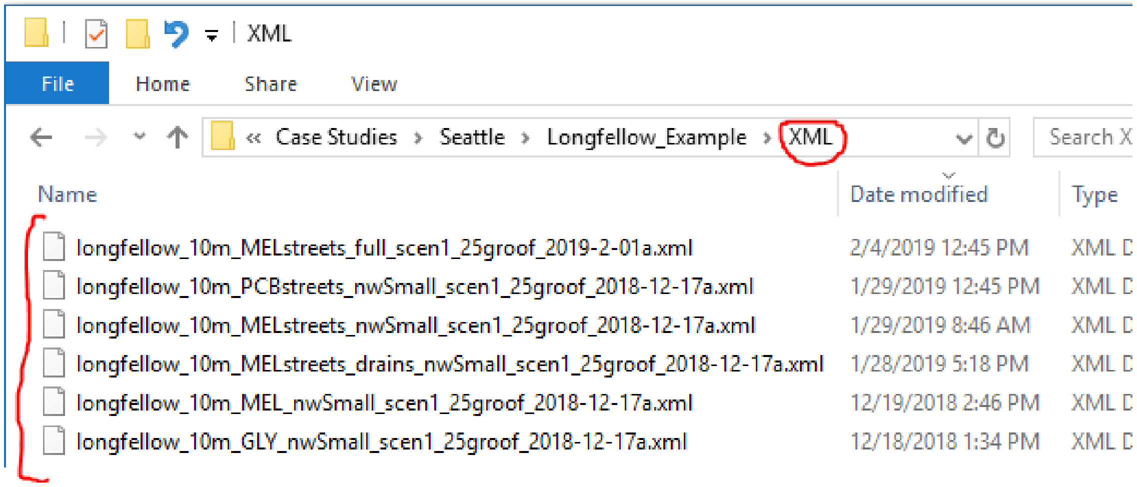
1. Start the VELMA GUI, following the steps from part II of this guide.

2. Open a File Explorer window and navigate to the XML folder of the Longfellow_Example directory. As an example: (details and file names will differ on your computer)


Seattle
File
Home
Share
View
« Velma › 0_VELMA Projects › Case Studies › Seattle
Search S
Longfellow_Example
File
Home
Share
View
« Case Studies › Seattle › Longfellow_Example
Search L

Name	Date modified	Type	Size
_2017Sept11_presentation	1/9/2018 12:47 PM	File folder	
Longfellow	12/19/2018 11:10 ...	File folder	
Longfellow_Example	2/8/2019 3:10 PM	File folder	
Pipers	5/24/2017 2:59 PM	File folder	
Publication	6/6/2018 1:39 PM	File folder	

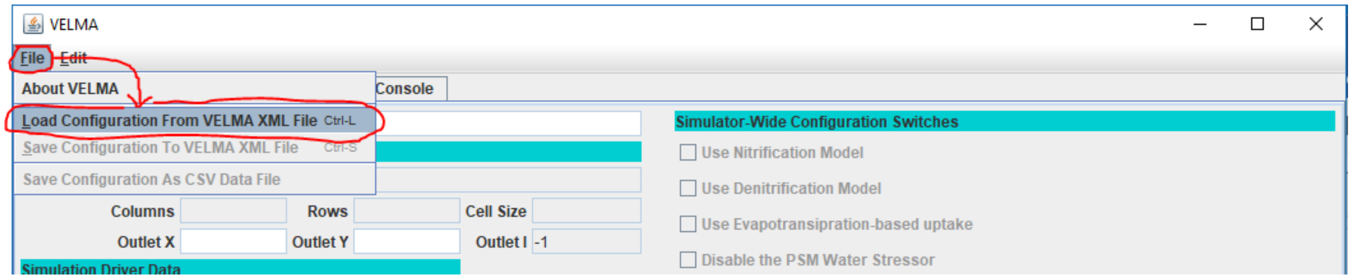
Name	Date modified	Type	Size
DataInputs	1/28/2019 5:07 PM	File folder	
Documents	2/8/2019 3:10 PM	File folder	
Outputs	2/4/2019 12:46 PM	File folder	
XML	2/4/2019 12:43 PM	File folder	



Name	Date modified	Type
longfellow_10m_MELstreets_full_scen1_25groof_2019-2-01a.xml	2/4/2019 12:45 PM	XML D
longfellow_10m_PCBstreets_nwSmall_scen1_25groof_2018-12-17a.xml	1/29/2019 12:45 PM	XML D
longfellow_10m_MELstreets_nwSmall_scen1_25groof_2018-12-17a.xml	1/29/2019 8:46 AM	XML D
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a.xml	1/28/2019 5:18 PM	XML D
longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml	12/19/2018 2:46 PM	XML D
longfellow_10m_GLY_nwSmall_scen1_25groof_2018-12-17a.xml	12/18/2018 1:34 PM	XML D

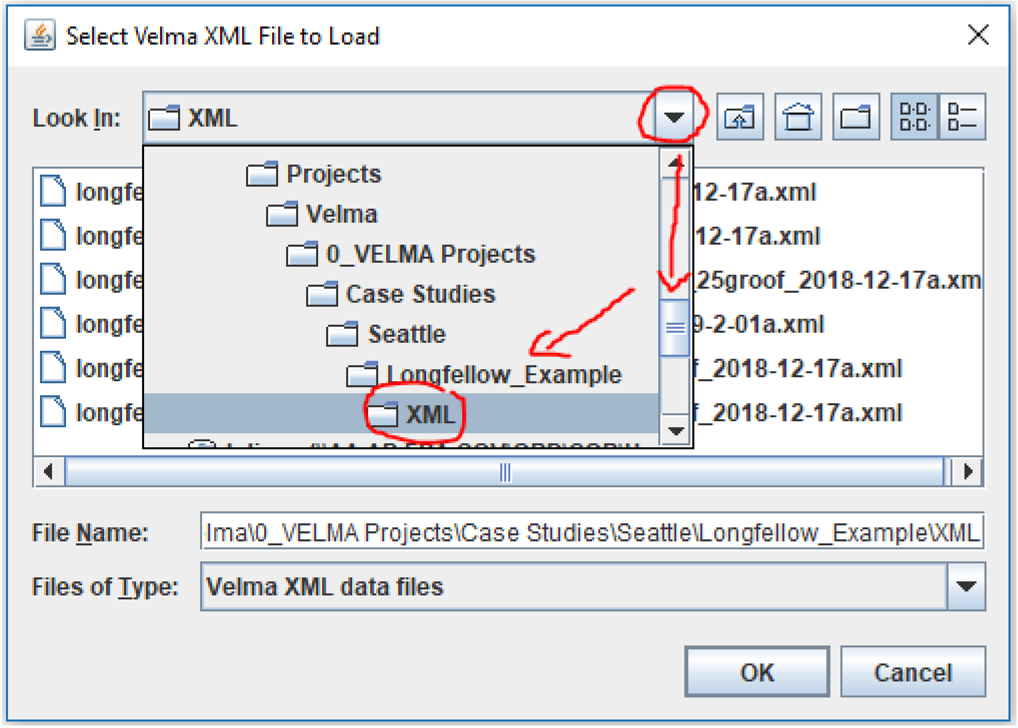
The XML directory contains .xml files – each file contains the configuration for one VELMA simulation scenario.

3. In the VELMA GUI, click the "File" → "Load Configuration From VELMA XML File" menu item:

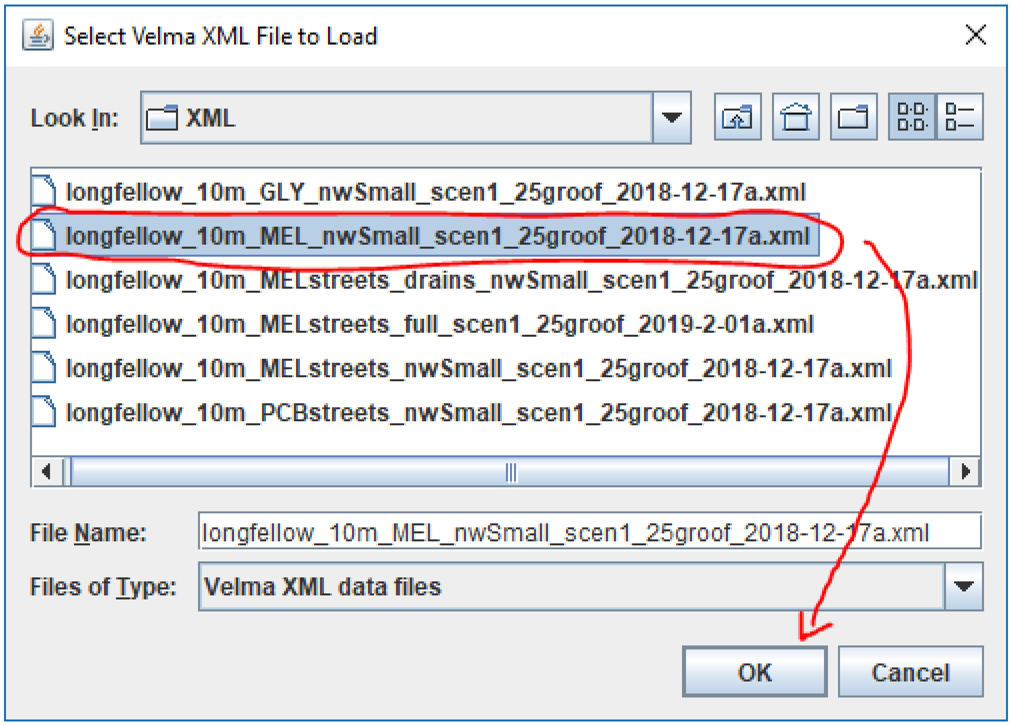

VELMA
File
Edit
About VELMA
Load Configuration From VELMA XML File Ctrl-L
Save Configuration To VELMA XML File Ctrl-S
Save Configuration As CSV Data File
Console
Columns
Rows
Cell Size
Outlet X
Outlet Y
Outlet I -1
Simulation Driver Data
Simulator-Wide Configuration Switches
Use Nitrification Model
Use Denitrification Model
Use Evapotranspiration-based uptake
Disable the PSM Water Stressor

This will open "Select Velma XML File to Load" file browser dialog window.

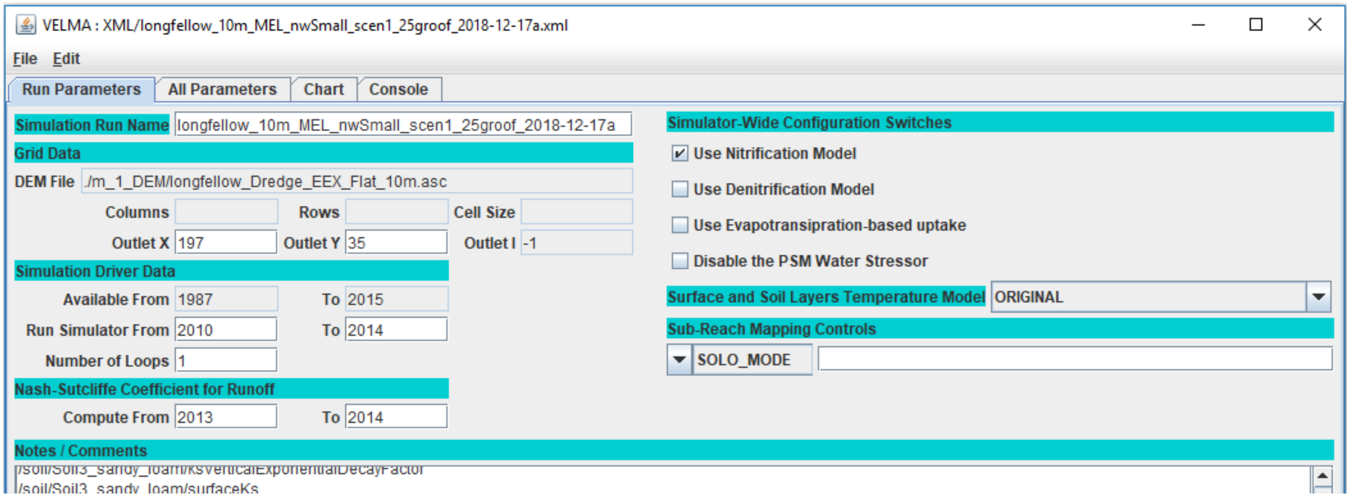
4. In the selector window, navigate to the LongfellowExample\XML directory using the "Look In:" directory field's drop-down navigation list:



5. Select the longfellow_10m_MELstreets_nwSmall_scen1_25groof_2018-12-17a.xml scenario's configuration file and click the OK button.


Select Velma XML File to Load
Look In: XML
longfellow_10m_GLY_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_MELstreets_full_scen1_25groof_2019-2-01a.xml
longfellow_10m_MELstreets_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_PCBstreets_nwSmall_scen1_25groof_2018-12-17a.xml
File Name: longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
Files of Type: Velma XML data files
OK
Cancel

6. The VELMA GUI will load the scenario's configuration.


VELMA : XML/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml

File Edit

Run Parameters | All Parameters | Chart | Console

Simulation Run Name: longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a

Grid Data

DEM File: ./m_1_DEM/longfellow_Dredge_EEX_Flat_10m.asc

Columns: Rows: Cell Size:

Outlet X: 197 Outlet Y: 35 Outlet I: -1

Simulation Driver Data

Available From: 1987 To: 2015

Run Simulator From: 2010 To: 2014

Number of Loops: 1

Nash-Sutcliffe Coefficient for Runoff

Compute From: 2013 To: 2014

Simulator-Wide Configuration Switches

[✓] Use Nitrification Model

[ ] Use Denitrification Model

[ ] Use Evapotranspiration-based uptake

[ ] Disable the PSM Water Stressor

Surface and Soil Layers Temperature Model: ORIGINAL

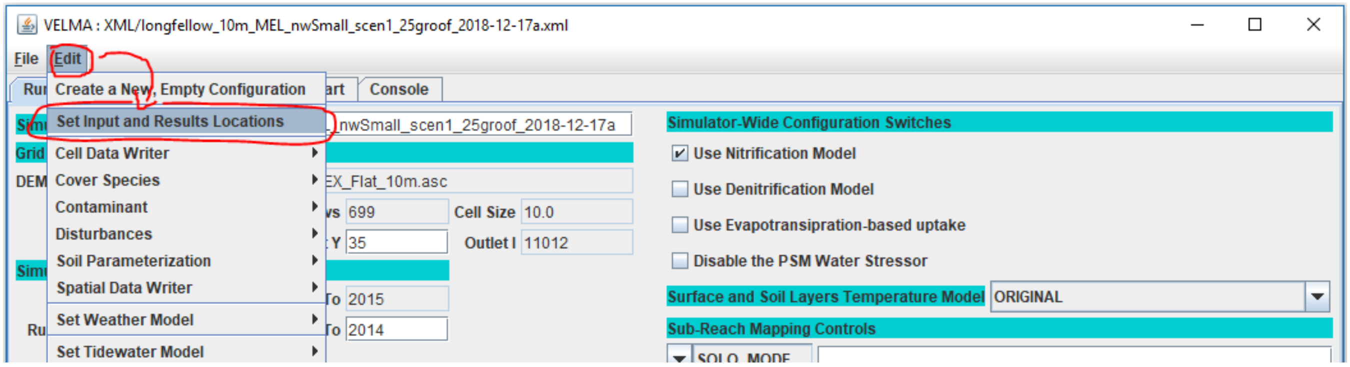
Sub-Reach Mapping Controls: SOLO_MODE

Notes / Comments

/soil/Soil3_sandy_loam/ksVerticalExponentialDecayFactor
/soil/Soil3_sandy_loam/surfaceKs

Notice that the "Columns", "Rows" and "Cell Size" parameters are blank, and the "Outlet I" parameter is -1. This is because the configuration's paths to its input and output (Results folder) data need to be changed to locations specific to your computer. (If these parameters are not blank, you should still explicitly set the Input and Output locations, as described in the following step!)

7. Set the scenario's Input and Output locations to correct values for your computer.
Click the "Edit" → "Set Input and Results Locations" menu item ("Results" is synonymous for "Output"):



Rows: 699 Cell Size: 10.0

Outlet Y: 35 Outlet I: 11012

To: 2015

To: 2014

Simulator-Wide Configuration Switches

[✓] Use Nitrification Model

[ ] Use Denitrification Model

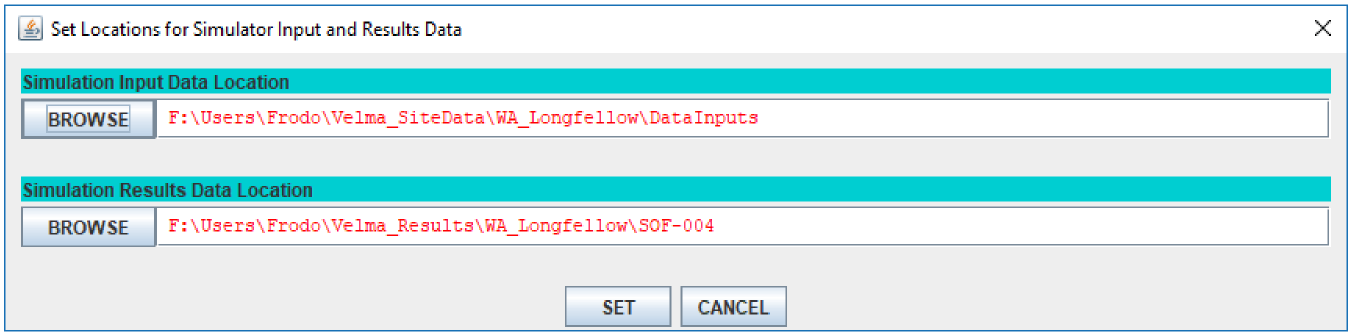
[ ] Use Evapotranspiration-based uptake

[ ] Disable the PSM Water Stressor

Surface and Soil Layers Temperature Model: ORIGINAL

Sub-Reach Mapping Controls: SOLO_MODE

This opens the "Set Locations for Simulator Input and Results Data" dialog window:


Set Locations for Simulator Input and Results Data

Simulation Input Data Location

BROWSE `F:\Users\Frodo\Velma_SiteData\WA_Longfellow\DataInputs`

Simulation Results Data Location

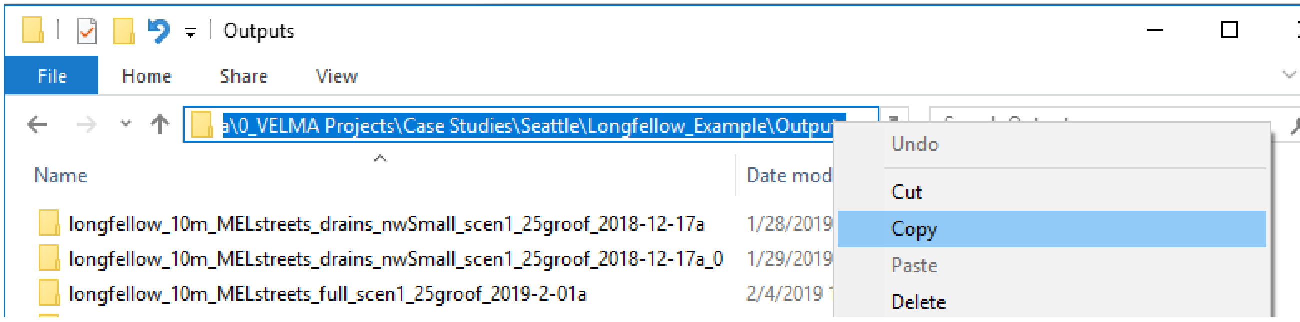
BROWSE `F:\Users\Frodo\Velma_Results\WA_Longfellow\SOF-004`

SET CANCEL

Notice that the Input and Results locations names are red. This indicates that the locations are not found on the current computer.
For these example scenarios, the input location is the Longfellow_example\DataInputs directory, and

the results (or outputs) location is the Longfellow_example\
Set correct input and output locations by any of the following methods:

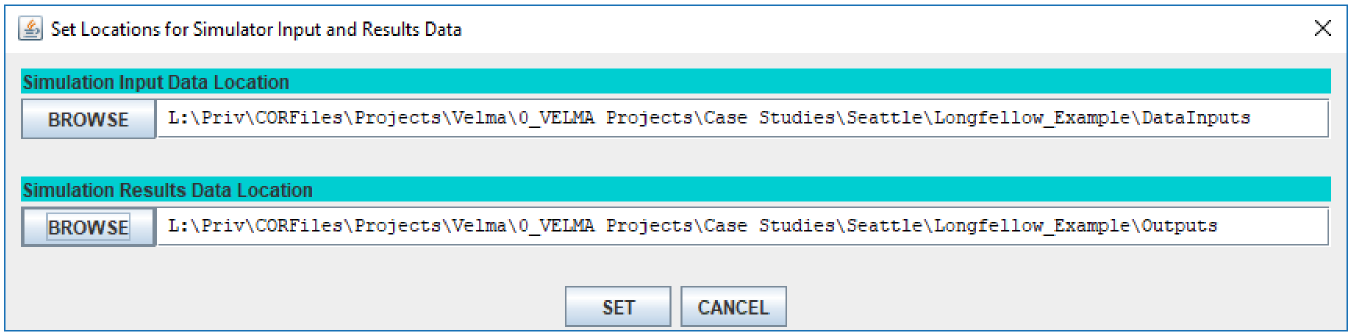
- Click the BROWSE buttons to open File Explorer windows, navigate to the appropriate directories and click "Open" to select them.
- Click the text fields for the Input and Results location names and type in the fully-qualified path names to appropriate directories.
- In a Windows File explorer, navigate to the appropriate directory, double-click the location name text box, then right-click and click the Copy item.


Outputs
File
Home
Share
View
a\0_VELMA Projects\Case Studies\Seattle\Longfellow_Example\Output
Name
Date mod
Undo
Cut
Copy
Paste
Delete

Name	Date mod
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a	1/28/2019
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a_0	1/29/2019
longfellow_10m_MELstreets_full_scen1_25groof_2019-2-01a	2/4/2019

Finally, return to VELMA GUI's "Set Locations …" window, click into the appropriate location name text filed and type Ctrl-v ("paste") to paste the full path in place.

Once the input and output locations have been set correctly, they will display in black.

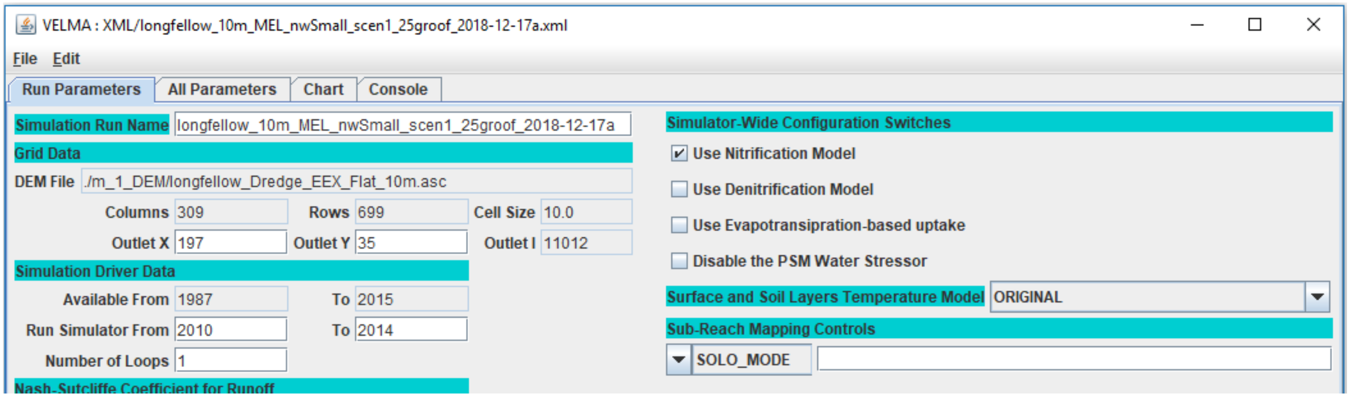

Set Locations for Simulator Input and Results Data
Simulation Input Data Location
BROWSE
Simulation Results Data Location
BROWSE
SET
CANCEL

Simulation Input Data Location: L:\Priv\CORFiles\Projects\Velma\0_VELMA Projects\Case Studies\Seattle\Longfellow_Example\DataInputs

Simulation Results Data Location: L:\Priv\CORFiles\Projects\Velma\0_VELMA Projects\Case Studies\Seattle\Longfellow_Example\Outputs

Click the SET button to set the locations for the configuration.

8. Once the Input and Output (a.k.a. "Results") locations have been correctly set, the Run Parameters panel should show valid values for the "Columns", "Rows", "Cell Size", and "Outlet I" parameter values:


VELMA : XML/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml

Simulation Run Name: longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a

Grid Data
DEM File: ./m_1_DEM/longfellow_Dredge_EEX_Flat_10m.asc
Columns: 309 Rows: 699 Cell Size: 10.0
Outlet X: 197 Outlet Y: 35 Outlet I: 11012

Simulation Driver Data
Available From: 1987 To: 2015
Run Simulator From: 2010 To: 2014
Number of Loops: 1

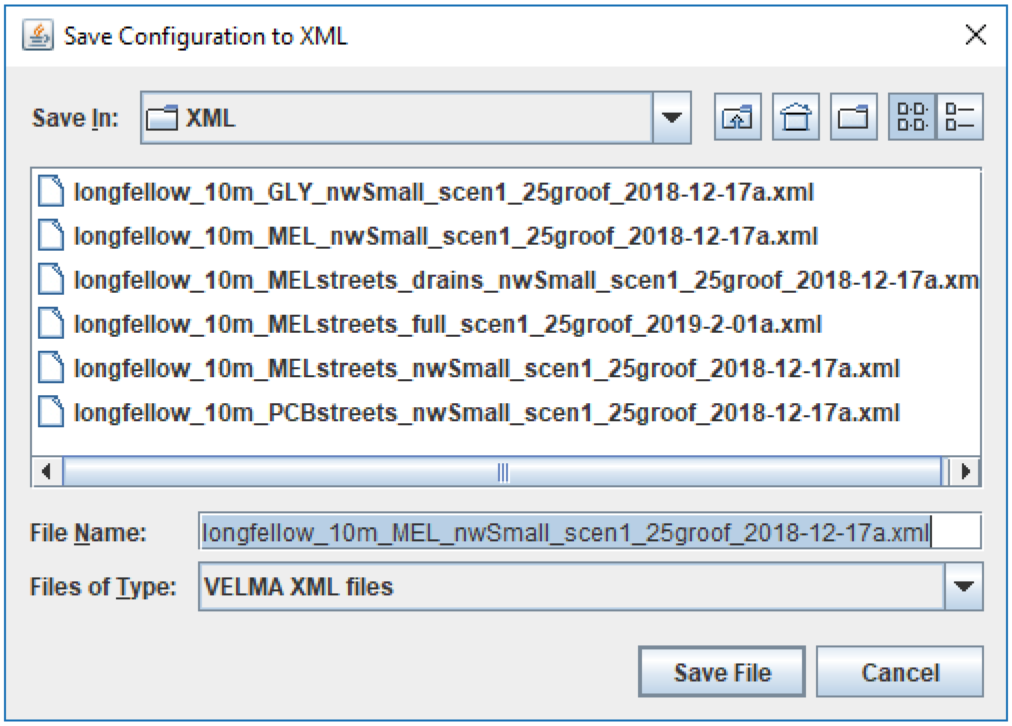
Nash-Sutcliffe Coefficient for Runoff

Simulator-Wide Configuration Switches
[✓] Use Nitrification Model
[ ] Use Denitrification Model
[ ] Use Evapotransipration-based uptake
[ ] Disable the PSM Water Stressor

Surface and Soil Layers Temperature Model: ORIGINAL

Sub-Reach Mapping Controls
SOLO_MODE

9. Save the new Input and Output locations you've specified for the scenario's configuration.
Changes make in the GUI are NOT automatically written back to the original .xml file!
Click the "File" → "Save Configuration to VELMA XML File" menu item.
This opens the "Save Configuration to XML dialog.
A box stating "The file already exists. Replace it?" will pop up. Hit "yes".


Save Configuration to XML

Save In: XML

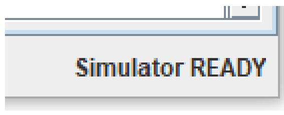
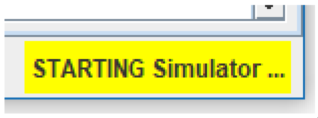
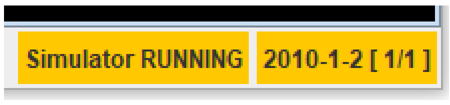
longfellow_10m_GLY_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a.xm
longfellow_10m_MELstreets_full_scen1_25groof_2019-2-01a.xml
longfellow_10m_MELstreets_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_PCBstreets_nwSmall_scen1_25groof_2018-12-17a.xml

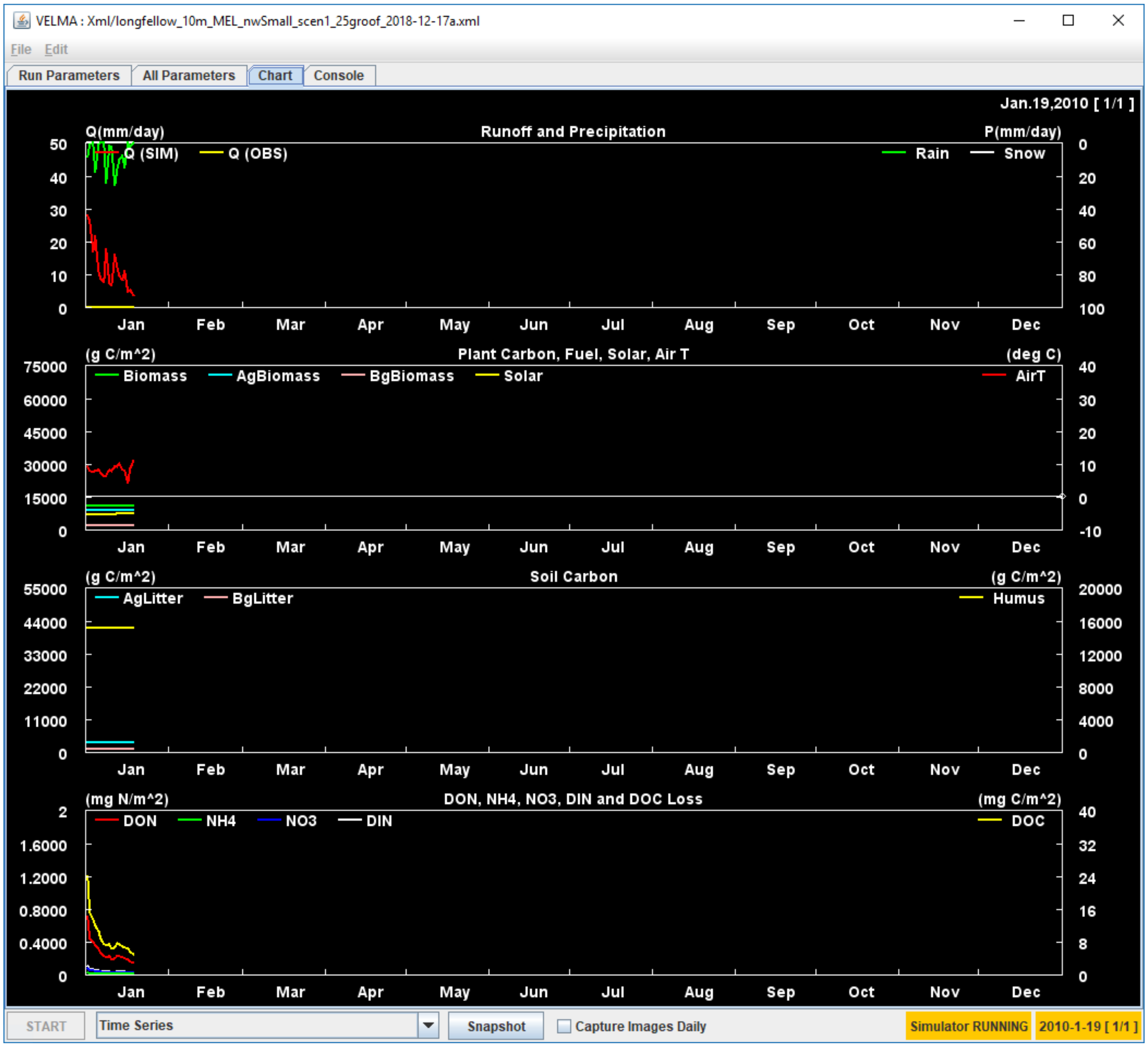
File Name: longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
Files of Type: VELMA XML files

The dialog will open to the same directory you loaded the .xml from, so you should be able to click the "Save File" button without changing anything.

10. Click the "Start" button in the lower-left corner of the VELMA GUI to start the simulation running.

Monitoring a Running VELMA Simulation

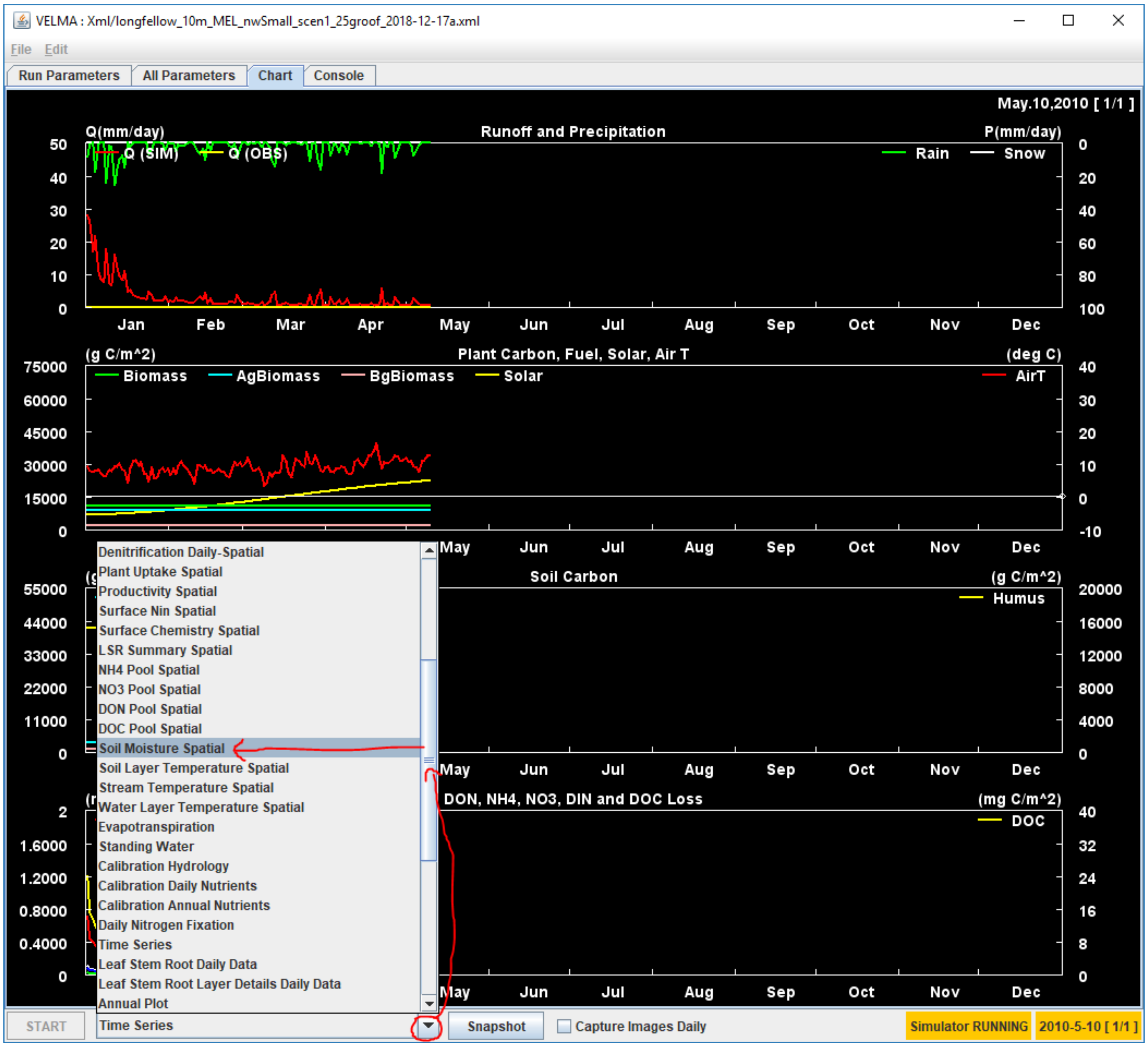
After the clicking Start, the simulator status indicator in the lower-right corner should change from: 
Simulator READY
to: 
STARTING Simulator ...
, which indicates that the VELMA simulator engine is initializing and preparing to run. Depending on the size of your simulation and how capable your computer is, this can take anywhere from a few seconds to many minutes.
When the simulator completes initializing, it will start running, and the lower-right corner status message will change to look like this: 
Simulator RUNNING 2010-1-2 [1/1]
.
The GUI will also automatically shift to display the "Chart" tab panel, with the "Time Series" chart as its default display as shown below:



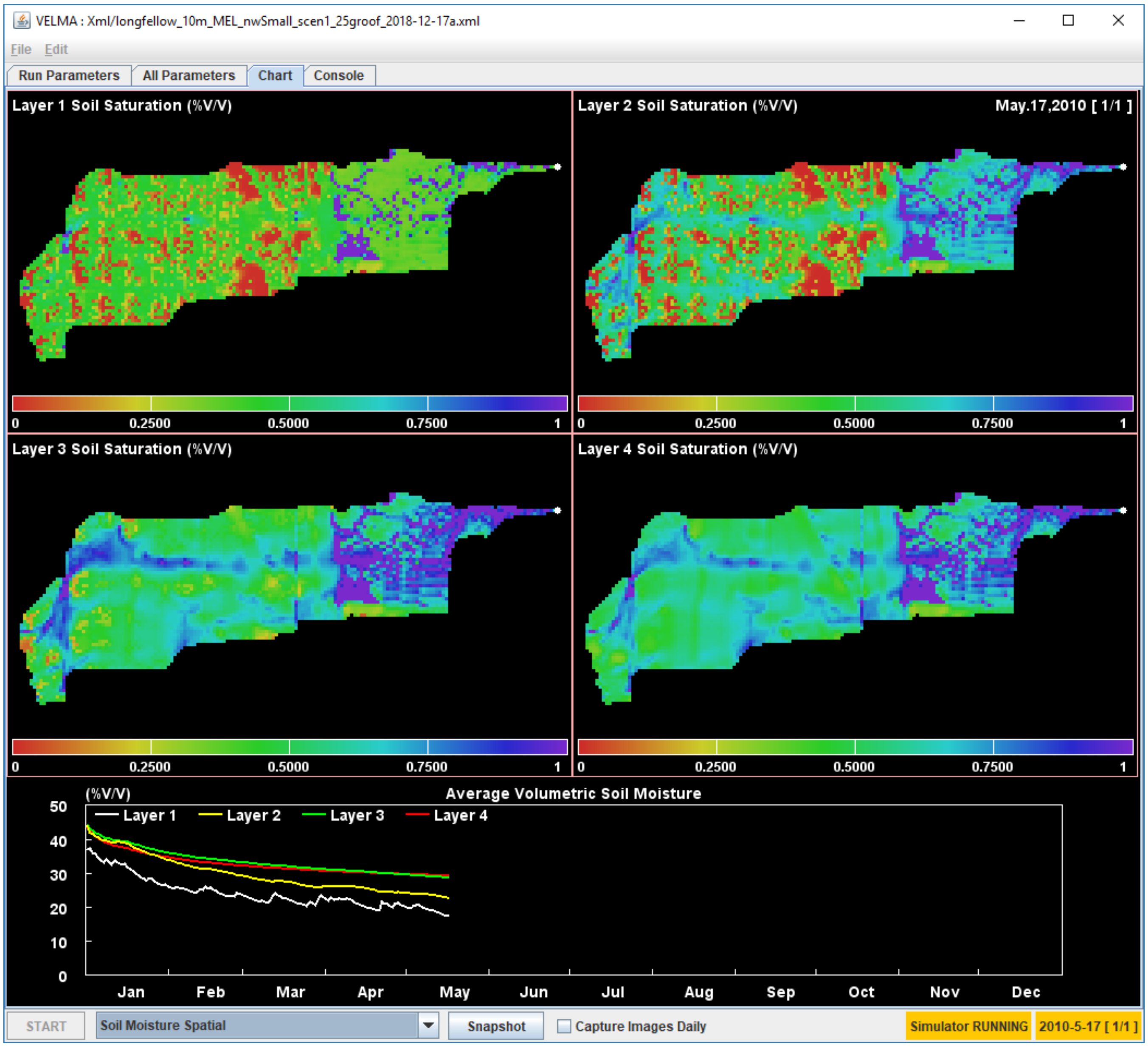
You can shift between the various tab panels in the GUI while a simulation is running, but you should not (and generally, cannot) alter the simulation parameters, although you can view them.

The most useful tab panels during a simulation run are the “Chart” and “Console” panels.

The type layout of runtime data displayed in the Chart panel is selectable via the Display Selector that appears next to the Start button when the Chart panel is active. Click the selector’s ▼ drop-down button to display the list of data/layouts. The following screen-capture shows using the selector’s list to change the Chart panel to display Soil Moisture spatial (and some temporal) data:


VELMA : Xml/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
May.10,2010 [1/1]
Runoff and Precipitation
Q(mm/day)
P(mm/day)
Q (SIM)
Q (OBS)
Rain
Snow
Plant Carbon, Fuel, Solar, Air T
(g C/m^2)
(deg C)
Biomass
AgBiomass
BgBiomass
Solar
AirT
Soil Carbon
(g C/m^2)
Humus
DON, NH4, NO3, DIN and DOC Loss
(mg C/m^2)
DOC
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Denitrification Daily-Spatial
Plant Uptake Spatial
Productivity Spatial
Surface Nin Spatial
Surface Chemistry Spatial
LSR Summary Spatial
NH4 Pool Spatial
NO3 Pool Spatial
DON Pool Spatial
DOC Pool Spatial
Soil Moisture Spatial
Soil Layer Temperature Spatial
Stream Temperature Spatial
Water Layer Temperature Spatial
Evapotranspiration
Standing Water
Calibration Hydrology
Calibration Daily Nutrients
Calibration Annual Nutrients
Daily Nitrogen Fixation
Time Series
Leaf Stem Root Daily Data
Leaf Stem Root Layer Details Daily Data
Annual Plot
START
Time Series
Snapshot
Capture Images Daily
Simulator RUNNING
2010-5-10 [1/1]

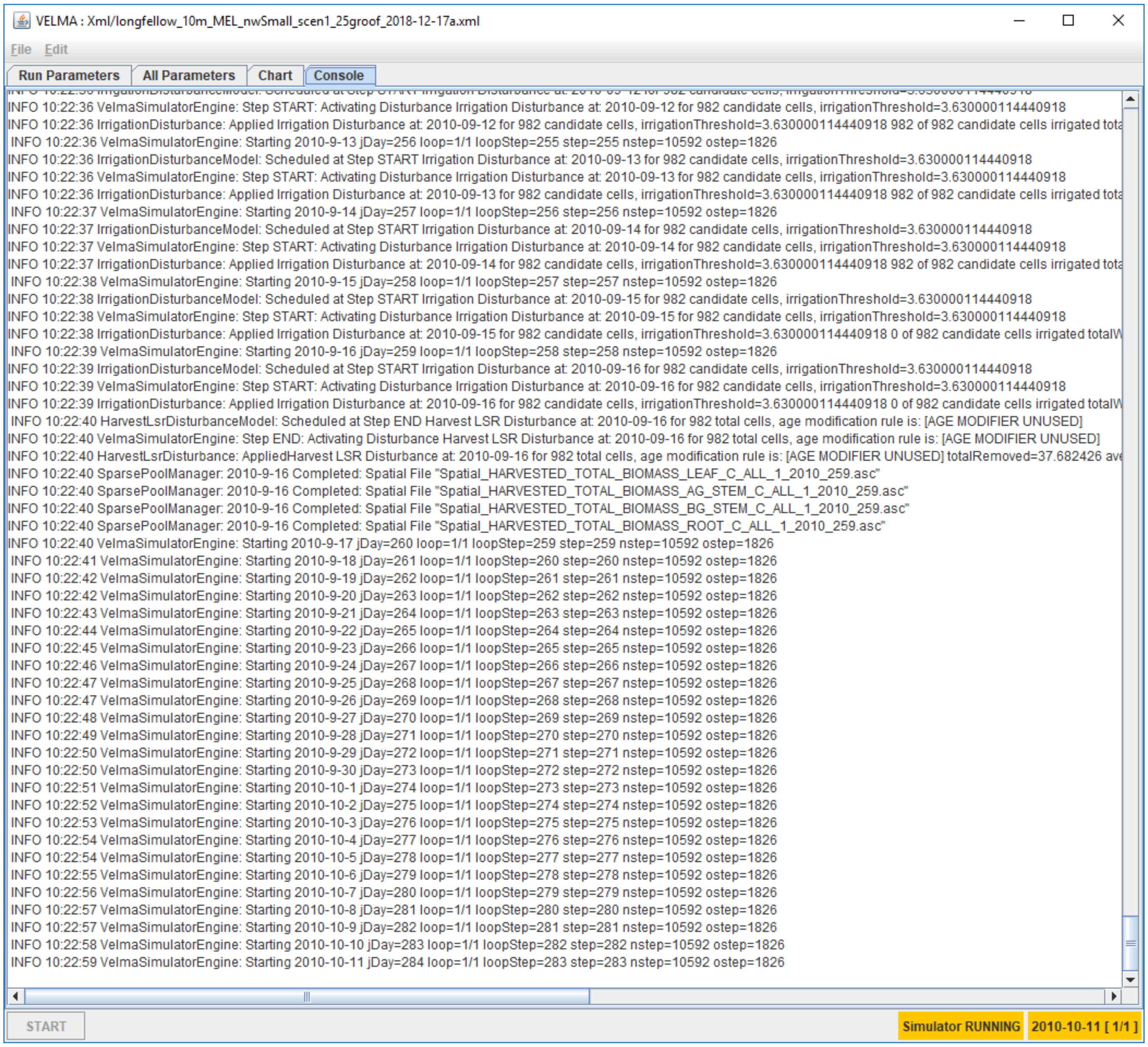
Here is what the Chart display looks like after the above selection:


VELMA : Xml/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Layer 1 Soil Saturation (%V/V)
Layer 2 Soil Saturation (%V/V)
May.17,2010 [1/1]
Layer 3 Soil Saturation (%V/V)
Layer 4 Soil Saturation (%V/V)
0 0.2500 0.5000 0.7500 1
(%V/V)
Average Volumetric Soil Moisture
Layer 1 Layer 2 Layer 3 Layer 4
50 40 30 20 10 0
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
START
Soil Moisture Spatial
Snapshot
Capture Images Daily
Simulator RUNNING
2010-5-17 [1/1]

Clicking the “Snapshot” button captures the current Chart display. The captured image appears in the simulation’s results directory as a .PNG file with the word “out_” prefixed to the simulation name.

Clicking the “Capture Images Daily” checkbox makes the GUI capture one .PNG per simulation step (i.e. day) until it is clicked again (i.e. unchecked). The images captured appear in the simulation’s results directory as a group of .PNG files, each name prefixed with “day_” and suffixed with the year and Julian day of the image (e.g. “_2011_278”) just before the “.png” file-extension.

The “Console” tab panel displays tracing statements about the current state of the simulation run. The following screen capture shows the Console for a typical simulation run. Note that the simulator prints at least one “INFO” statement per simulation day, and each statement contains a real-time time-stamp, along with simulation-time date and day information. Periodic disturbances that were configured as part of the simulation are also noted.



```
INFO 10:22:36 VelmaSimulatorEngine: Step START: Activating Disturbance Irrigation Disturbance at: 2010-09-12 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:36 IrrigationDisturbance: Applied Irrigation Disturbance at: 2010-09-12 for 982 candidate cells, irrigationThreshold=3.630000114440918 982 of 982 candidate cells irrigated tota
 INFO 10:22:36 VelmaSimulatorEngine: Starting 2010-9-13 jDay=256 loop=1/1 loopStep=255 step=255 nstep=10592 ostep=1826
INFO 10:22:36 IrrigationDisturbanceModel: Scheduled at Step START Irrigation Disturbance at: 2010-09-13 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:36 VelmaSimulatorEngine: Step START: Activating Disturbance Irrigation Disturbance at: 2010-09-13 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:36 IrrigationDisturbance: Applied Irrigation Disturbance at: 2010-09-13 for 982 candidate cells, irrigationThreshold=3.630000114440918 982 of 982 candidate cells irrigated tota
 INFO 10:22:37 VelmaSimulatorEngine: Starting 2010-9-14 jDay=257 loop=1/1 loopStep=256 step=256 nstep=10592 ostep=1826
INFO 10:22:37 IrrigationDisturbanceModel: Scheduled at Step START Irrigation Disturbance at: 2010-09-14 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:37 VelmaSimulatorEngine: Step START: Activating Disturbance Irrigation Disturbance at: 2010-09-14 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:37 IrrigationDisturbance: Applied Irrigation Disturbance at: 2010-09-14 for 982 candidate cells, irrigationThreshold=3.630000114440918 982 of 982 candidate cells irrigated tota
 INFO 10:22:38 VelmaSimulatorEngine: Starting 2010-9-15 jDay=258 loop=1/1 loopStep=257 step=257 nstep=10592 ostep=1826
INFO 10:22:38 IrrigationDisturbanceModel: Scheduled at Step START Irrigation Disturbance at: 2010-09-15 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:38 VelmaSimulatorEngine: Step START: Activating Disturbance Irrigation Disturbance at: 2010-09-15 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:38 IrrigationDisturbance: Applied Irrigation Disturbance at: 2010-09-15 for 982 candidate cells, irrigationThreshold=3.630000114440918 0 of 982 candidate cells irrigated totalW
 INFO 10:22:39 VelmaSimulatorEngine: Starting 2010-9-16 jDay=259 loop=1/1 loopStep=258 step=258 nstep=10592 ostep=1826
INFO 10:22:39 IrrigationDisturbanceModel: Scheduled at Step START Irrigation Disturbance at: 2010-09-16 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:39 VelmaSimulatorEngine: Step START: Activating Disturbance Irrigation Disturbance at: 2010-09-16 for 982 candidate cells, irrigationThreshold=3.630000114440918
INFO 10:22:39 IrrigationDisturbance: Applied Irrigation Disturbance at: 2010-09-16 for 982 candidate cells, irrigationThreshold=3.630000114440918 0 of 982 candidate cells irrigated totalW
 INFO 10:22:40 HarvestLsrDisturbanceModel: Scheduled at Step END Harvest LSR Disturbance at: 2010-09-16 for 982 total cells, age modification rule is: [AGE MODIFIER UNUSED]
INFO 10:22:40 VelmaSimulatorEngine: Step END: Activating Disturbance Harvest LSR Disturbance at: 2010-09-16 for 982 total cells, age modification rule is: [AGE MODIFIER UNUSED]
INFO 10:22:40 HarvestLsrDisturbance: AppliedHarvest LSR Disturbance at: 2010-09-16 for 982 total cells, age modification rule is: [AGE MODIFIER UNUSED] totalRemoved=37.682426 ave
INFO 10:22:40 SparsePoolManager: 2010-9-16 Completed: Spatial File "Spatial_HARVESTED_TOTAL_BIOMASS_LEAF_C_ALL_1_2010_259.asc"
INFO 10:22:40 SparsePoolManager: 2010-9-16 Completed: Spatial File "Spatial_HARVESTED_TOTAL_BIOMASS_AG_STEM_C_ALL_1_2010_259.asc"
INFO 10:22:40 SparsePoolManager: 2010-9-16 Completed: Spatial File "Spatial_HARVESTED_TOTAL_BIOMASS_BG_STEM_C_ALL_1_2010_259.asc"
INFO 10:22:40 SparsePoolManager: 2010-9-16 Completed: Spatial File "Spatial_HARVESTED_TOTAL_BIOMASS_ROOT_C_ALL_1_2010_259.asc"
INFO 10:22:40 VelmaSimulatorEngine: Starting 2010-9-17 jDay=260 loop=1/1 loopStep=259 step=259 nstep=10592 ostep=1826
 INFO 10:22:41 VelmaSimulatorEngine: Starting 2010-9-18 jDay=261 loop=1/1 loopStep=260 step=260 nstep=10592 ostep=1826
 INFO 10:22:42 VelmaSimulatorEngine: Starting 2010-9-19 jDay=262 loop=1/1 loopStep=261 step=261 nstep=10592 ostep=1826
 INFO 10:22:42 VelmaSimulatorEngine: Starting 2010-9-20 jDay=263 loop=1/1 loopStep=262 step=262 nstep=10592 ostep=1826
 INFO 10:22:43 VelmaSimulatorEngine: Starting 2010-9-21 jDay=264 loop=1/1 loopStep=263 step=263 nstep=10592 ostep=1826
 INFO 10:22:44 VelmaSimulatorEngine: Starting 2010-9-22 jDay=265 loop=1/1 loopStep=264 step=264 nstep=10592 ostep=1826
 INFO 10:22:45 VelmaSimulatorEngine: Starting 2010-9-23 jDay=266 loop=1/1 loopStep=265 step=265 nstep=10592 ostep=1826
 INFO 10:22:46 VelmaSimulatorEngine: Starting 2010-9-24 jDay=267 loop=1/1 loopStep=266 step=266 nstep=10592 ostep=1826
 INFO 10:22:47 VelmaSimulatorEngine: Starting 2010-9-25 jDay=268 loop=1/1 loopStep=267 step=267 nstep=10592 ostep=1826
 INFO 10:22:47 VelmaSimulatorEngine: Starting 2010-9-26 jDay=269 loop=1/1 loopStep=268 step=268 nstep=10592 ostep=1826
 INFO 10:22:48 VelmaSimulatorEngine: Starting 2010-9-27 jDay=270 loop=1/1 loopStep=269 step=269 nstep=10592 ostep=1826
 INFO 10:22:49 VelmaSimulatorEngine: Starting 2010-9-28 jDay=271 loop=1/1 loopStep=270 step=270 nstep=10592 ostep=1826
 INFO 10:22:50 VelmaSimulatorEngine: Starting 2010-9-29 jDay=272 loop=1/1 loopStep=271 step=271 nstep=10592 ostep=1826
 INFO 10:22:50 VelmaSimulatorEngine: Starting 2010-9-30 jDay=273 loop=1/1 loopStep=272 step=272 nstep=10592 ostep=1826
 INFO 10:22:51 VelmaSimulatorEngine: Starting 2010-10-1 jDay=274 loop=1/1 loopStep=273 step=273 nstep=10592 ostep=1826
 INFO 10:22:52 VelmaSimulatorEngine: Starting 2010-10-2 jDay=275 loop=1/1 loopStep=274 step=274 nstep=10592 ostep=1826
 INFO 10:22:53 VelmaSimulatorEngine: Starting 2010-10-3 jDay=276 loop=1/1 loopStep=275 step=275 nstep=10592 ostep=1826
 INFO 10:22:54 VelmaSimulatorEngine: Starting 2010-10-4 jDay=277 loop=1/1 loopStep=276 step=276 nstep=10592 ostep=1826
 INFO 10:22:54 VelmaSimulatorEngine: Starting 2010-10-5 jDay=278 loop=1/1 loopStep=277 step=277 nstep=10592 ostep=1826
 INFO 10:22:55 VelmaSimulatorEngine: Starting 2010-10-6 jDay=279 loop=1/1 loopStep=278 step=278 nstep=10592 ostep=1826
 INFO 10:22:56 VelmaSimulatorEngine: Starting 2010-10-7 jDay=280 loop=1/1 loopStep=279 step=279 nstep=10592 ostep=1826
 INFO 10:22:57 VelmaSimulatorEngine: Starting 2010-10-8 jDay=281 loop=1/1 loopStep=280 step=280 nstep=10592 ostep=1826
 INFO 10:22:57 VelmaSimulatorEngine: Starting 2010-10-9 jDay=282 loop=1/1 loopStep=281 step=281 nstep=10592 ostep=1826
 INFO 10:22:58 VelmaSimulatorEngine: Starting 2010-10-10 jDay=283 loop=1/1 loopStep=282 step=282 nstep=10592 ostep=1826
 INFO 10:22:59 VelmaSimulatorEngine: Starting 2010-10-11 jDay=284 loop=1/1 loopStep=283 step=283 nstep=10592 ostep=1826
```

Reviewing the Results of a VELMA Simulator Run

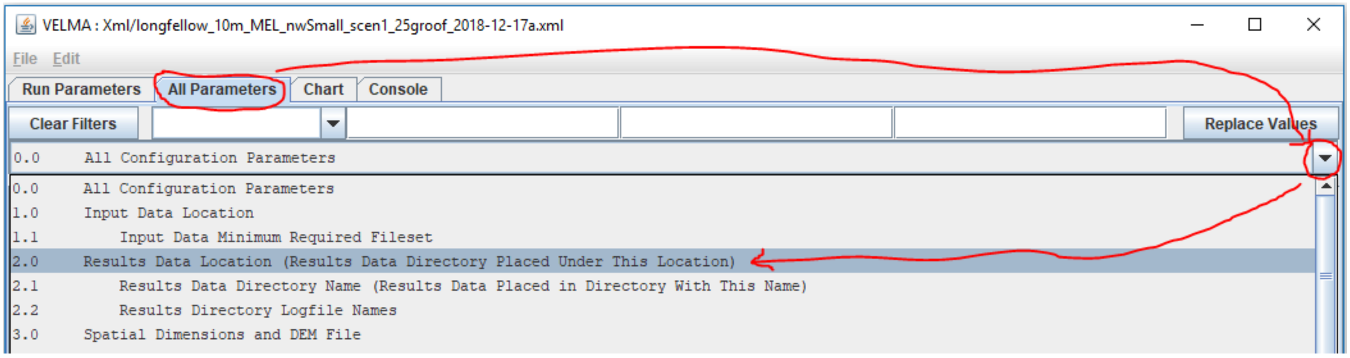
Section IV Topics:

- Results Location
- Types of Results

Results Location: Where Are Results Files Located?

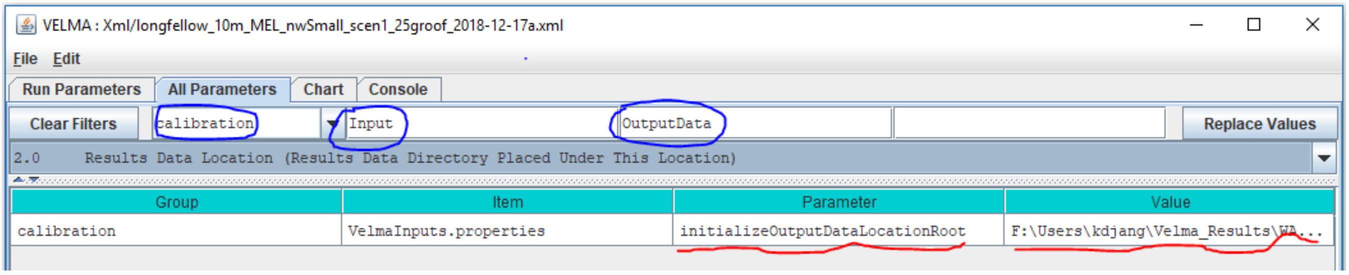
VELMA simulation results are written to multiple files in a directory specified by the simulation configuration. You specified the output location in step 7 above, as part of loading and running a simulation, but if you've forgotten what the location was, you can find it using the "All Parameters" tab panel's parameters table:

1. Click the drop-down selector button along the upper-right side of the All Parameters tab panel and open the panel's Parameters Outline selector list:



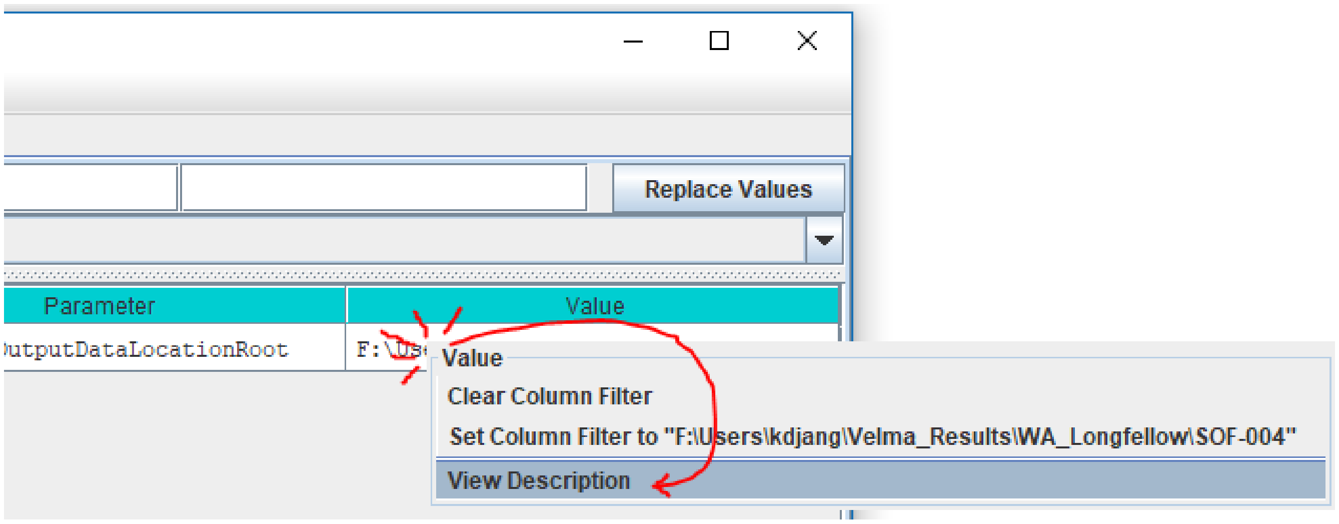
```
0.0     All Configuration Parameters
0.0     All Configuration Parameters
1.0     Input Data Location
1.1        Input Data Minimum Required Fileset
2.0     Results Data Location (Results Data Directory Placed Under This Location)
2.1        Results Data Directory Name (Results Data Placed in Directory With This Name)
2.2        Results Directory Logfile Names
3.0     Spatial Dimensions and DEM File
```

2. Click "2.0 Results Data Location …" – this will populate the parameter column filters with a set of filter terms (circled in blue below) that filter down the parameters displayed in the table. In this case, the single parameter displayed is the one that specifies the location of your simulation run's results. That Parameter is `initializeOutputDataLocationRoot`, and its Value is the fully-qualified path to directory that contains your simulation's results directory.

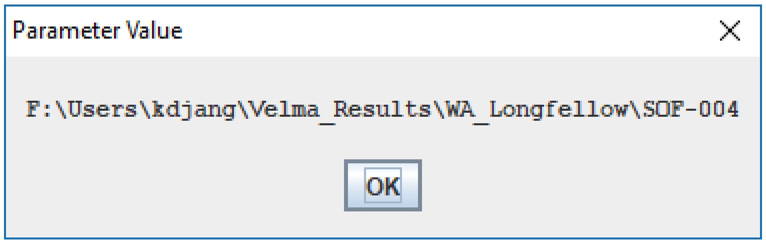

VELMA : Xml/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
File
Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
calibration
Input
OutputData
Replace Values
2.0 Results Data Location (Results Data Directory Placed Under This Location)

Group	Item	Parameter	Value
calibration	VelmaInputs.properties	initializeOutputDataLocationRoot	F:\Users\kdjang\Velma_Results\WA...

3. The value (i.e. path) may be truncated by the bounds of the table's column width.
You can click-drag the side of the GUI to make it wider, but a quick way to see the entire results location is to right-click the Value column field and select the "View Description" option in the context menu that opens:



This opens a small dialog box that contains the full value of the parameter:



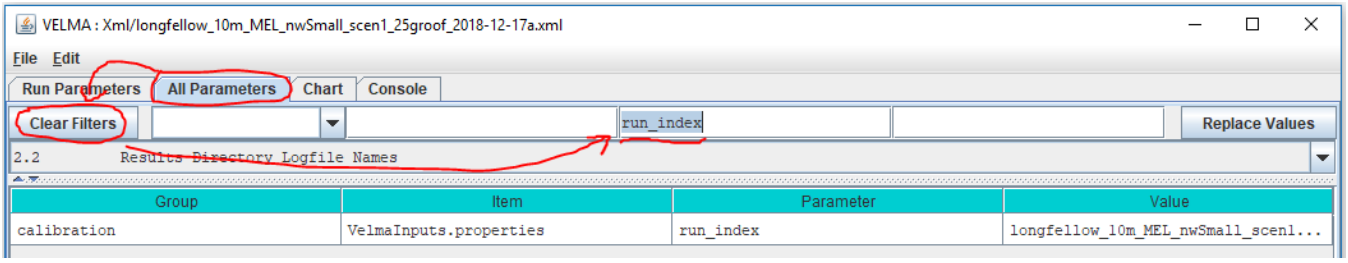
```
F:\Users\kdjang\Velma_Results\WA_Longfellow\SOF-004
```

(Unfortunately, you cannot click-highlight → copy the text from this dialog window. Hopefully, we can implement that handy feature in the future.)

As mentioned above, the value of the `initializeOutputDataLocationRoot` parameter is not the directory that your simulation results are stored in. They are stored in a directory of their own, *underneath* the `initializeOutputDataLocationRoot` location.

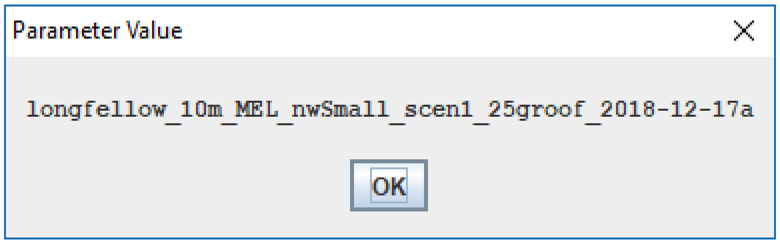
The name of your simulation results is specified by the `run_index` parameter.
To find its value:

1. In VELMA GUI's "All Parameter's tab panel, click the "Clear Filters" button, then type `run_index` (not leading or trailing whitespace, single underbar "_" character) into the middle filter text field, then press the Return (a.k.a. Enter) key:


VELMA : Xml/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
run_index
Replace Values
2.2 Results Directory Logfile Names

Group	Item	Parameter	Value
calibration	VelmaInputs.properties	run_index	longfellow_10m_MEL_nwSmall_scen1...

2. The entire value may not be visible in the column's field – use the right-click → "View Description" mentioned above to display the full name:



```
longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a
```

So, for the above examples, the simulator will write output files to the following directory:

```
F:\Users\kdjang\Velma_Results\WA_Longfellow\SOF-
004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a
```

Assuming you've run the example simulation (as outlined by this guide above) find the results directory for the example simulation run on your computer and open that location in a File Explorer window.

Types of Results: What Data is Available in The Output Files?

As previously noted, a single VELMA simulation run produces multiple results files of various types of data. Here are descriptions of a few of the most pertinent for the example simulation you just ran.

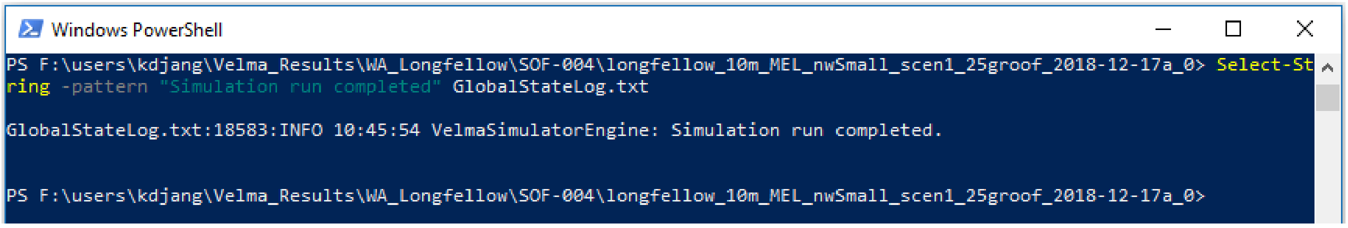
`GlobalStateLog.txt`

This text file contains a copy of most of the information that is written to the VELMA GUI's "Console" panel. You can view its contents in a text editor (e.g. Microsoft's Notepad).

A successful simulation run should contain a log message like following near the bottom of the file:

```
INFO 10:45:54 VelmaSimulatorEngine: Simulation run completed.
```

You can search for this message in a PowerShell window using the Select-String command, as a quick way of verifying the simulation ran to completion:



```
PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0> Select-St
ring -pattern "Simulation run completed" GlobalStateLog.txt

GlobalStateLog.txt:18583:INFO 10:45:54 VelmaSimulatorEngine: Simulation run completed.


PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0>
```

If the simulation in the example screenshot above hadn't completed, the -Pattern "Simulation run completed" would not have been found in GlobalStateLog.txt and the Select-String command would have returned a blank line.

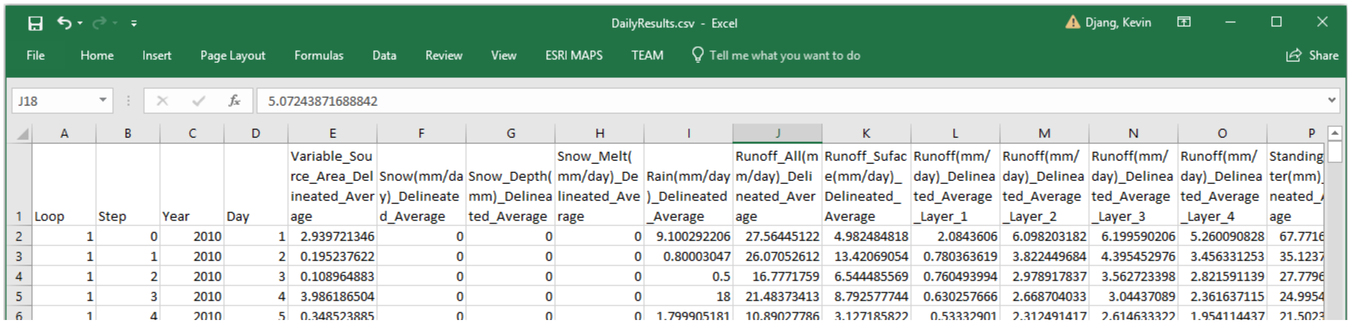
The GlobalStateLog.txt file contains only most – instead of all – of what is written to VELMA GUI's Console panel because setting up the GlobalStateLog.txt file itself is part of the VELMA simulator's initialization. Before that initialization is completed, VELMA can echo log messages to the Console window, but not to GlobalStateLog.txt – because it hasn't been initialized yet.

You can usually ignore this distinction. However, if you find a GlobalStateLog.txt file *without* the "Simulation run completed" message, but also no error or warning messages, then it's likely that an error occurred early in the simulator initialization process. Check the GUI's Console panel (use the panel's scroll bar, if necessary, to review the very top of the log) for warning or error messages.

`DailyResults.csv`

This is a file of comma-separated values (.csv) that reports a lot of the core data state of the VELMA simulator. You can open this file in Notepad, but it's more useful to view it in Microsoft Excel, or load it into R for analysis.

Here's a screenshot of the first few rows of the example run's DailyResults.csv file:

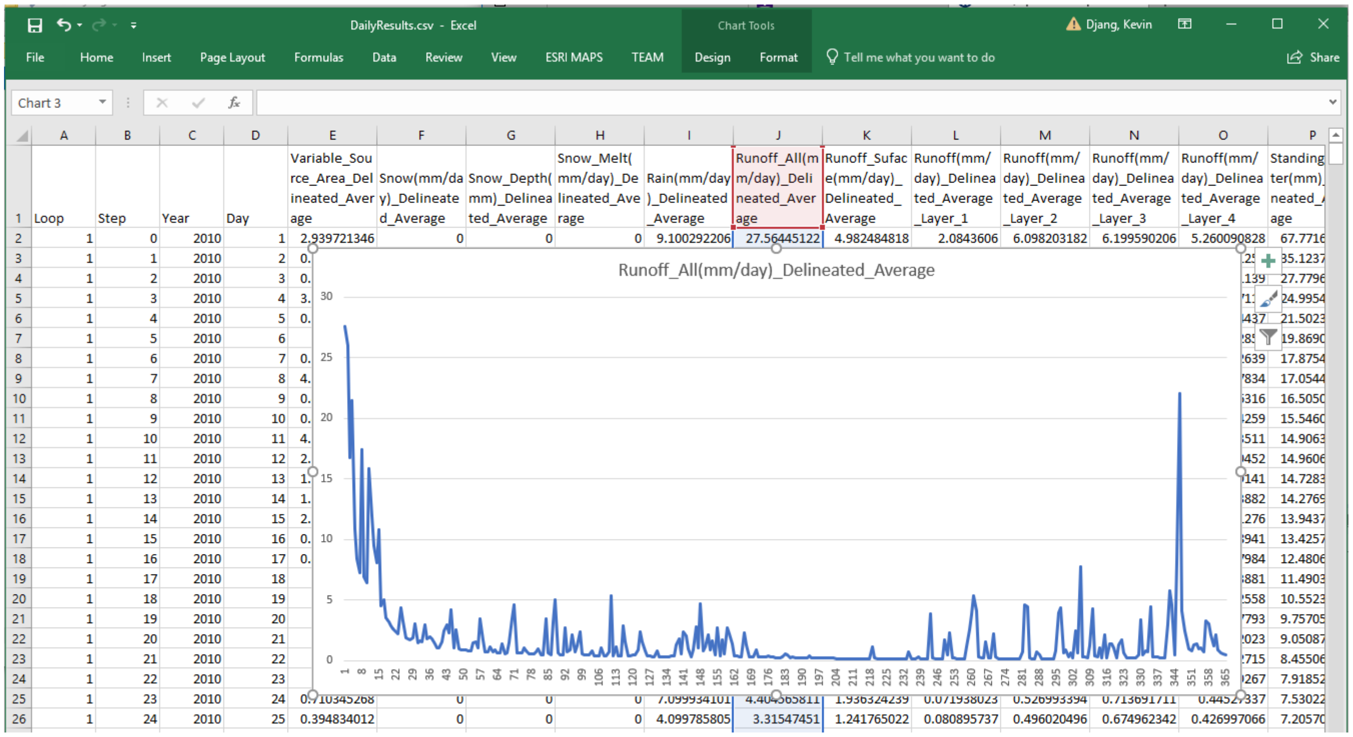


5.07243871688842

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Loop	Step	Year	Day	Variable_Source_Area_Delineated_Average	Snow(mm/day)_Delineated_Average	Snow_Depth(mm)_Delineated_Average	Snow_Melt(mm/day)_Delineated_Average	Rain(mm/day)_Delineated_Average	Runoff_All(mm/day)_Delineated_Average	Runoff_Sufface(mm/day)_Delineated_Average	Runoff(mm/day)_Delineated_Average_Layer_1	Runoff(mm/day)_Delineated_Average_Layer_2	Runoff(mm/day)_Delineated_Average_Layer_3	Runoff(mm/day)_Delineated_Average_Layer_4	Standing ter(mm)_neated_age
2	1	0	2010	1	2.939721346	0	0	0	9.100292206	27.56445122	4.982484818	2.0843606	6.098203182	6.199590206	5.260090828	67.7716
3	1	1	2010	2	0.195237622	0	0	0	0.80003047	26.07052612	13.42069054	0.780363619	3.822449684	4.395452976	3.456331253	35.1237
4	1	2	2010	3	0.108964883	0	0	0	0.5	16.7771759	6.544485569	0.760493994	2.978917837	3.562723398	2.821591139	27.7796
5	1	3	2010	4	3.986186504	0	0	0	18	21.48373413	8.792577744	0.630257666	2.668704033	3.04437089	2.361637115	24.9954
6	1	4	2010	5	0.348523885	0	0	0	1.799905181	10.89027786	3.127185822	0.53332901	2.312491417	2.614633322	1.954114437	21.5023

Each row in the file is one simulation step's worth of data, and the first four columns in each row provide information about which simulation step the data is for.

Using Excel, provides a quick way to visualize DailyResults data via graphs.
Here is a graph of the first year's "Runoff_All" column data:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Loop	Step	Year	Day	Variable_Source_Area_Delineated_Average	Snow(mm/day)_Delineated_Average	Snow_Depth(mm)_Delineated_Average	Snow_Melt(mm/day)_Delineated_Average	Rain(mm/day)_Delineated_Average	Runoff_All(mm/day)_Delineated_Average	Runoff_Sufface(mm/day)_Delineated_Average	Runoff(mm/day)_Delineated_Average_Layer_1	Runoff(mm/day)_Delineated_Average_Layer_2	Runoff(mm/day)_Delineated_Average_Layer_3	Runoff(mm/day)_Delineated_Average_Layer_4	Standing ter(mm)_neated_age
2	1	0	2010	1	2.939721346	0	0	0	9.100292206	27.56445122	4.982484818	2.0843606	6.098203182	6.199590206	5.260090828	67.7716
3	1	1	2010	2	0.										.25	35.1237
4	1	2	2010	3	0.										.139	27.7796
5	1	3	2010	4	3.										71	24.9954
6	1	4	2010	5	0.										437	21.5023
7	1	5	2010	6											85	19.8690
8	1	6	2010	7	0.										639	17.8754
9	1	7	2010	8	4.										834	17.0544
10	1	8	2010	9	0.										316	16.5050
11	1	9	2010	10	0.										259	15.5460
12	1	10	2010	11	4.										511	14.9063
13	1	11	2010	12	2.										452	14.9606
14	1	12	2010	13	1.										141	14.7283
15	1	13	2010	14	1.										882	14.2769
16	1	14	2010	15	2.										.276	13.9437
17	1	15	2010	16	0.										941	13.4257
18	1	16	2010	17	0.										984	12.4806
19	1	17	2010	18											881	11.4903
20	1	18	2010	19											558	10.5523
21	1	19	2010	20											793	9.75705
22	1	20	2010	21											023	9.05087
23	1	21	2010	22											715	8.45506
24	1	22	2010	23											267	7.91852
25	1	23	2010	24	0.710345268	0	0	0	7.099934101	4.404565811	1.936324239	0.071938023	0.526993394	0.713691711	0.44527337	7.53022
26	1	24	2010	25	0.394834012	0	0	0	4.099785805	3.31547451	1.241765022	0.080895737	0.496020496	0.674962342	0.426997066	7.20570

Runoff_All is an important data column: it reports the amount of water (in mm/day) that flows out of the watershed's outlet. When observed data for the watershed is available (and converted to matching units) it can be compared to Runoff_All either graphically (i.e. as in Excel above), or numerically (e.g. calculation of a Nash-

Sutcliffe coefficient) to determine how well the simulated runoff matches reality.
(Unfortunately, we do not have observed runoff data for our example scenario.)

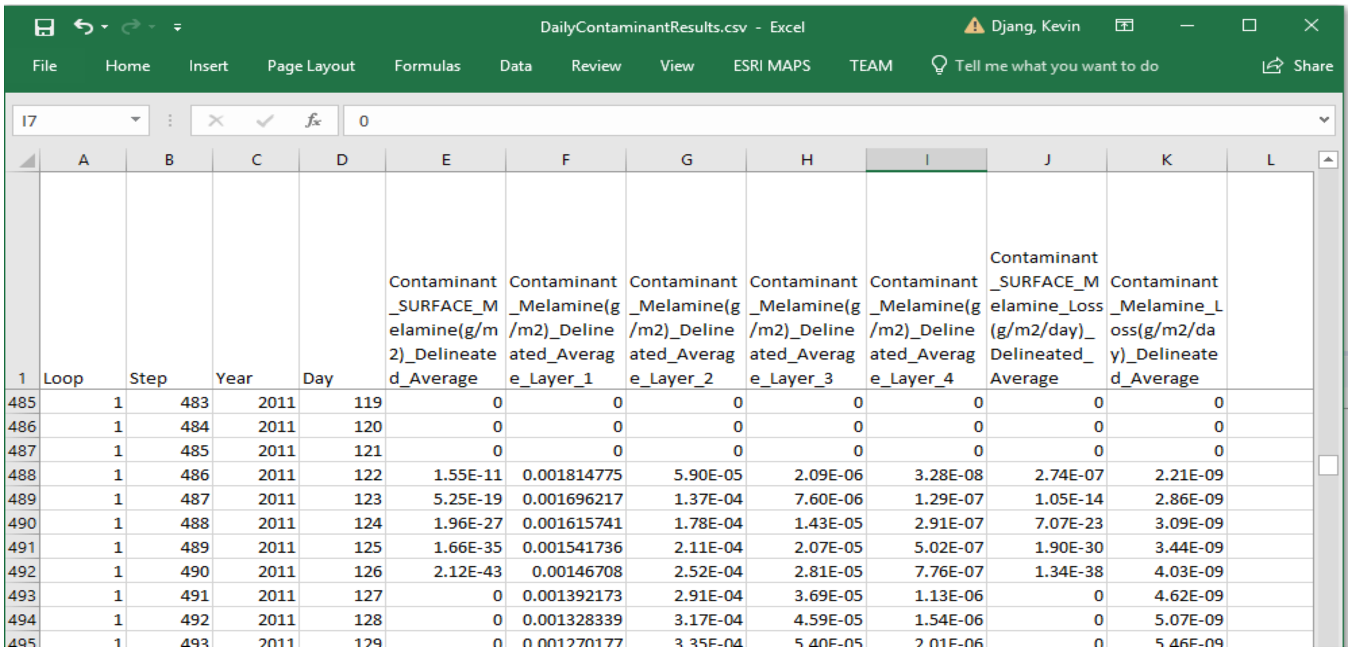
The majority of the data columns in DailyResults are "Delineated_Average" data.
Delineated averages are computed by summing the amounts for every cell in the watershed, then dividing the total by the number of cells in the watershed. The amount reported does not necessarily represent any specific cell amount, only any average.

`DailyContaminantResults.csv`

VELMA simulation configurations that include one or more contaminants will include a ContaminantResults.csv file as part of the output. As its name suggests, this file's structure mimics DailyResult's, but it contains different data in its columns (apart from the first four columns, which duplicate DailyResults).
For each contaminant configured in the simulation, it reports layer-specific, delineated-average daily amounts, along with surface and aggregate (i.e. not layer-specific) soil loss amounts. The loss amounts represent the amount of contaminant (in g/m2) transported out of the watershed each simulation day.

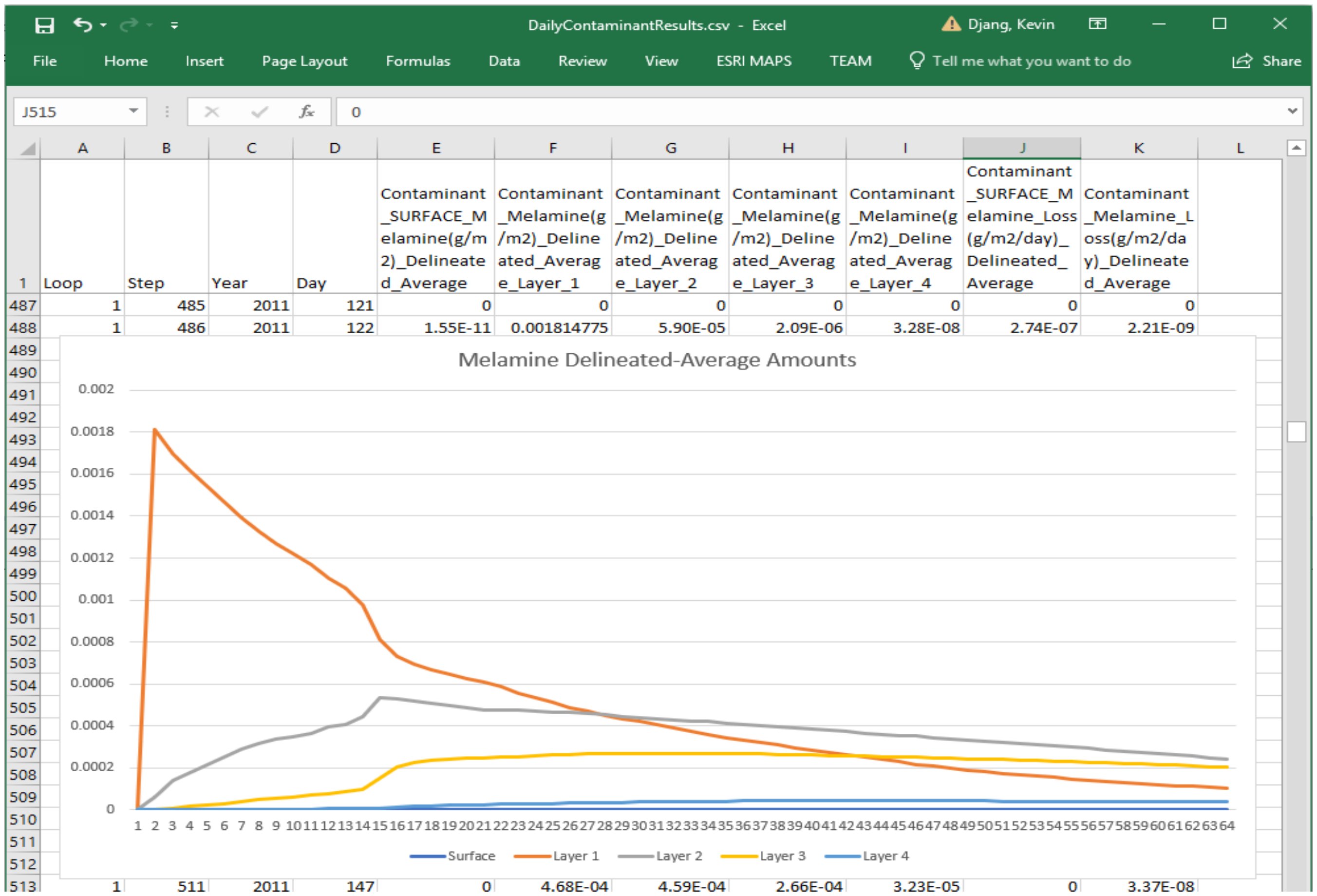
Our example simulation has one contaminant configured (Melamine), with cell-specific deposition during April of the second year of the simulation run, so VELMA generates a DailyContaminantResults file with data columns for Melamine.

Here is a screen capture of the DailyContaminantResults.csv in Excel, showing a few of the days surrounding the deposition:



	A	B	C	D	E	F	G	H	I	J	K
1	Loop	Step	Year	Day	Contaminant_SURFACE_Melamine(g/m2)_Delineated_Average	Contaminant_Melamine(g/m2)_Delineated_Average_Layer_1	Contaminant_Melamine(g/m2)_Delineated_Average_Layer_2	Contaminant_Melamine(g/m2)_Delineated_Average_Layer_3	Contaminant_Melamine(g/m2)_Delineated_Average_Layer_4	Contaminant_SURFACE_Melamine_Loss(g/m2/day)_Delineated_Average	Contaminant_Melamine_Loss(g/m2/day)_Delineated_Average
485	1	483	2011	119	0	0	0	0	0	0	0
486	1	484	2011	120	0	0	0	0	0	0	0
487	1	485	2011	121	0	0	0	0	0	0	0
488	1	486	2011	122	1.55E-11	0.001814775	5.90E-05	2.09E-06	3.28E-08	2.74E-07	2.21E-09
489	1	487	2011	123	5.25E-19	0.001696217	1.37E-04	7.60E-06	1.29E-07	1.05E-14	2.86E-09
490	1	488	2011	124	1.96E-27	0.001615741	1.78E-04	1.43E-05	2.91E-07	7.07E-23	3.09E-09
491	1	489	2011	125	1.66E-35	0.001541736	2.11E-04	2.07E-05	5.02E-07	1.90E-30	3.44E-09
492	1	490	2011	126	2.12E-43	0.00146708	2.52E-04	2.81E-05	7.76E-07	1.34E-38	4.03E-09
493	1	491	2011	127	0	0.001392173	2.91E-04	3.69E-05	1.13E-06	0	4.62E-09
494	1	492	2011	128	0	0.001328339	3.17E-04	4.59E-05	1.54E-06	0	5.07E-09
495	1	493	2011	129	0	0.001270177	3.35E-04	5.40E-05	2.01E-06	0	5.46E-09

Here is an example of using Excel's graphing functions to quickly view how the surface and per-layer delineated-average amounts vary over the first 60 days after deposition:



	A	B	C	D	E	F	G	H	I	J	K
1	Loop	Step	Year	Day	Contaminant _SURFACE_Melamine(g/m2)_Delineated_Average	Contaminant _Melamine(g/m2)_Delineated_Average_Layer_1	Contaminant _Melamine(g/m2)_Delineated_Average_Layer_2	Contaminant _Melamine(g/m2)_Delineated_Average_Layer_3	Contaminant _Melamine(g/m2)_Delineated_Average_Layer_4	Contaminant _SURFACE_Melamine_Loss (g/m2/day)_ Delineated_ Average	Contaminant _Melamine_Loss(g/m2/day)_Delineated_Average
487	1	485	2011	121	0	0	0	0	0	0	0
488	1	486	2011	122	1.55E-11	0.001814775	5.90E-05	2.09E-06	3.28E-08	2.74E-07	2.21E-09
513	1	511	2011	147	0	4.68E-04	4.59E-04	2.66E-04	3.23E-05	0	3.37E-08

Cell-Specific Results

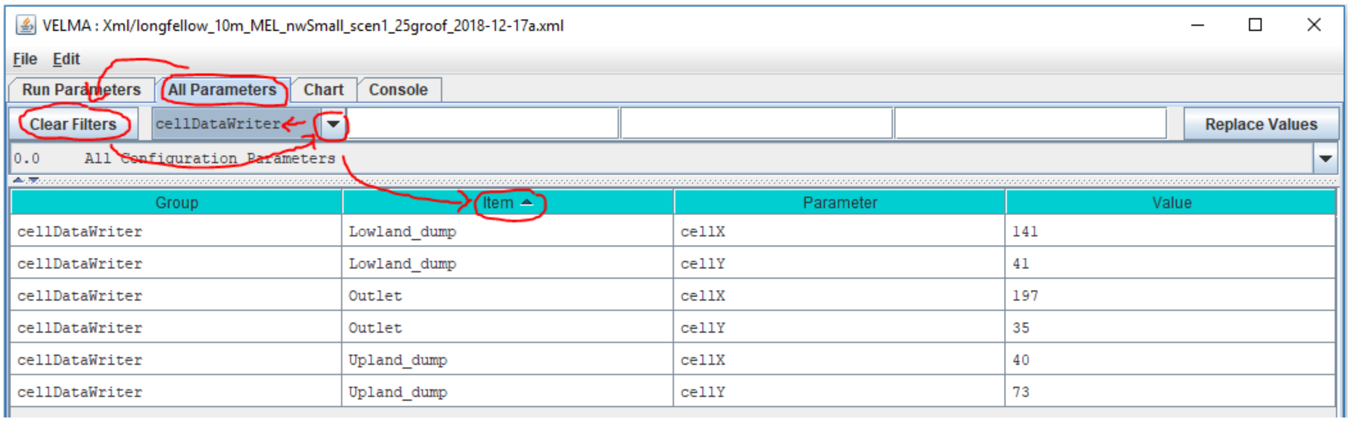
As mentioned above, results in the DailyResults.csv file are mostly delineated averages. To gather cell-specific data, a simulation must specify one or more Cell Data Writers, or Spatial Data Writers as part of the simulation scenario's configuration.

A *Cell Data Writer* configuration tells the VELMA simulator to gather and report a wide variety of data for a specific cell location within the watershed. The results appear in a comma-separated value (.csv) file -- with a file name that includes the cell location -- in the output directory.

A *Spatial Data Writer* configuration tells the VELMA simulator to gather the amounts of one specific data type and report it for every cell. The results appear one or more Grid ASCII (.asc) map files – with file names that include the data type and simulation date-stamp – in the output directory.

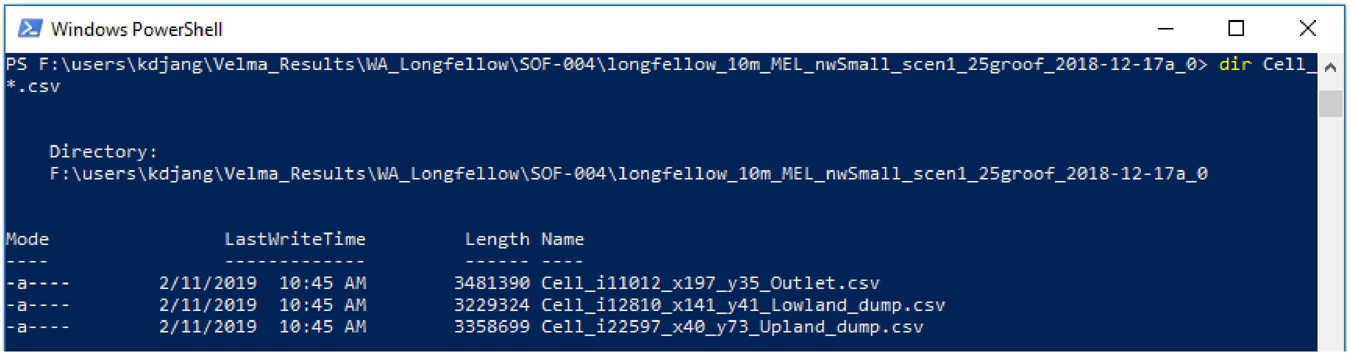
Cell Data Writer Files

Our example simulation includes 3 Cell Data Writer configurations:


VELMA : Xml/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
cellDataWriter
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
cellDataWriter	Lowland_dump	cellX	141
cellDataWriter	Lowland_dump	cellY	41
cellDataWriter	Outlet	cellX	197
cellDataWriter	Outlet	cellY	35
cellDataWriter	Upland_dump	cellX	40
cellDataWriter	Upland_dump	cellY	73

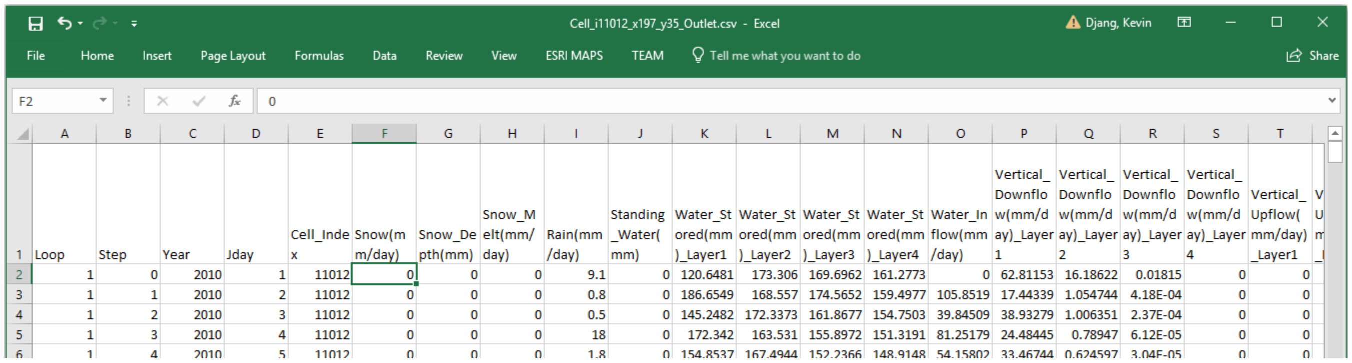
As you can see, each configuration includes an Item name, and x and y cell location parameter values. Each configuration generates cell-specific data in an output directory file with a cell-specific name (you can use the Powershell, as shown below, or Windows Explorer to obtain this information):



```
PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0> dir Cell_
*.csv


    Directory:
    F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        2/11/2019  10:45 AM        3481390 Cell_i11012_x197_y35_Outlet.csv
-a----        2/11/2019  10:45 AM        3229324 Cell_i12810_x141_y41_Lowland_dump.csv
-a----        2/11/2019  10:45 AM        3358699 Cell_i22597_x40_y73_Upland_dump.csv
```

Like DailyResults and DailyContaminantResults .csv files, a row of data in a Cell_ .csv file represents one simulation step, with specific data type amounts for that step in each column. However, unlike DailyResults, the data amounts are not averaged – they are raw amounts – and are specific to a cell location.

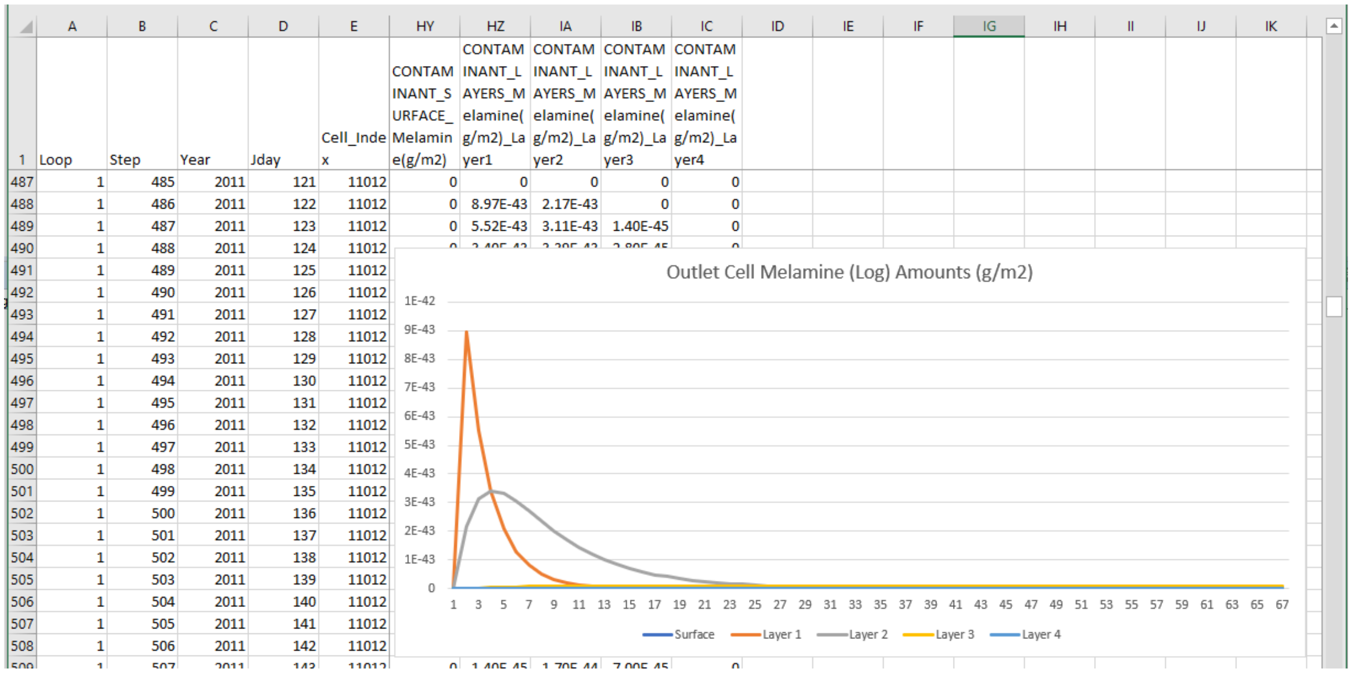
Here is a screen-capture of the upper-right corner of the Output Cell_ results .csv file opened in Excel:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Loop	Step	Year	Jday	Cell_Index	Snow(mm/day)	Snow_Depth(mm)	Snow_Melt(mm/day)	Rain(mm/day)	Standing_Water(mm)	Water_Stored(mm)_Layer1	Water_Stored(mm)_Layer2	Water_Stored(mm)_Layer3	Water_Stored(mm)_Layer4	Water_Inflow(mm/day)	Vertical_Downflow(mm/day)_Layer1	Vertical_Downflow(mm/day)_Layer2	Vertical_Downflow(mm/day)_Layer3	Vertical_Downflow(mm/day)_Layer4	Vertical_Upflow(mm/day)_Layer1
2	1	0	2010	1	11012	0	0	0	9.1	0	120.6481	173.306	169.6962	161.2773	0	62.81153	16.18622	0.01815	0	0
3	1	1	2010	2	11012	0	0	0	0.8	0	186.6549	168.557	174.5652	159.4977	105.8519	17.44339	1.054744	4.18E-04	0	0
4	1	2	2010	3	11012	0	0	0	0.5	0	145.2482	172.3373	161.8677	154.7503	39.84509	38.93279	1.006351	2.37E-04	0	0
5	1	3	2010	4	11012	0	0	0	18	0	172.342	163.531	155.8972	151.3191	81.25179	24.48445	0.78947	6.12E-05	0	0
6	1	4	2010	5	11012	0	0	0	1.8	0	154.8537	167.4944	152.2366	148.9148	54.15802	33.46744	0.624597	3.04E-05	0	0

When a simulation scenario includes Contaminants (as our example scenario does), Cell Data Writer files include rows for the Contaminant lateral flow (the amount hydrologically transferred out of the cell) and pool amounts (the raw amount in each layer of the cell and on the surface).

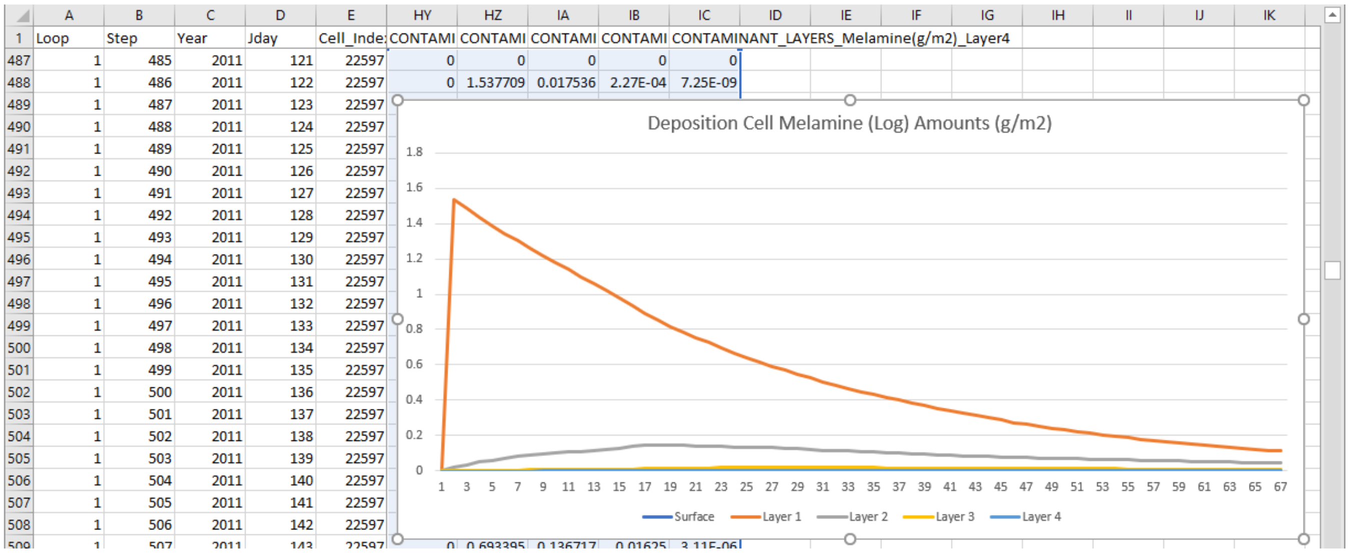
Here is a screen capture of our example scenario's outlet cell Melamine amounts for the first 60 days after deposition:



	A	B	C	D	E	HY	HZ	IA	IB	IC
1	Loop	Step	Year	Jday	Cell_Index	CONTAMINANT_SURFACE_Melamine(g/m2)	CONTAMINANT_LAYERS_Melamine(g/m2)_Layer1	CONTAMINANT_LAYERS_Melamine(g/m2)_Layer2	CONTAMINANT_LAYERS_Melamine(g/m2)_Layer3	CONTAMINANT_LAYERS_Melamine(g/m2)_Layer4
487	1	485	2011	121	11012	0	0	0	0	0
488	1	486	2011	122	11012	0	8.97E-43	2.17E-43	0	0
489	1	487	2011	123	11012	0	5.52E-43	3.11E-43	1.40E-45	0
490	1	488	2011	124	11012	0	[illegible]	[illegible]	[illegible]	0
491	1	489	2011	125	11012					
492	1	490	2011	126	11012					
493	1	491	2011	127	11012					
494	1	492	2011	128	11012					
495	1	493	2011	129	11012					
496	1	494	2011	130	11012					
497	1	495	2011	131	11012					
498	1	496	2011	132	11012					
499	1	497	2011	133	11012					
500	1	498	2011	134	11012					
501	1	499	2011	135	11012					
502	1	500	2011	136	11012					
503	1	501	2011	137	11012					
504	1	502	2011	138	11012					
505	1	503	2011	139	11012					
506	1	504	2011	140	11012					
507	1	505	2011	141	11012					
508	1	506	2011	142	11012					
509	1	507	2011	143	11012	0	[illegible]	[illegible]	[illegible]	0

Compare this with the DailyResults delineated-averaged graph of the *average* Melamine across the watershed. For the same time span, graphs show some general similiarity, but the specific amounts at the Outlet are distinct from the delineated-average of the watershed as a whole.

To further emphasize the distinction, here is a graph of the same simulation time span, but for one of the other Cell Data Writers configurations in our example. This Cell Data Writer was configured to capture data at one of the cells Melamine is deposited (unlike the outlet cell):

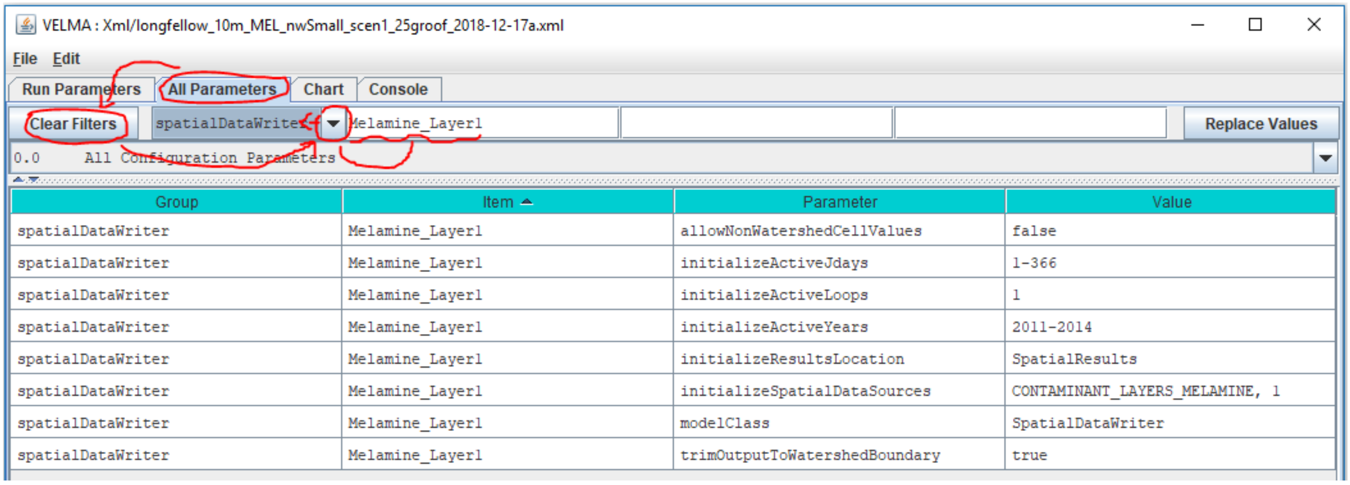


	A	B	C	D	E	HY	HZ	IA	IB	IC
1	Loop	Step	Year	Jday	Cell_Inde:	CONTAMI	CONTAMI	CONTAMI	CONTAMI	CONTAMINANT_LAYERS_Melamine(g/m2)_Layer4
487	1	485	2011	121	22597	0	0	0	0	0
488	1	486	2011	122	22597	0	1.537709	0.017536	2.27E-04	7.25E-09
489	1	487	2011	123	22597					
490	1	488	2011	124	22597					
491	1	489	2011	125	22597					
492	1	490	2011	126	22597					
493	1	491	2011	127	22597					
494	1	492	2011	128	22597					
495	1	493	2011	129	22597					
496	1	494	2011	130	22597					
497	1	495	2011	131	22597					
498	1	496	2011	132	22597					
499	1	497	2011	133	22597					
500	1	498	2011	134	22597					
501	1	499	2011	135	22597					
502	1	500	2011	136	22597					
503	1	501	2011	137	22597					
504	1	502	2011	138	22597					
505	1	503	2011	139	22597					
506	1	504	2011	140	22597					
507	1	505	2011	141	22597					
508	1	506	2011	142	22597					
509	1	507	2011	143	22597	0	0.693395	0.136717	0.01625	3.11E-06

For further information about Cell Data Writers, refer to:
Tutorial D.8 – Cell Data Writer Configuration
in the "HowTo" documentation folder for VELMA 2.1.

Spatial Data Writer Files

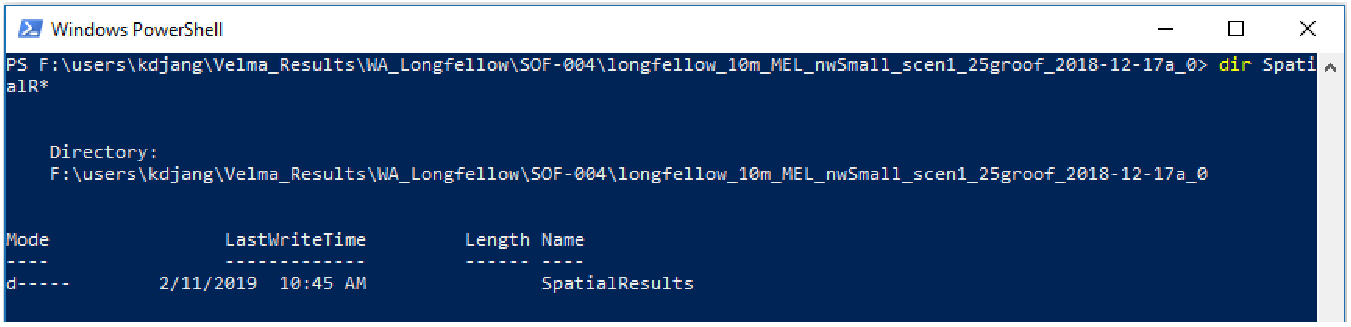
Our example simulation includes several Spatial Data Writer configurations. The following screen capture shows the "All Parmeters" table filtered to display the parameters for one of them:



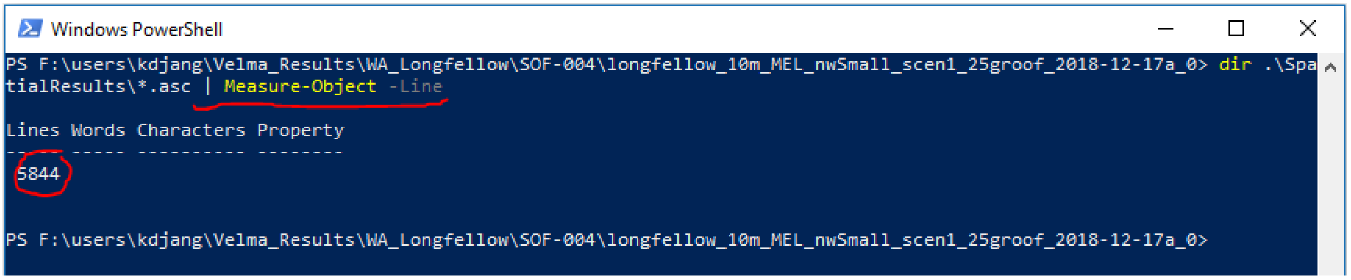
Group	Item	Parameter	Value
spatialDataWriter	Melamine_Layer1	allowNonWatershedCellValues	false
spatialDataWriter	Melamine_Layer1	initializeActiveJdays	1-366
spatialDataWriter	Melamine_Layer1	initializeActiveLoops	1
spatialDataWriter	Melamine_Layer1	initializeActiveYears	2011-2014
spatialDataWriter	Melamine_Layer1	initializeResultsLocation	SpatialResults
spatialDataWriter	Melamine_Layer1	initializeSpatialDataSources	CONTAMINANT_LAYERS_MELAMINE, 1
spatialDataWriter	Melamine_Layer1	modelClass	SpatialDataWriter
spatialDataWriter	Melamine_Layer1	trimOutputToWatershedBoundary	true

Configuring a SpatialDataWriter requires more parameters than a Cell DataWriter, because the data pool and the simulation steps to capture must be specified for the simulator. This Spatial Data Writer above catpures the Melamine amounts in soil layer 1 (per the `initializeSpatialDataSources` parameter) for each day (simulation step) in the range of years [2011, 2014]. It will write the data as series of uniquely-named .asc files to a directory named "SpatialResults" (the `initializeResultsLocation` parameter's value) which the simulator will create as a subdirectory of primary output directory during the simulation run.
(All of the Spatial Data Writer in our example simulation scenario are configured to report the same simulation states to the same `initializeResultsLocation`.)

Checking our example simulation output directory confirms that "SpatialResults" exists, and that it contains (a lot) of *.asc files:



```
PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0> dir Spati
alR*


    Directory:
    F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        2/11/2019  10:45 AM                SpatialResults
```



```
PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0> dir .\Spa
tialResults\*.asc | Measure-Object -Line

Lines Words Characters Property
----- ----- ---------- --------
 5844


PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0>
```

(The "`| Measure-Object -Line`*" portion of the PowerShell command to list all *.asc files in .\SpatialResults directory reidirects the command to report the number of files, instead of listing their names.)*

The fact that we've generated 5,844 .asc files of spatially-explicit (i.e. map) data highlights a rule of thumb about Spatial Data Writers: each file written reports amounts for the every cell in the watershed, but only for one simulation step. To record one year's worth of Spatial Data for one data pool requires writing 365 (or 366 in a leap year) files, and this multiplies as more Spatial Data Writers are configured for more data pools. If you need spatially-expicit maps of the simulation's runtime behavior, this is worth the running time and disk space it incurs – however, it's prudent to ensure your computer has enough disk space before you start.

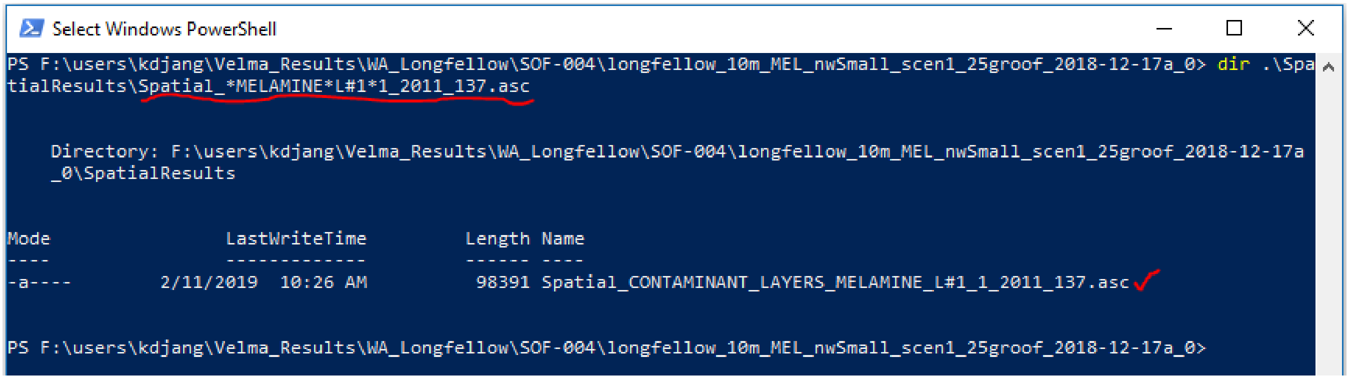
You can view the .asc files created by Spatial Data Writers in ESRI's GIS tools, or any GIS tools that support the ESRI Grid ASCII file format. You can also view them in VISTAs, or JPDEM, or open them in a simple text editor (e.g. Microsoft's Notepad) and cut-paste the contents into Excel. This last option is unwieldy for anything but small (in terms of row and column count) .asc files.

Spatial Data Writer file names are prefixed with the text "Spatial_", and have the dot-extension ".asc". Their names include the name of the data they report, and the loop, year and Julian day of the simulation run that the data was reported for.

As an example of finding specific file: referring to the contaminant data graph from the deposition cell's Cell Data Writer file in the previous section, Melamine first appears in soil layer 1 at the deposition cell on Julian day 122 of 2011. To look at the state of Melamine amounts in soil layer 1 fifteen days after that (Julian day 137), we need to find the .asc file with the following file name characteristics:

- The prefix "Spatial_"
- followed by the name of the data ("MELAMINE")
- followed by a Layer indicator ("L#1")
- ending with the loop, year and Julian day values ("1_2011_137" in this case)
- with ".asc" as the dot-extension suffix.

Without knowing the exact filename, this information should be enough to find the file we want, using the dir command, in a PowerShell window:



```
PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0> dir .\Spa
tialResults\Spatial_*MELAMINE*L#1*1_2011_137.asc


    Directory: F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a
    _0\SpatialResults


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        2/11/2019  10:26 AM          98391 Spatial_CONTAMINANT_LAYERS_MELAMINE_L#1_1_2011_137.asc ✓


PS F:\users\kdjang\Velma_Results\WA_Longfellow\SOF-004\longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a_0>
```

Changing A VELMA Simulation Configuration

Section V. Topics

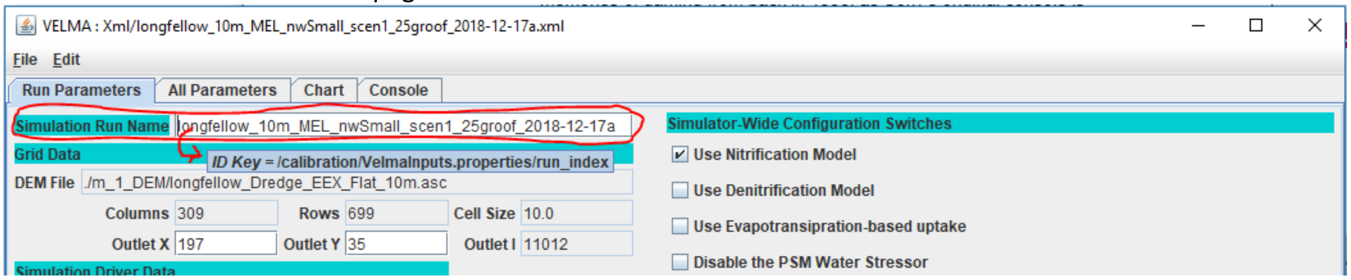
- Create a Copy
- Change the Copy
- Save Your Work
- Remove Obsolete Original Parameters
- Save Your Work(!)

This section demonstrates how to modify and existing VELMA simulation configuration to create a related, but different scenario: we'll use the original example introduced in this guide as a starting point, then change how and where its Melamine contaminant is deposited during the simulation run.

Create a Copy of the Original Example Simulation Configuration

1. Start the VELMA GUI and load the
   `longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml`
   configuration file, as described in parts II and III of this guide, however do not start the simulation running.

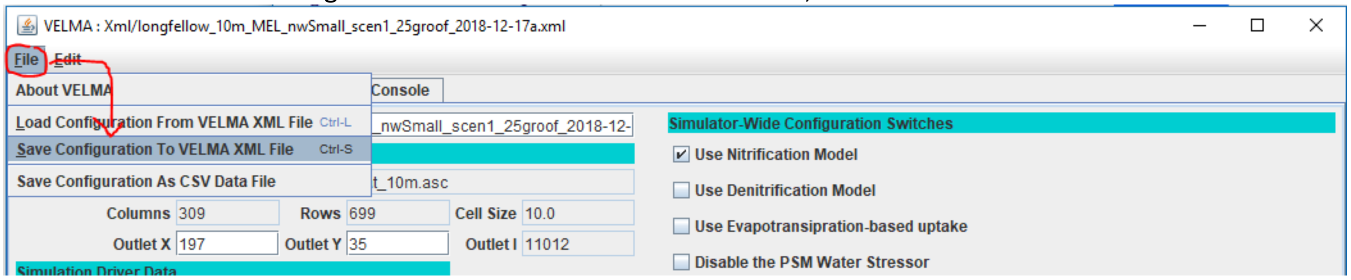
2. The name of the example simulation is in the "Simulation Run Name" text field, in the upper-left corner of the GUI's "Run Parameters" page.


VELMA : Xml/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Grid Data
Simulation Driver Data
Simulator-Wide Configuration Switches
Simulation Run Name longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a
ID Key = /calibration/VelmaInputs.properties/run_index
DEM File ./m_1_DEM/longfellow_Dredge_EEX_Flat_10m.asc
Columns 309 Rows 699 Cell Size 10.0
Outlet X 197 Outlet Y 35 Outlet I 11012
✓ Use Nitrification Model
☐ Use Denitrification Model
☐ Use Evapotranspiration-based uptake
☐ Disable the PSM Water Stressor

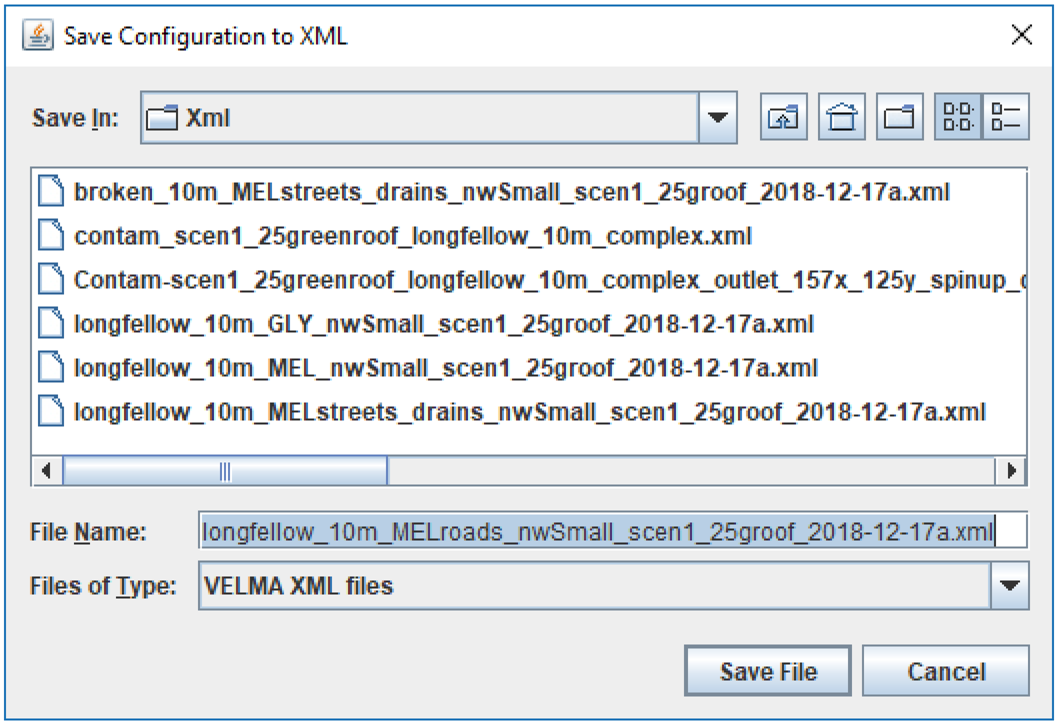
   Pro tip: click into the text box, then leave your mouse cursor hovering over it – after few seconds, a pop-up tool-tip message will appear, displaying the parameter's name.

3. Edit the value of the Simulation Run Name from:
   `longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a`
   to
   `longfellow_10m_MELroads_nwSmall_scen1_25groof_2018-12-17a`
   and then press the Tab key (The Return/Enter key will not do, you must press Tab).

4. Click the File →"Save Configuration to VELMA XML File" menu item,


VELMA : Xml/longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
File Edit
About VELMA
Load Configuration From VELMA XML File Ctrl-L
Save Configuration To VELMA XML File Ctrl-S
Save Configuration As CSV Data File
Console
Simulation Driver Data
Simulator-Wide Configuration Switches
_nwSmall_scen1_25groof_2018-12-
t_10m.asc
Columns 309 Rows 699 Cell Size 10.0
Outlet X 197 Outlet Y 35 Outlet I 11012
✓ Use Nitrification Model
☐ Use Denitrification Model
☐ Use Evapotranspiration-based uptake
☐ Disable the PSM Water Stressor

   which opens the "Save Configuration to XML" dialog window:



broken_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a.xml
contam_scen1_25greenroof_longfellow_10m_complex.xml
Contam-scen1_25greenroof_longfellow_10m_complex_outlet_157x_125y_spinup_
longfellow_10m_GLY_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_MEL_nwSmall_scen1_25groof_2018-12-17a.xml
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a.xml

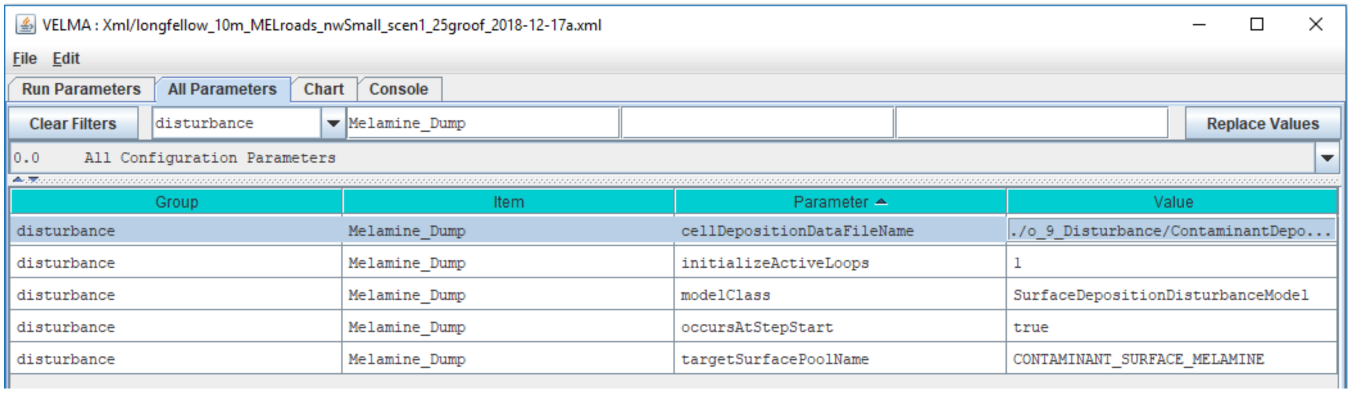
File Name: longfellow_10m_MELroads_nwSmall_scen1_25groof_2018-12-17a.xml
Files of Type: VELMA XML files

The dialog should open to the same directory that you loaded the original example configuration .xml file from, and the File Name should already contain the new Simulation Run Name you specified in step 3 above.
Click “Save File”.
You now have a copy of the original example .xml file saved in the same directory under the new Simulation Run Name.

Change the Copy of the Simulation Configuration to Make a New Scenario

The original example scenario deposited 10g of Melamine onto the surface of one watershed cell on the 122nd day of 2011, and onto another watershed cell 60 days later in the same year. It used a Surface Deposition Disturbance configuration to do this.

To view the Deposition Disturbance’s configuration, click the “All Parameters” tab, then the Clear Filters button, then click the drop-down button and select “disturbance” in the Group selector filter. Finally, type `Melamine_Dump` into the Item filter text field:

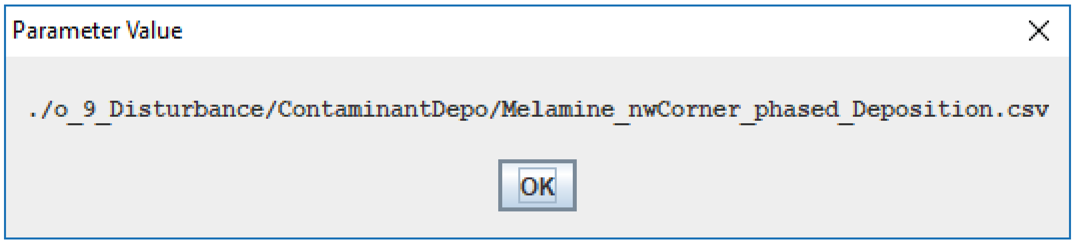


disturbance | Melamine_Dump

0.0 All Configuration Parameters

Group	Item	Parameter	Value
disturbance	Melamine_Dump	cellDepositionDataFileName	./o_9_Disturbance/ContaminantDepo...
disturbance	Melamine_Dump	initializeActiveLoops	1
disturbance	Melamine_Dump	modelClass	SurfaceDepositionDisturbanceModel
disturbance	Melamine_Dump	occursAtStepStart	true
disturbance	Melamine_Dump	targetSurfacePoolName	CONTAMINANT_SURFACE_MELAMINE

Although this configuration has relatively few parameters, it is not evident where the deposition location and dates mentioned earlier are specified. That's because they are "hidden" inside file specified by the `cellDepositionDataFileName` parameter: its value is the name of a file that contains the deposition details:



```
./o_9_Disturbance/ContaminantDepo/Melamine_nwCorner_phased_Deposition.csv
```

Printing the contents of the file reveals the schedule:

```
22597,2011,122,10
12810,2012,182,10
```

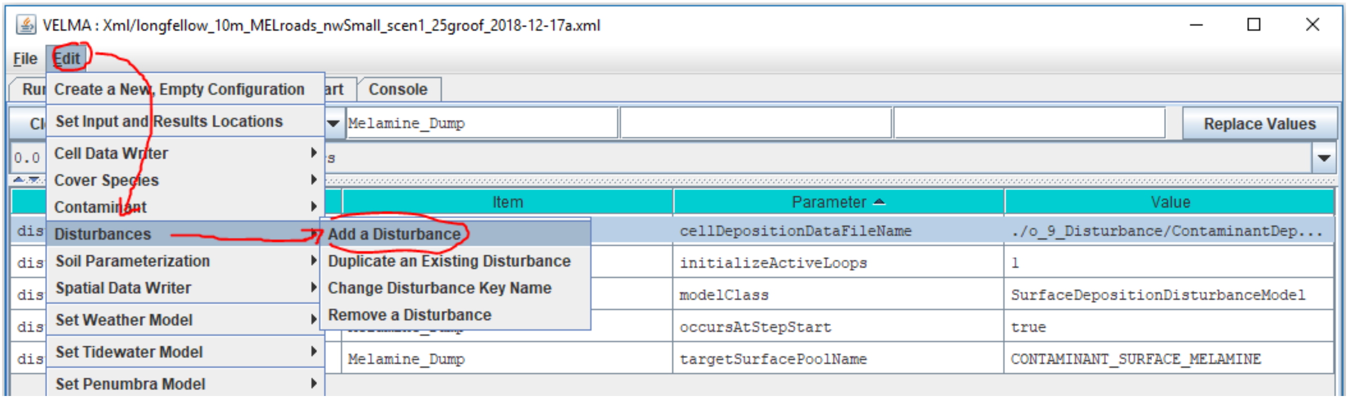
The first number in each row is the linear index of a cell in the watershed.
The trio of numbers that follow indicate the year, Julian day, and the amount of contaminant to deposit on the cells surface.

The deposition data file can contain many more cells than this example, and each cell can have multiple trios of (year, jday, amounts) associated with it. This allows configuration of a wide variety of deposition scenarios.

However, VELMA provides another way to effect deposition of contaminants onto the cells of a simulation's watershed: specify a map (an .asc file) with deposition amounts, as well as when the amounts in the map should be applied to the cells. This kind of disturbance configuration is called a "Set Spatial Data by Map" disturbance, a long-winded name providing a good idea of its function.

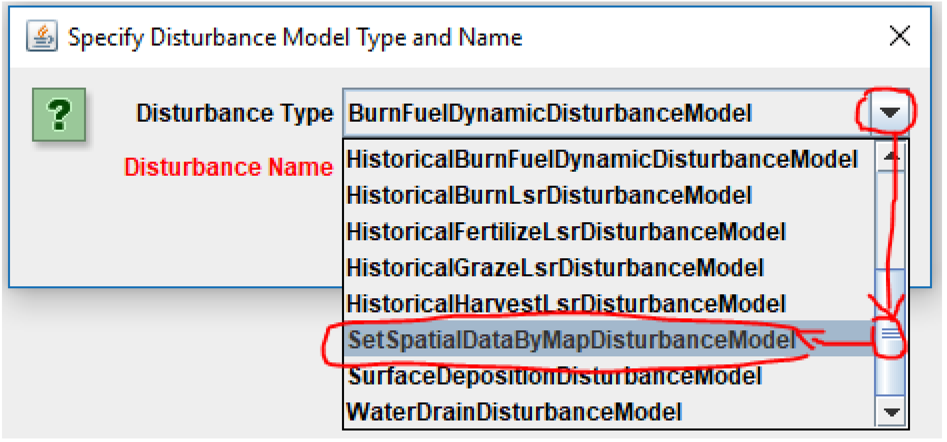
VELMA allows multiple disturbance configurations within one overall simulation configuration, so let's add the new disturbance before removing the old.

1. In the VELMA GUI's All Parameters tab, click the Edit →Disturbances → "Add a Disturbance" sub-menu item.

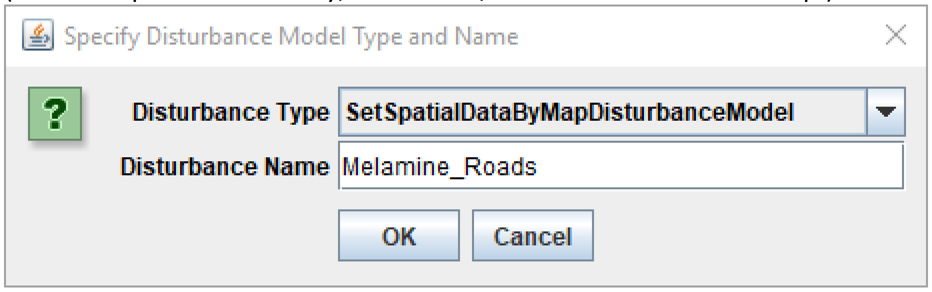


Item	Parameter	Value
	cellDepositionDataFileName	./o_9_Disturbance/ContaminantDep...
	initializeActiveLoops	1
	modelClass	SurfaceDepositionDisturbanceModel
	occursAtStepStart	true
Melamine_Dump	targetSurfacePoolName	CONTAMINANT_SURFACE_MELAMINE

2. In the “Specify Disturbance Model Type and Name” dialog, click the drop-down list of the “Disturbance Type” selector and select “SetSpatialDataByMapDisturbanceModel”.
(You may have to sroll down in the list to see it):


Specify Disturbance Model Type and Name
Disturbance Type
BurnFuelDynamicDisturbanceModel
Disturbance Name
HistoricalBurnFuelDynamicDisturbanceModel
HistoricalBurnLsrDisturbanceModel
HistoricalFertilizeLsrDisturbanceModel
HistoricalGrazeLsrDisturbanceModel
HistoricalHarvestLsrDisturbanceModel
SetSpatialDataByMapDisturbanceModel
SurfaceDepositionDisturbanceModel
WaterDrainDisturbanceModel

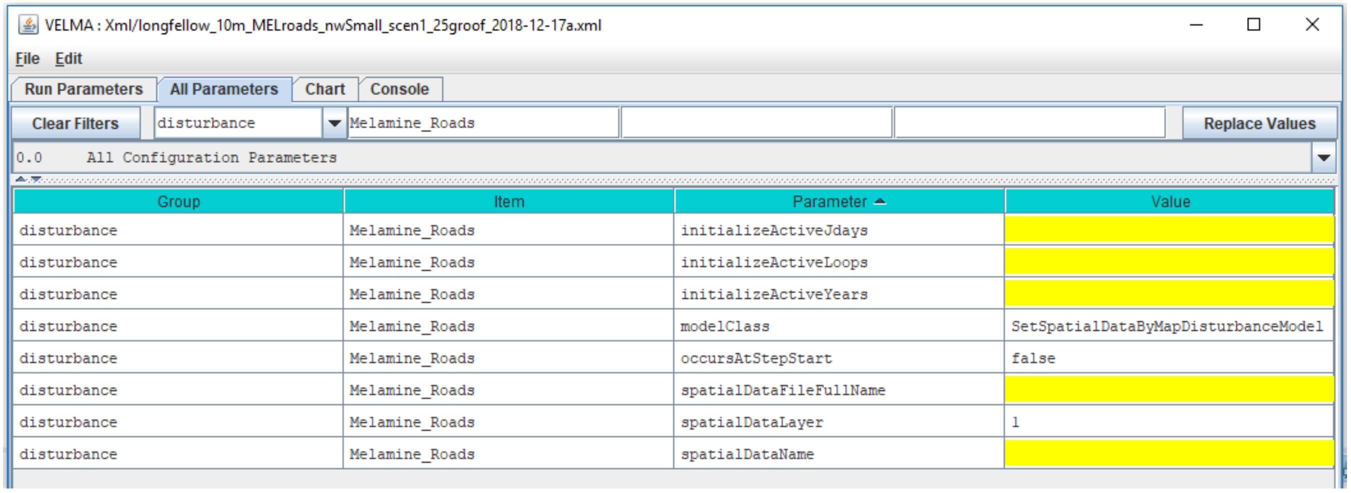
3. Specify the Disturbance Name as “Melamine_Roads”, then press the Tab key.
(You must press the Tab key, the Return/Enter won’t do for this step.)



Disturbance Type: SetSpatialDataByMapDisturbanceModel
Disturbance Name: Melamine_Roads

At this point, the “Disturbance Name” label should stop displaying in red.
Click the OK button.

4. The All Parameters tab panel should be automatically filtered to display the new disturbance parameters it’s parameterization table:


VELMA : Xml/longfellow_10m_MELroads_nwSmall_scen1_25groof_2018-12-17a.xml

Group	Item	Parameter	Value
disturbance	Melamine_Roads	initializeActiveJdays	
disturbance	Melamine_Roads	initializeActiveLoops	
disturbance	Melamine_Roads	initializeActiveYears	
disturbance	Melamine_Roads	modelClass	SetSpatialDataByMapDisturbanceModel
disturbance	Melamine_Roads	occursAtStepStart	false
disturbance	Melamine_Roads	spatialDataFileFullName	
disturbance	Melamine_Roads	spatialDataLayer	1
disturbance	Melamine_Roads	spatialDataName	

5. Some parameter's values are filled in by default, while others are highlighted in yellow to indicate that their values are missing.
   Click into the "Value" column fields of the following parameters and set their values as follows:
   - `initializeActiveJdays = 122`
   - `initializeActiveLoops = 1`
   - `initializeActiveYears = 2011`

   Do not change the modelClass parameter's value: its default is correct as-is.

   - occursAtStepStart = true (edit the default "false" to "true")

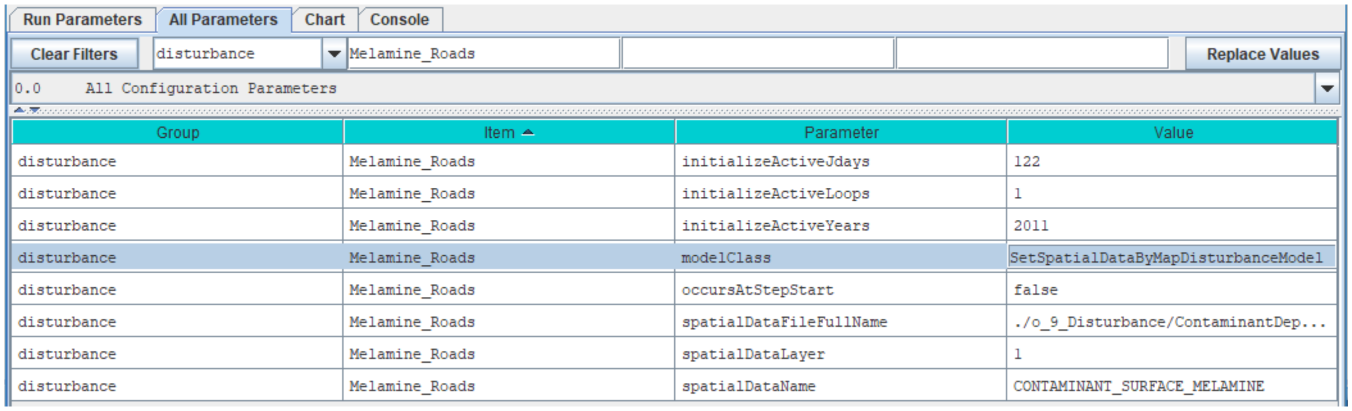
   The above parameterization causes deposition on the same date as the first of the two cells in the original example's Surface Deposition Disturbance configuration.

   - spatialDataFileFullName =
     `./o_9_Disturbance/ContaminantDepo/AllRoadCells_1d0.asc`

   The above file and location are part of the example simulation's DataInputs files collection.
   The AllRoadCells_1d0.asc file specifies 1.0 (grams) for every cell identified as a road cell within the simulation watershed's DEM map area, and 0.0 grams for every other cell in the map.
   Preparing this map is not a trivial undertaking, and requires GIS tools and expertise.
   Refer to other sources for further information.
   - `spatialDataLayer = 1`
   - `spatialDataName = CONTAMINANT_SURFACE_MELAMINE`

   The parameters above tell the disturbance which data pool to deposit amounts into, and which layer of the pool.

   The disturbance's parameterization should now look like this:



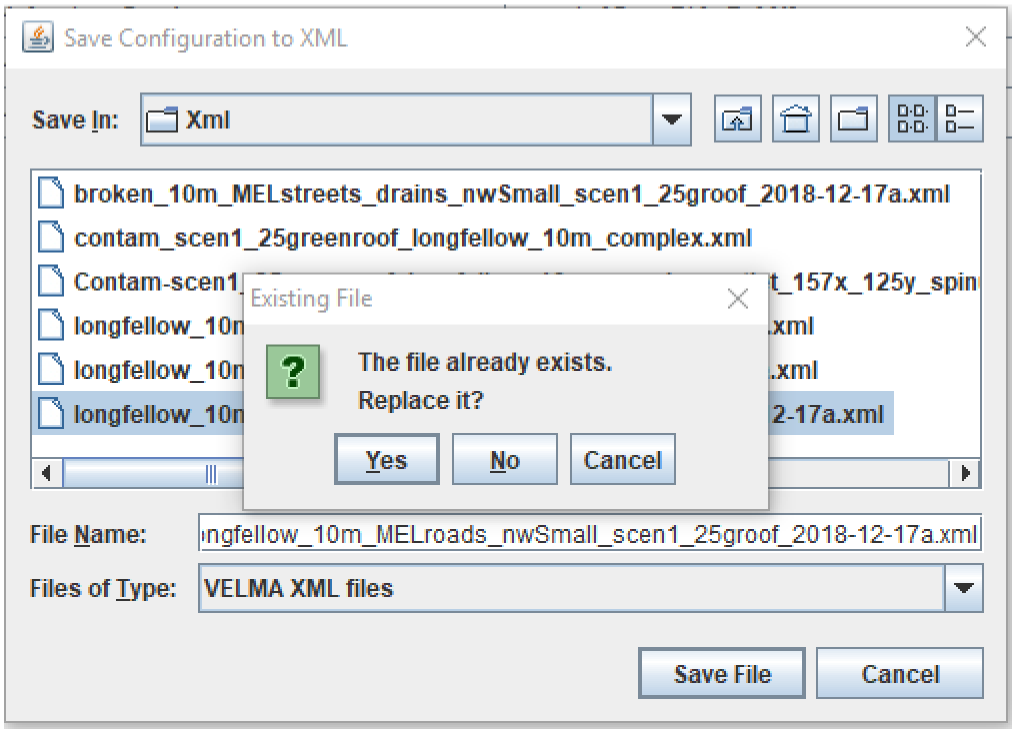
disturbance | Melamine_Roads

0.0 All Configuration Parameters

Group	Item	Parameter	Value
disturbance	Melamine_Roads	initializeActiveJdays	122
disturbance	Melamine_Roads	initializeActiveLoops	1
disturbance	Melamine_Roads	initializeActiveYears	2011
disturbance	Melamine_Roads	modelClass	SetSpatialDataByMapDisturbanceModel
disturbance	Melamine_Roads	occursAtStepStart	false
disturbance	Melamine_Roads	spatialDataFileFullName	./o_9_Disturbance/ContaminantDep...
disturbance	Melamine_Roads	spatialDataLayer	1
disturbance	Melamine_Roads	spatialDataName	CONTAMINANT_SURFACE_MELAMINE

Save Your Work!

Take a moment to save the new scenario’s current state, by repeating step 4 from the “Create a Copy …” instructions above. This time, when you click the “Save File” button, the GUI will warn you that you are about to overwrite an existing .xml file.



broken_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a.xml
contam_scen1_25greenroof_longfellow_10m_complex.xml
Contam-scen1 ... t_157x_125y_spin
longfellow_10m ... xml
longfellow_10m ... .xml
longfellow_10m ... 2-17a.xml

The file already exists.
Replace it?

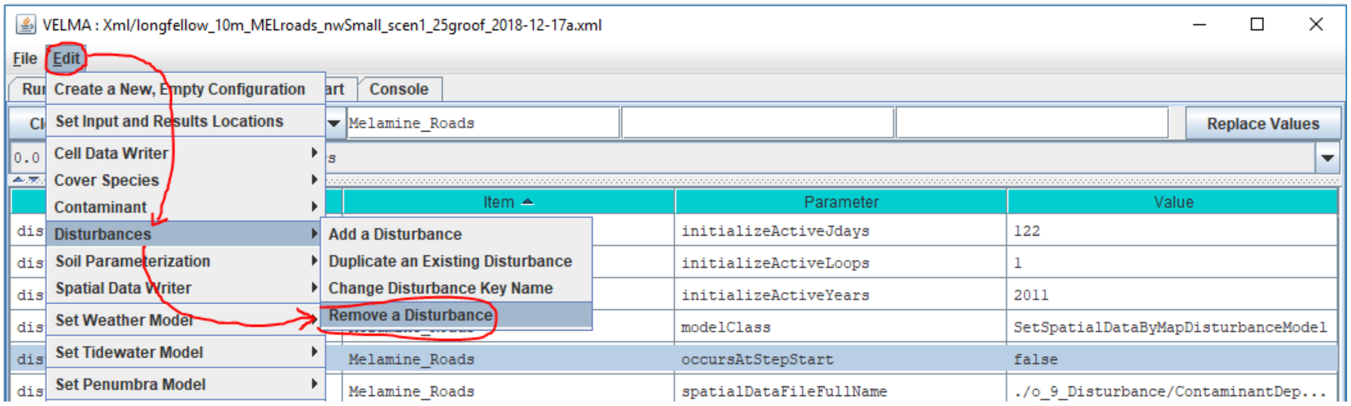
File Name: ngfellow_10m_MELroads_nwSmall_scen1_25groof_2018-12-17a.xml
Files of Type: VELMA XML files

That’s what you want in this case, so click “Yes” to save/overwrite and proceed.

Remove the Original Example’s Surface Deposition Disturbance From the New Scenario

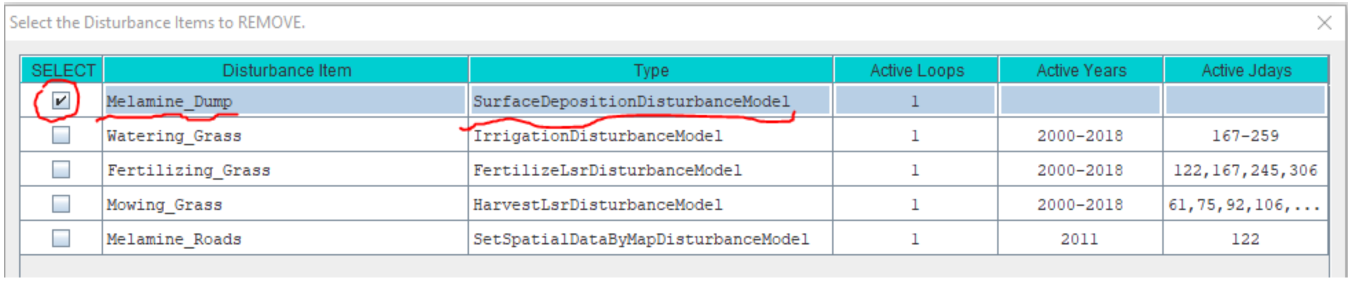
Now that we’ve added a new contaminant deposition (by way of the Set-by-Map disturbance) to the new scenario, we can remove the original scenario’s deposition disturbance.

1. Click the Edit → Disturbances → "Remove a Disturbance" sub-menu item:



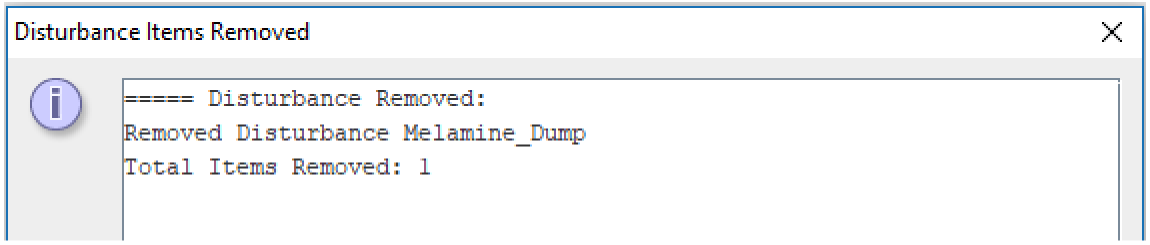
Item	Parameter	Value
	initializeActiveJdays	122
	initializeActiveLoops	1
	initializeActiveYears	2011
	modelClass	SetSpatialDataByMapDisturbanceModel
Melamine_Roads	occursAtStepStart	false
Melamine_Roads	spatialDataFileFullName	./o_9_Disturbance/ContaminantDep...

2. In the "Select the Disturbance Items to REMOVE", click the checkbox for the "Melamine_Dump" SurfaceDepositionDisturbanceModel in the SELECT column of the table:


Select the Disturbance Items to REMOVE.

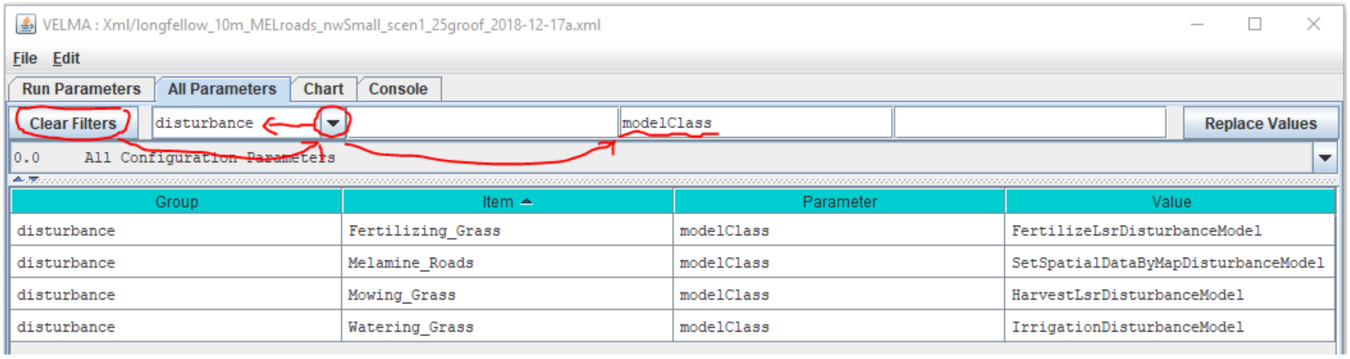
SELECT	Disturbance Item	Type	Active Loops	Active Years	Active Jdays
☑	Melamine_Dump	SurfaceDepositionDisturbanceModel	1		
☐	Watering_Grass	IrrigationDisturbanceModel	1	2000-2018	167-259
☐	Fertilizing_Grass	FertilizeLsrDisturbanceModel	1	2000-2018	122,167,245,306
☐	Mowing_Grass	HarvestLsrDisturbanceModel	1	2000-2018	61,75,92,106,...
☐	Melamine_Roads	SetSpatialDataByMapDisturbanceModel	1	2011	122

Then click the OK button at the bottom of the dialog box, and the OK button at the bottom of the subsequent confirmation box:


Disturbance Items Removed

```
===== Disturbance Removed:
Removed Disturbance Melamine_Dump
Total Items Removed: 1
```

3. You can confirm that the original "Melamine_Dump" disturbance has been removed by click "Clear Filters" then selecting the "disturbance" drop-down filter, and typing `modelClass` in the Parameter filter text field:


VELMA : Xml/longfellow_10m_MELroads_nwSmall_scen1_25groof_2018-12-17a.xml
File
Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
disturbance
modelClass
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
disturbance	Fertilizing_Grass	modelClass	FertilizeLsrDisturbanceModel
disturbance	Melamine_Roads	modelClass	SetSpatialDataByMapDisturbanceModel
disturbance	Mowing_Grass	modelClass	HarvestLsrDisturbanceModel
disturbance	Watering_Grass	modelClass	IrrigationDisturbanceModel

Save Your Work!

Repeat step 4 from the “Create a Copy …” instructions above.

Run the New Scenario

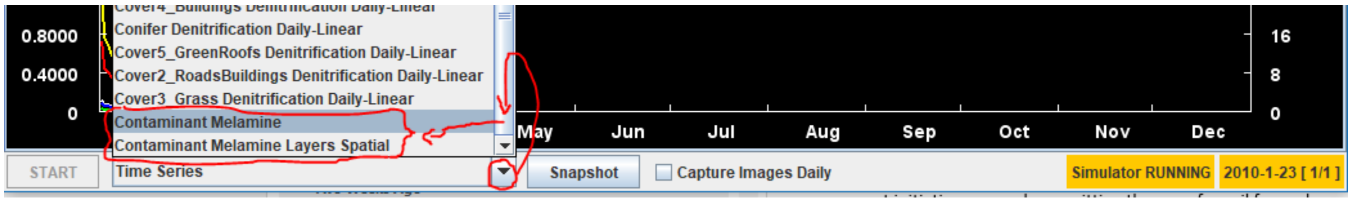
Section VI Topics:

- Start the New Scenario
- Contaminant-specific Runtime Displays
- Comparison of Results

At this point you can click the Start button and run the new scenario, or, if you closed the VELMA GUI after saving the new scenario to an .xml file, re-start the GUI and re-load the new scenario, then start the scenario running.

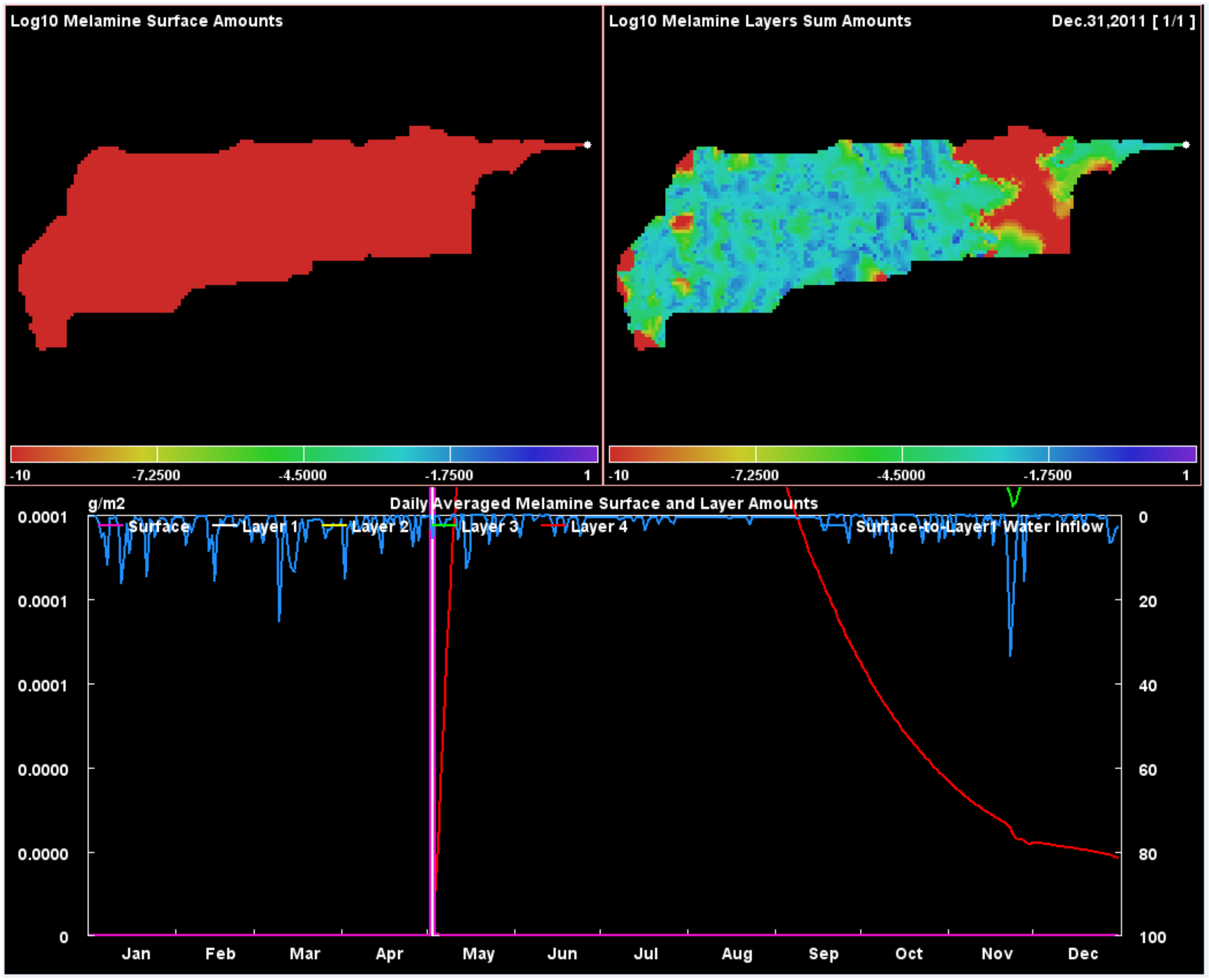
Contaminant Runtime Displays

When the GUI is running a scenario that includes one or more contaminant configurations, the GUI provides two types of displays for charting contaminant amounts during the simulation run:


0.8000
0.4000
0
Cover4_Buildings Denitrification Daily-Linear
Conifer Denitrification Daily-Linear
Cover5_GreenRoofs Denitrification Daily-Linear
Cover2_RoadsBuildings Denitrification Daily-Linear
Cover3_Grass Denitrification Daily-Linear
Contaminant Melamine
Contaminant Melamine Layers Spatial
Time Series
START
Snapshot
Capture Images Daily
Simulator RUNNING
2010-1-23 [1/1]
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
16
8
0

When the new scenario simulation run reaches the 2nd year (2011), select “Contaminant Melamine”.

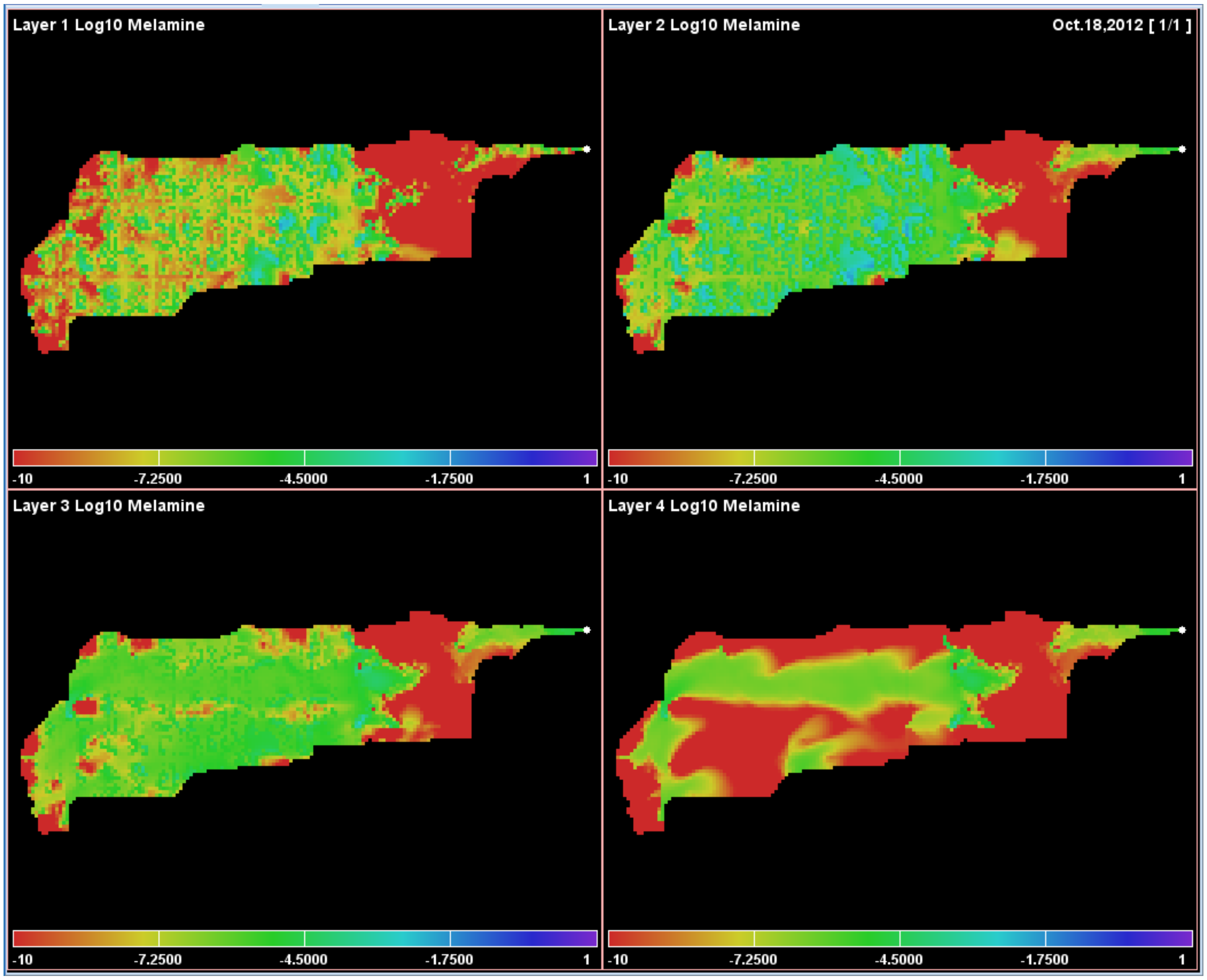
The “Contaminant Melamine” display shows spatially-explicit contaminant amounts for the surface and for the sum of the amounts in the soil layers, as well as the daily, delineated-average amount in the watershed:



(The Daily Averaged values displayed in the bottom chart are the same values reported for surface and layers in the DailyContaminantsResults.csv file.)

Notice that the spatial (map) portions of the display chart the Log10(amount) value per cell.
The total range of amounts can be several orders of magnitude. Displaying using a log scale provides a color ramp that is easier to see.

When the new scenario simulation run reaches the middle of the 3rd year (2012), select “Contaminant Melamine Layers Spatial”. This display shows spatially-explicit, layer-specific contaminant amounts for each of the four soil layers:

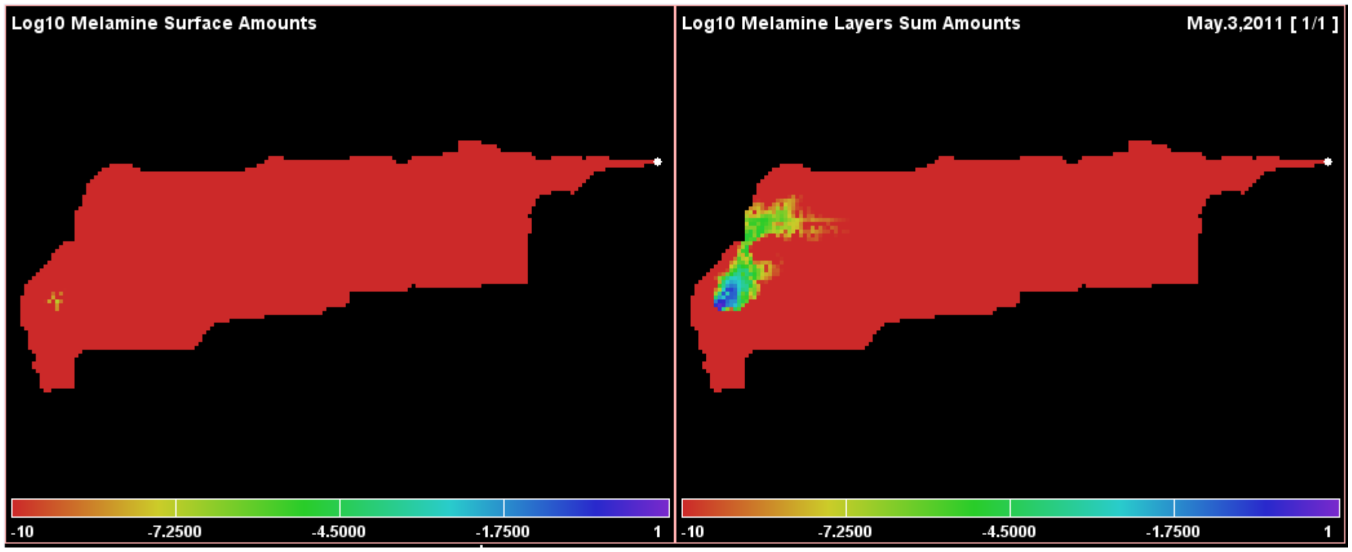


A Quick Comparison / Contrast Between the Original and Modified (New) Scenarios

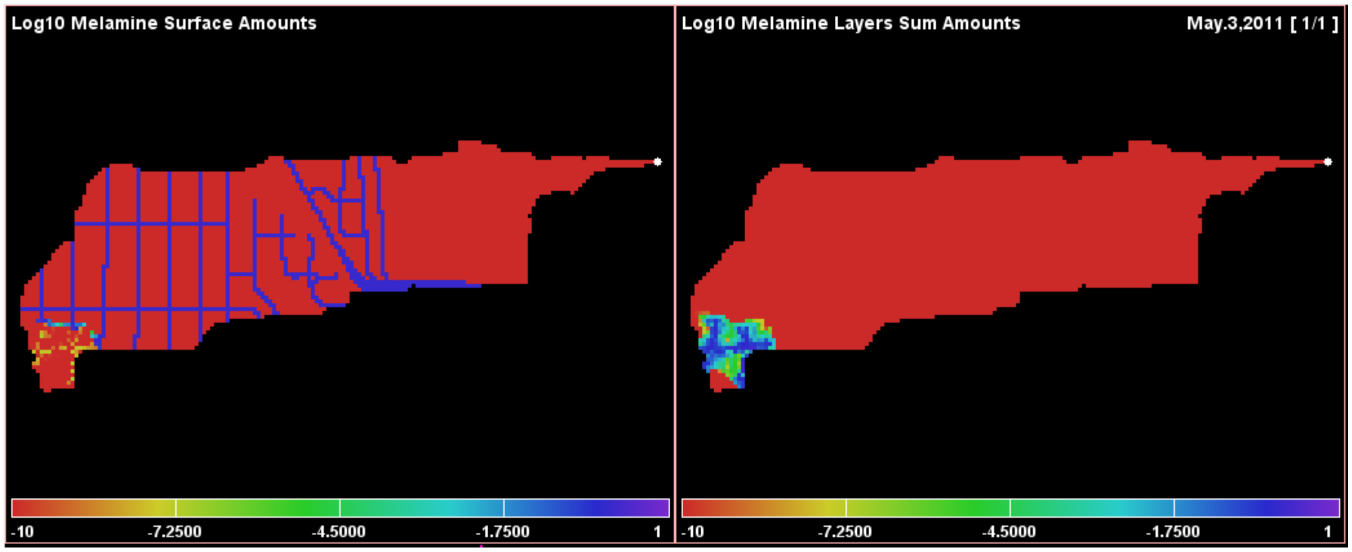
The difference between our two example simulation scenarios is where we deposit the contaminant onto the surface of the watershed’s cells. However, in both scenarios, the first deposition occurs the same day of 2011.

With that in mind, here are the spatial portions of the “Contaminant Melamine” displays for both simulations, on the same day, shortly after deposition.

Original Scenario: Deposition at a specific cell:


Log10 Melamine Surface Amounts
Log10 Melamine Layers Sum Amounts
May.3,2011 [1/1]
-10
-7.2500
-4.5000
-1.7500
1

Modified (New) Scenario: Deposition on all road cells:


Log10 Melamine Surface Amounts
Log10 Melamine Layers Sum Amounts
May.3,2011 [1/1]
-10
-7.2500
-4.5000
-1.7500
1

Some points to keep in mind while comparing the displays above:

- Melamine is relatively soluble and travels readily when waterflow occurs.
- In the example watershed, the highest and steepest cell elevations are along the west (left) side of the map. This seems to explain the rapid development of a melamine plume at that location, that is, steep slopes favor rapid runoff from impermeable roads onto more permeable off-road surfaces.
- The original example's deposition cell location has high permeability, while road cells do not.

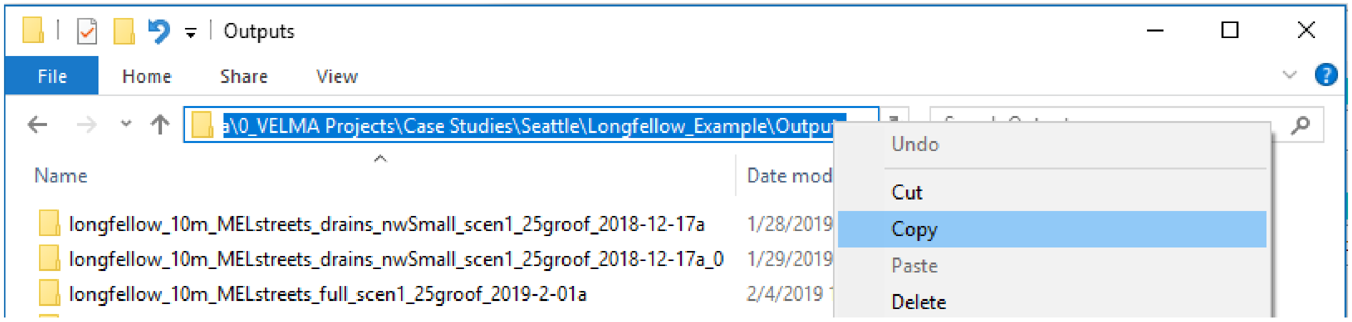
For additional information on how permeability is specified, see:
Tutorial E.1 – Mapping Surface Layer Permeabilities.

Appendix 1 – Miscellaneous Tips and Shortcuts

How To Avoid Typing File Paths in Windows 10

You can avoid typing the name of a directory name as a parameter for VELMA by instead finding it in a File Explorer window and copy-pasting it from there.

Navigate the File Explorer to the desired directory, double-click the location name text box, then right-click and click the Copy item. (Pro Tip: after double-clicking to highlight-select, you can type Ctrl-c instead of using the "Copy" menu item.)


Outputs
File
Home
Share
View
a\0_VELMA Projects\Case Studies\Seattle\Longfellow_Example\Output
Name
Date mod
Undo
Cut
Copy
Paste
Delete

```
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a      1/28/2019
longfellow_10m_MELstreets_drains_nwSmall_scen1_25groof_2018-12-17a_0    1/29/2019
longfellow_10m_MELstreets_full_scen1_25groof_2019-2-01a                 2/4/2019
```

The text of the full path to the File Explorer's current location is now copied to the Windows system clipboard.

You can right-click → Paste (Pro Tip: or type Ctrl-v) that location text into VELMA parameter value fields, or onto the command line of a PowerShell.

Whitespace in Paths in Windows 10 and VELMA

The Windows 10 PowerShell console window separates text typed onto its command line by whitespace. This means that any directory or file path containing whitespace must be enclosed in double-quote characters to preserve the entire path as a single element on the command line.

For example, without quotes, the following path:

```
C:\User\dpines\a\path with some\whitespace\foo.txt
```

Is "seen" by the PowerShell console as three separate elements:

```
“C:\User\dpines\a\path”, “with”, and “some\whitespace\foo.txt”
```

To force PowerShell to "see" the path as a single element, it needs to be enclosed in double-quotes, like this:

```
    "C:\User\dpines\a\path with some\whitespace\foo.txt"
```

However, when paths containing whitespace are specified as parameter values in the VELMA GUI, they **must not** be enclosed in double-quotes.

Windows 10 Path Separators and VELMA

Windows 10 uses the backslash "\" character as a separator between directory and file names.

Example: `C:\this\that\and\the\other.txt`

Other operating systems (e.g. OSX and Linux) use the forward-slash "/" character.

VELMA *tries* to be platform-neutral about path separators.

When you specify a path, file, or path-and-file name as a VELMA configuration parameter in the VELMA GUI's "Run Parameters" and "All Parameters" panels, you *should* be able to use either the backslash "\" or forward-slash "/" character as a separator.
However, the following rules of thumb apply:

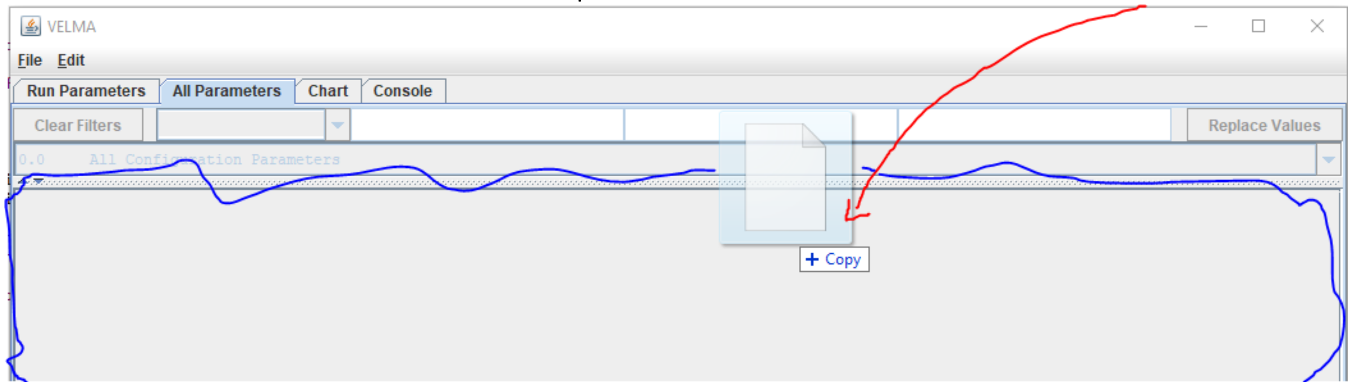
- Within a single path, please be consistent.
  For example, don't specify: `C:\this/that\and/the/other.txt`
- Although it tries to accept both, VELMA *prefers* forward-slash path separators.
  If VELMA is rejecting an otherwise-legitimate, directory or file parameter value, edit its name to use forward-slashes and see if this resolves the issue.
-

Drag-and-Drop Simulation Configuration Scenario Files into VELMA GUI

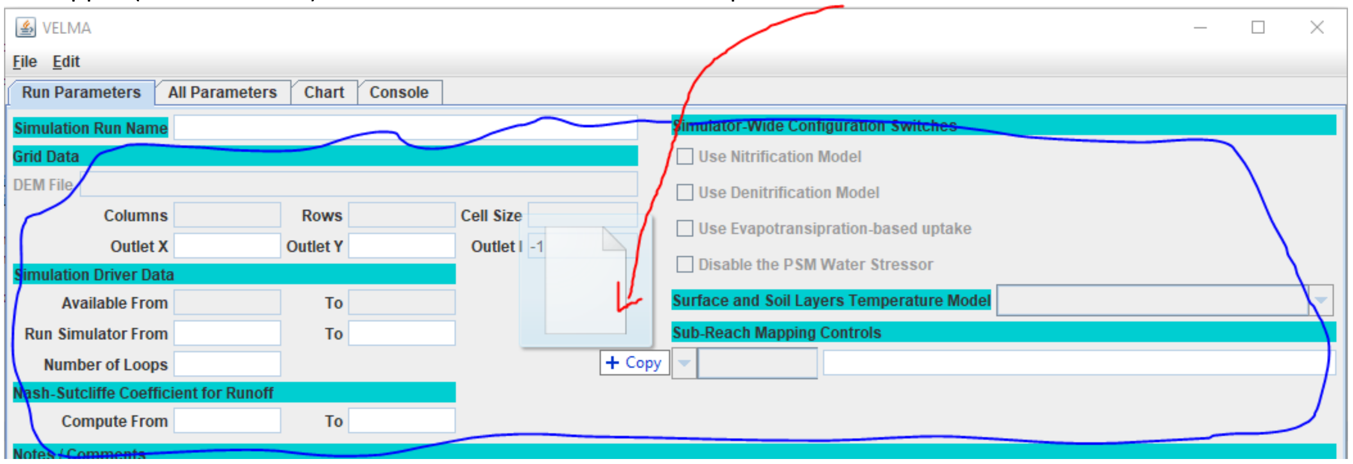
Instead of using the File → "Load Configuration From VELMA XML File" menu item to load a VELMA scenario, you can locate the file in Windows File Explorer, then click-select-drag that file onto the VELMA GUI and unclick ("drop") it

Caveat: There are only 2 areas within the GUI where you may successfully "drop" .xml files:

- The main table area of the "All Parameters" tab panel:



- The upper (i.e. non-Notes) area of the "Run Parameters" tab panel:



In both cases, if the .xml file is a valid VELMA simulation configuration file, the GUI loads its contents.

A.4 | Adding Stormwater Drains for Urban & Mixed-Use Watershed Applications

Overview *(Tutorial A.4 – Adding Stormwater Drains for Urban Contaminant Fate & Transport Applications)*

Stormwater infrastructure – especially storm drains (catchments) and pipes when coupled with impervious surfaces – has a major impact on urban runoff and needs to be explicitly modeled to accurately predict contaminant fate and transport in urban watersheds.

This tutorial describes how to use VELMA's Water Drain Disturbance routine to set up stormwater drains and pipes for simulating transfers water between non-adjacent grid cells, or remove it from the simulation watershed.

The Water Drain Disturbance provides a way to redistribute or remove water within a watershed at specific steps of a VELMA Simulation run.

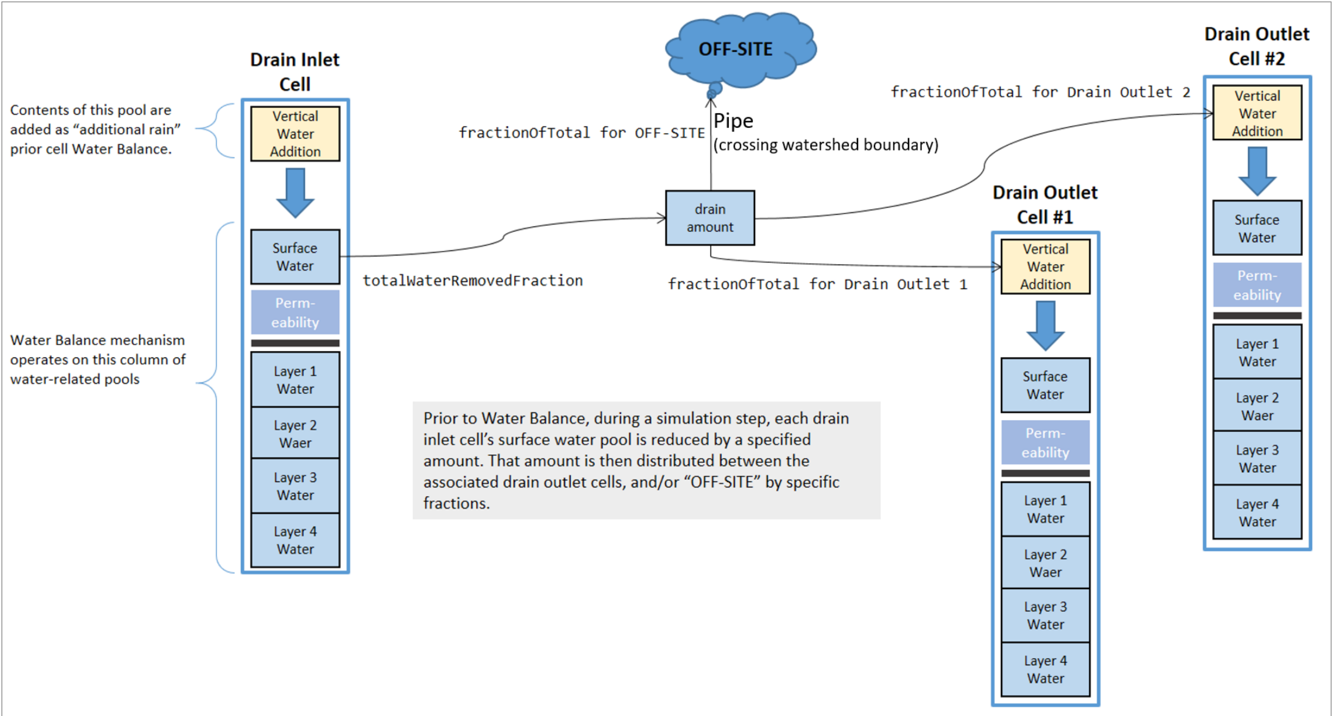
As of September 2021, the Water Drain Disturbance routine has been shown to accurately represent stormwater runoff of surface water and entrained dissolved contaminants via stormwater drains and pipes, via the water drain disturbance feature in VELMA 2.1 (McKane et al. 2021).

General Procedure for Implementing Water Drain Disturbance

The Water Drain Disturbance transfers water between non-adjacent cells or removes it from the simulation watershed. It provides a way to redistribute or remove water within a watershed at specific steps of a VELMA Simulation run. Examples of use would be to explicitly simulate transfer of water through stormwater systems, detection ponds, or culverts.

Users specify one or more drain inlet cell locations for a given Water Drain Disturbance, along with parameters that determine when, and under what conditions, those drain inlet cells are to transfer water from the Surface Water spatial pool.

Users also specify whether the water drained from inlet cells is transferred to drain outlet cells in the same watershed, or is transferred beyond the watershed boundary, which will result in water transferred out to not be represented in the simulations hydrograph and hydrology statistics.


Drain Inlet Cell
Contents of this pool are added as "additional rain" prior cell Water Balance.
Vertical Water Addition
Surface Water
Perm-eability
Layer 1 Water
Layer 2 Waer
Layer 3 Water
Layer 4 Water
Water Balance mechanism operates on this column of water-related pools
OFF-SITE
fractionOfTotal for OFF-SITE
Pipe
(crossing watershed boundary)
drain amount
totalWaterRemovedFraction
fractionOfTotal for Drain Outlet 1
fractionOfTotal for Drain Outlet 2
Prior to Water Balance, during a simulation step, each drain inlet cell's surface water pool is reduced by a specified amount. That amount is then distributed between the associated drain outlet cells, and/or "OFF-SITE" by specific fractions.
Drain Outlet Cell #1
Drain Outlet Cell #2

Conceptual diagram of VELMA 2.1 cell-to-cell transfers of water (and solubilized chemicals) via explicitly modeled stormwater infrastructure features, including curbs, drains, and pipes. VELMA's Water Drain Disturbance routine transfers water and chemicals between non-adjacent cells or removes it from the simulation watershed. It provides a way to redistribute or remove water within a watershed at specific steps of a VELMA Simulation run. Users specify one or more drain inlet cell locations for a given Water Drain Disturbance, along with parameters that determine when, and under what conditions, those drain inlet cells are drained. Users also specify whether the water drained from inlet cells is transferred to drain outlet cells in the same watershed or is removed from the watershed.

Water Drain Disturbances Have Several Significant Usage Restrictions

When configuring a simulation with Water Drain Disturbances, keep the following in mind:

1. Water Drain Disturbances use within a VELMA's Parallel Mode is not recommended, though technically feasible. To be executed correctly, all water transfers must occur within the same sub-reach, or all water leaving a sub-reach should do so. Any water transfers that do occur across VELMA Parallel sub-reaches will be lost to the simulation due to how VELMA Parallel tracks water flowing from independent to dependent sub-reaches. See "HowTo VELMA Parallel Mode.docx" for VELMA Parallel mode details.

2. VELMA simulation configurations that employ Water Drain Disturbances must set their Sub-Reach mapping to SOLO_MODE, and this means that runoff data is not available except for the watershed's primary outlet.

3. Water Drain Disturbances affect not only water amounts, but also transports nutrients and contaminants that are present at the corresponding surface pool (nutrients and contaminants that are within the layers are not impacted by this disturbance routine). Nutrient and contaminant quantities are transferred based on the percentage of the water volume transferred. Example: if 100% of surface water is transferred, 100% of contaminants at that same explicit cell surface location is also transferred. If 50% of surface water is transferred, 50% of contaminants at that same explicit cell surface location is transferred. The Water Drain Disturbance CSV file and optional impervious ASC map can both impact the water volume that is transferred.

Overview: Configuration Steps for a Water Drain Disturbance

Adding one or more Water Drain Disturbances to a VELMA simulation configuration via JVelma involves the following two steps:

1. Create the .CSV drain configuration data file designating the surface water transfer locations and percentages of water or amount of water in millimeters (when transfers are explicit by amount only water available is transferred; never more than any water present at any timestep).
    a. For applications requiring the explicit representation of water movement from catchment (inlet) to outflow (outfall) the use of Geographic Information Systems (GIS) data is leveraged to create CSV files for:
        i. urban applications representing stormwater systems.
        ii. green infrastructure or agricultural applications representing land use change.
        iii. Culverts installed under roadway systems (urban and forestry).

2. Add a new Water Drain Disturbance parameterization group to the simulation configuration, employing the drain configuration data file.

Creating a Drain Configuration Data File

A Water Drain Disturbance must know: 1) the catchment or inlet water drainage cell locations (formatted as 1D i-index form), 2) the outlet or outfall cell locations where drained water emerges (formatted as 1D i-index form) when the water should be transferred to a location within the delineation; otherwise if the water should transfer out of delineation the i-index can be blank or any valid i-index outside the delineation, and 3) when water is available how much water is transferred per timestep. See ***Formatting Notes*** below.

Water amounts transferred are controllable for both the fraction of inlet water to drain, and what fraction of the drained water to transfer to each connected outlet (when multiple outlets are present). The drain configuration data file specifies all this information in the form of a .CSV table. Each row of the drain configuration data file represents one inlet, the fraction of water inlet water to drain, and zero or more outlets, and the fraction of the drained water each receives. See ***Formatting Notes*** below.

For example, the water drain data below, configures a network of 2 inlets. Each inlet drains water to 2 separate outlets, and one of the "outlets" is "OFF-SITE" – meaning the water is removed from the simulation watershed when drainage occurs.

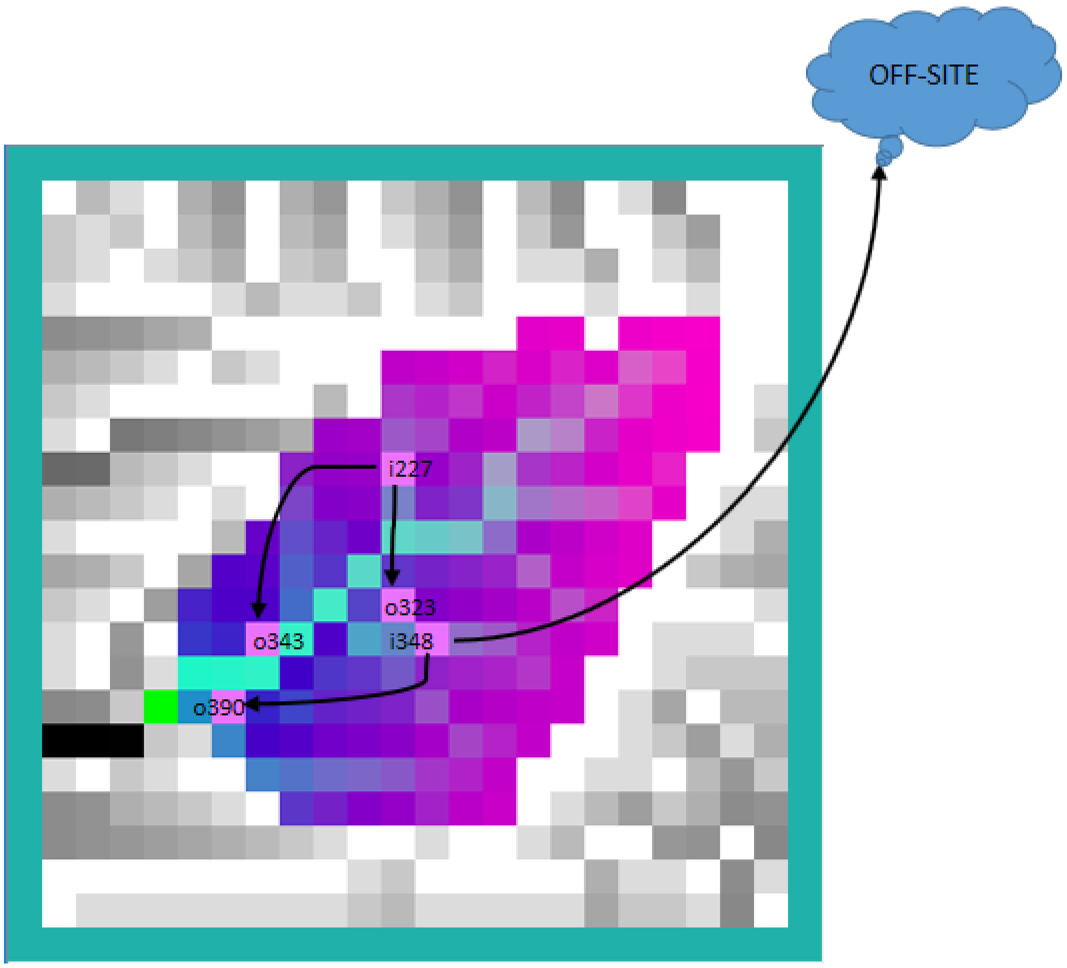
```
Inlet, inFraction, #ofOutlets, Outlet, outFraction, Outlet, outFraction
227,   0.90,       2,          343,    0.50,        323,    0.50
348,   0.75,       2,             ,    0.75,        390,    0.25
```

Formatting Notes:

- The #ofOutlets field value must match number of Outlet,outFraction pairs of fields that follow it.
- The inFraction, and all outFraction values must be in the range [0.0, 1.0]

- The outFraction values for a given inlet must collectively sum to 1.
- Specify the OFF-SITE location as an empty Outlet location field paired with an outFraction value.
- The OFF-SITE location may be specified for multiple inlets, but not multiple times for a single inlet.
- The example data above has header row, but a header row is not required.

The example configuration above results in the network of drains illustrated below:

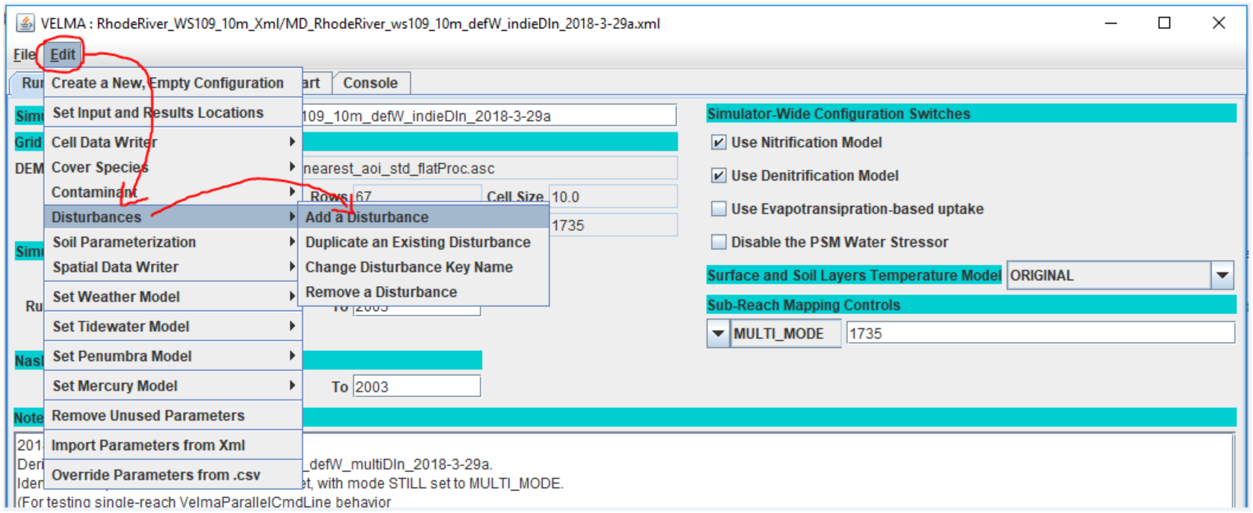


The following table summarizes the fields (left to right) for one row of a drain configuration data .CSV file:

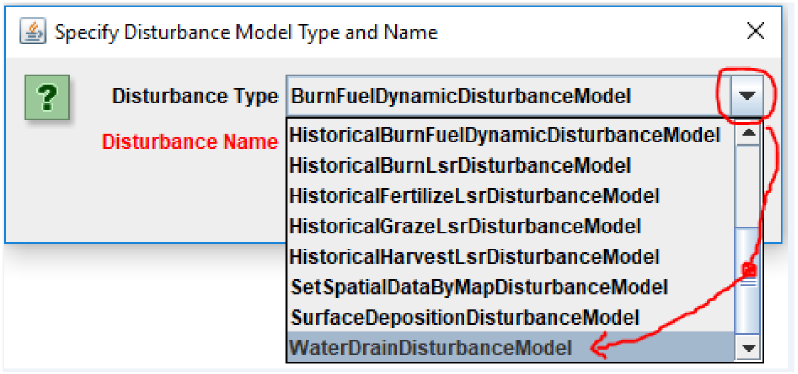
Field	Meaning
0	The linear index of an inlet cell location
1	The fraction of this inlet's water to drain to outlets, range [0.0, 1.0]
2	The number of outlets associated with this inlet
3	The linear index of the first outlet cell location associated with this inlet. Setting the value to blank (but remember the comma!) indicates OFF-SITE drainage.
4	The fraction of the water drained from the inlet transferred to the first outlet cell's Vertical Water Addition pool, range [0.0, 1.0]
< . . . >	*<repeat fields 3 and 4 for each additional drain outlet associated with this inlet>*
n	The linear index of the nth outlet's cell location
n+1	The fraction of the water drained from the inlet transferred to the nth outlet.

Adding a Water Drain Disturbance Parameterization to a VELMA Configuration

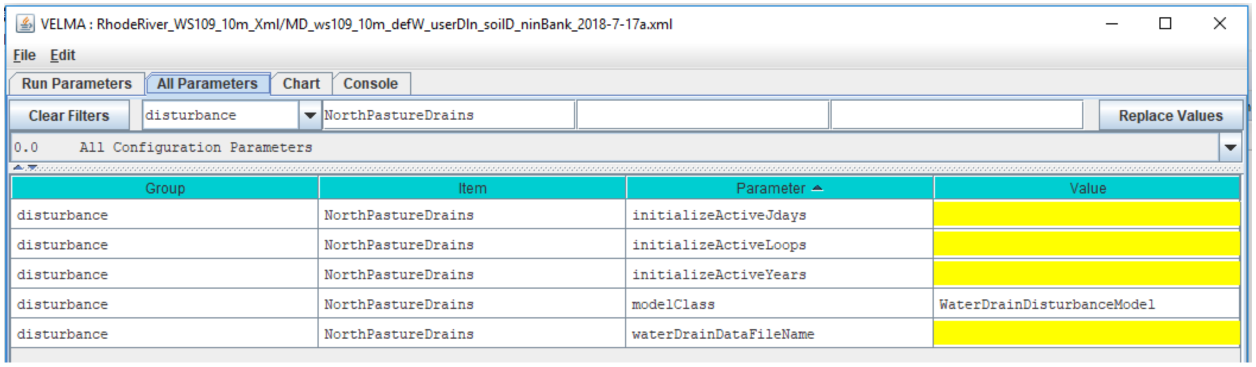
Start with an appropriate VELMA simulation configuration .xml file.
Load the file into JVelma and confirm that the enableReachMapping parameter's value is SOLO_MODE.
Next, click the Edit -> Disturbances -> "Add a Disturbance" menu item, as shown below:


VELMA : RhodeRiver_WS109_10m_Xml/MD_RhodeRiver_ws109_10m_defW_indieDln_2018-3-29a.xml
File
Edit
Create a New, Empty Configuration
Set Input and Results Locations
Cell Data Writer
Cover Species
Contaminant
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Set Penumbra Model
Set Mercury Model
Remove Unused Parameters
Import Parameters from Xml
Override Parameters from .csv
Add a Disturbance
Duplicate an Existing Disturbance
Change Disturbance Key Name
Remove a Disturbance
Console
Simulator-Wide Configuration Switches
✓ Use Nitrification Model
✓ Use Denitrification Model
Use Evapotranspiration-based uptake
Disable the PSM Water Stressor
Surface and Soil Layers Temperature Model ORIGINAL
Sub-Reach Mapping Controls
MULTI_MODE 1735
Rows 67 Cell Size 10.0
To 2003

In the "Specify Disturbance Model Type and Name" pop-up dialog that opens, click the drop-down selector, scroll down, and click-select the "WaterDrainDisturbanceModel" type:


Specify Disturbance Model Type and Name
Disturbance Type
BurnFuelDynamicDisturbanceModel
Disturbance Name
HistoricalBurnFuelDynamicDisturbanceModel
HistoricalBurnLsrDisturbanceModel
HistoricalFertilizeLsrDisturbanceModel
HistoricalGrazeLsrDisturbanceModel
HistoricalHarvestLsrDisturbanceModel
SetSpatialDataByMapDisturbanceModel
SurfaceDepositionDisturbanceModel
WaterDrainDisturbanceModel

After selecting WaterDrainDisturbanceModel as the type, enter a unique name for the disturbance in the "Disturbance Name' field of the dialog. Finally, click the OK button, which adds a new parameterization group to the simulation configuration and changes JVelma's display to the "All Parameters" tab, with the item-level filter set to only display the parameters for the newly added Water Drain Disturbance.


VELMA : RhodeRiver_WS109_10m_Xml/MD_ws109_10m_defW_userDln_soilD_ninBank_2018-7-17a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
disturbance
NorthPastureDrains
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
disturbance	NorthPastureDrains	initializeActiveJdays	
disturbance	NorthPastureDrains	initializeActiveLoops	
disturbance	NorthPastureDrains	initializeActiveYears	
disturbance	NorthPastureDrains	modelClass	WaterDrainDisturbanceModel
disturbance	NorthPastureDrains	waterDrainDataFileName	

Parameters That Must Be Left Alone

The modelClass parameter is set by JVelma when you add the Surface Deposition Disturbance to the simulation configuration. Do Not Change this parameter's value. Ever.

Parameterizing When Drainage Occurs

Specify values for the initializeActiveLoops, initializeActiveYears, and initializeActiveJdays, parameters to determine the loops, years, and Julian days of the simulation when drainage will occur. These parameters accept single values, a comma-separated set of values, a hyphen-separated value range, or a combination of values and ranges.

For example, to schedule drainage during the 3rd loop of a simulation, in years 2000 and 2002, for the first month of each active year:

initializeActiveLoops = 3
initializeActiveYears = 2000, 2002
initializeActiveJdays = 1-31

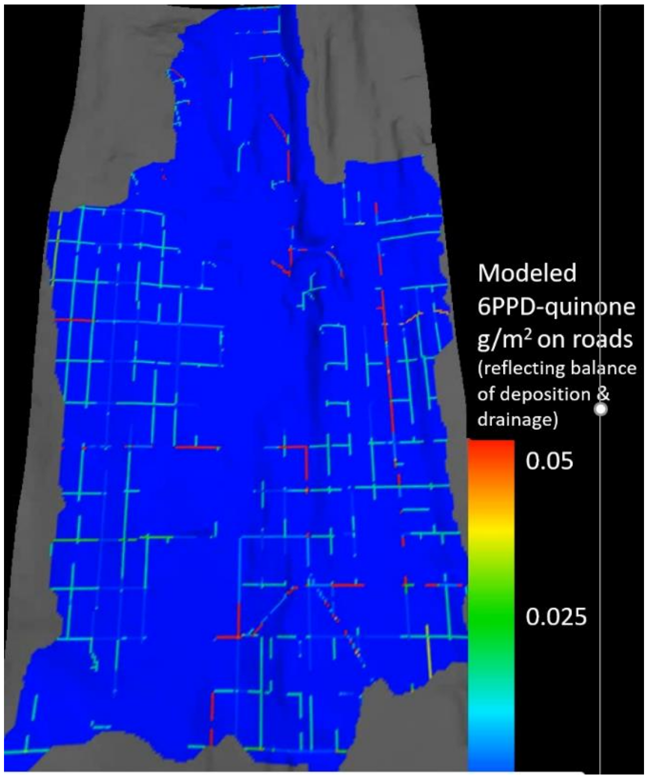
Parameterizing Where and How Much Drainage Occurs

All the details required for drainage inlet, outlet, and quantity are contained within the drain configuration data .CSV file (see the "Creating a Drain Configuration Data File" section above). Set the waterDrainDataFilename parameter to point to that file.
You may specify the file by a fully-qualified path and name, a partial path and name, or only the name of the file. When the waterDrainDataFilename parameter's value isn't fully-qualified, JVelma assumes the file's location is relative to the input location specified for the simulation run by the inputDataLocationRootName and inputDataLocationFileName (startups group) parameters.

Results Output for Water Drainage

The GlobalStateLog.txt file records an INFO statement for each WaterDrainDisturbance occurrence during a simulation run. No additional map files or DailyResults.csv columns are automatically generated. A simple visualization of WaterDrainDisturbance events can be viewed using the "VelmaDrainDisturbanceVisualizer.py" code. This code requires minor, yet direct, changes to the Python code.

The logged WaterDrainDisturbance events can also be processed into ASCII maps using the "curb_n_drainToAsciiMapConvertor.py" code, then provided to the VISTAS 3D visualization tool.

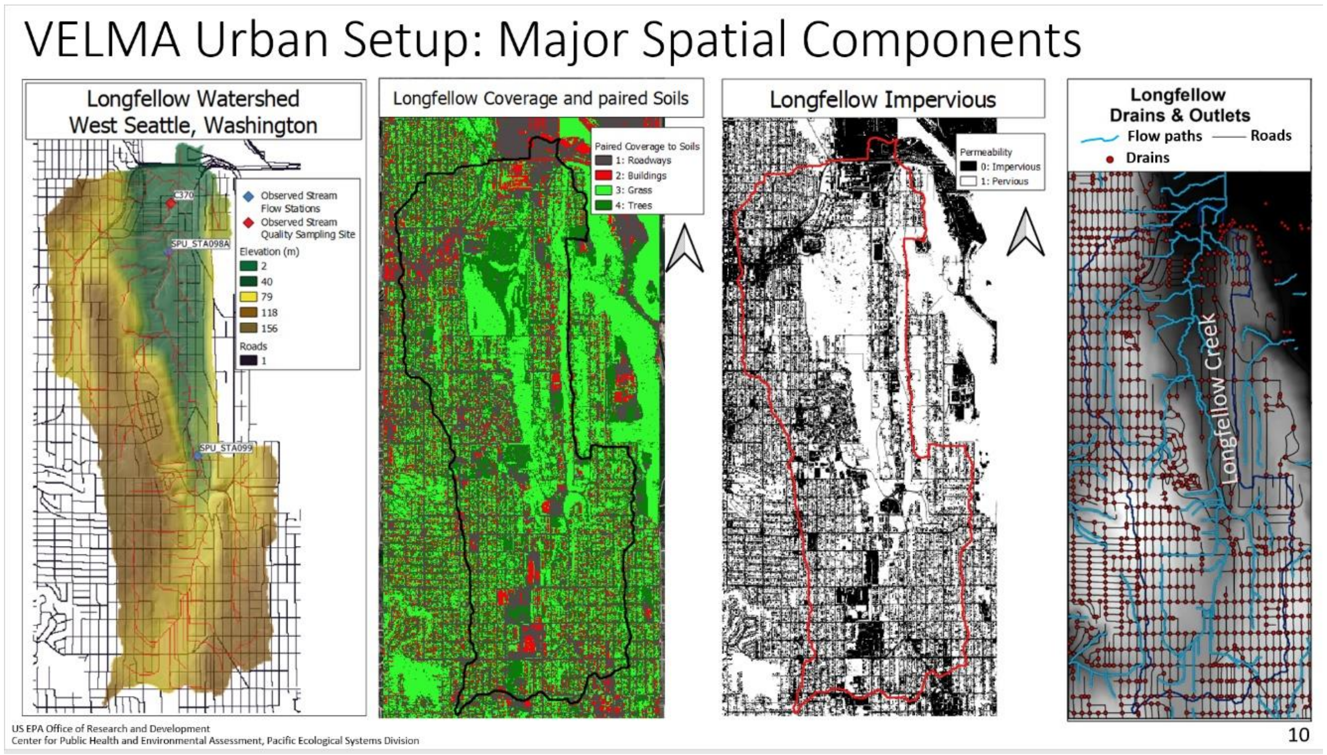


Above is a frame from a VISTAS video of 6PPD-quinone being deposited and routed along road surfaces due to the Water Drain Disturbance routing water along road surfaces to stormwater catchments. Details on this setup and generally how to a Water Drain Disturbance below.

Example of explicit Stormwater System modeling using the Water Drain Disturbance

Part of VELMA additions leading to version 2.1 allowed for the spatially explicit replicate municipal high-resolution spatial data describing stormwater infrastructure (roads, curbs, drains, pipe inlet and outlets) for densely developed urban watershed.

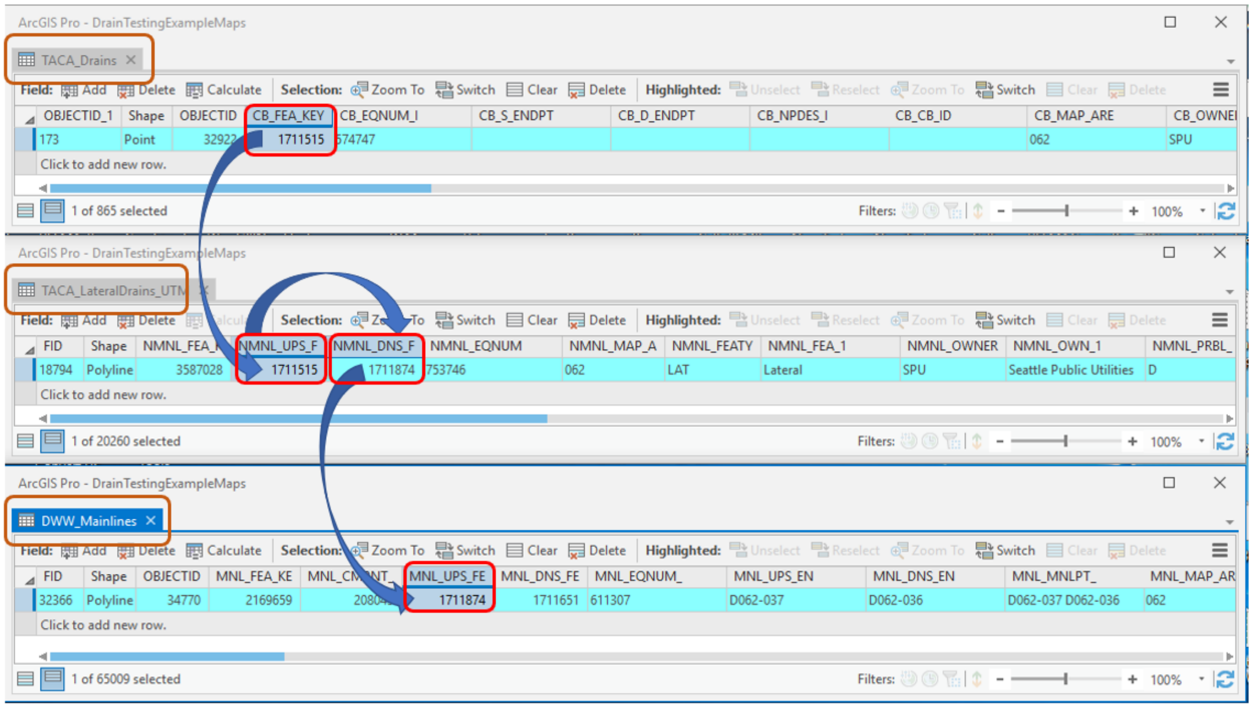
The Longfellow watershed is located within City of Seattle Washington's West Seattle area. Most of this watershed is heavily urbanized with some lesser areas consisting of more natural areas such as the golf course and riparian corridor along the Longfellow Creek. But these more natural areas are overshadowed by this watershed's end to end coverage of impervious roadways and parking lots paired with a modern stormwater system. The VELMA Urban Setup figure below represents the major components that allow VELMA to simulate urban landscapes. This HowTo document focuses on the translation of GIS stormwater data seen in the far-right map panel to the CSV file format described above.


VELMA Urban Setup: Major Spatial Components
Longfellow Watershed West Seattle, Washington
Observed Stream Flow Stations
Observed Stream Quality Sampling Site
C370
SPU_STA098A
SPU_STA099
Elevation (m)
2
40
79
118
156
Roads
1
Longfellow Coverage and paired Soils
Paired Coverage to Soils
1: Roadways
2: Buildings
3: Grass
4: Trees
Longfellow Impervious
Permeability
0: Impervious
1: Pervious
Longfellow Drains & Outlets
Flow paths
Roads
Drains
Longfellow Creek
US EPA Office of Research and Development
Center for Public Health and Environmental Assessment, Pacific Ecological Systems Division
10

VELMA set up for explicitly modeling actual stormwater infrastructure in Longfellow Creek watershed, Seattle, WA (McKane et al. 2021).

The City of Seattle has publicly available GIS data representing their stormwater systems catchments, lateral pipes, mainline pipes, and outfall locations. These four stormwater components are separated regarding the GIS spatial data, yet all linked within the shapefile attributes tables using variable keys. Catchments contain a unique key linking to a lateral pipe or mainline pipe, lateral pipes and mainline pipes all have an uphill key linking to a downhill key linking to the next pipe piece, and all outfall locations have a key linking to final mainline pipe of that section of the network.

The Java based programming tool called "VelmaEndPtFromDrainWalker.java" was developed to "walk" the City of Seattle's stormwater GIS data when provided the above key information as three data tables. The below image represents the tabular walking from key to key among one catchment to one lateral pipe to a mainline pipe. Most lateral pipes are the direct connection between the catchment and mainline pipes. There are cases where more than one lateral pipe was installed, and in those cases multiple lateral pipes are "walked".



ArcGIS Pro - DrainTestingExampleMaps

TACA_Drains

OBJECTID_1	Shape	OBJECTID	CB_FEA_KEY	CB_EQNUM_I	CB_S_ENDPT	CB_D_ENDPT	CB_NPDES_I	CB_CB_ID	CB_MAP_ARE	CB_OWNE
173	Point	32922	1711515	574747					062	SPU

Click to add new row.

1 of 865 selected

ArcGIS Pro - DrainTestingExampleMaps

TACA_LateralDrains_UTM

FID	Shape	NMNL_FEA_K	NMNL_UPS_F	NMNL_DNS_F	NMNL_EQNUM	NMNL_MAP_A	NMNL_FEATY	NMNL_FEA_1	NMNL_OWNER	NMNL_OWN_1	NMNL_PRBL_
18794	Polyline	3587028	1711515	1711874	753746	062	LAT	Lateral	SPU	Seattle Public Utilities	D

Click to add new row.

1 of 20260 selected

ArcGIS Pro - DrainTestingExampleMaps

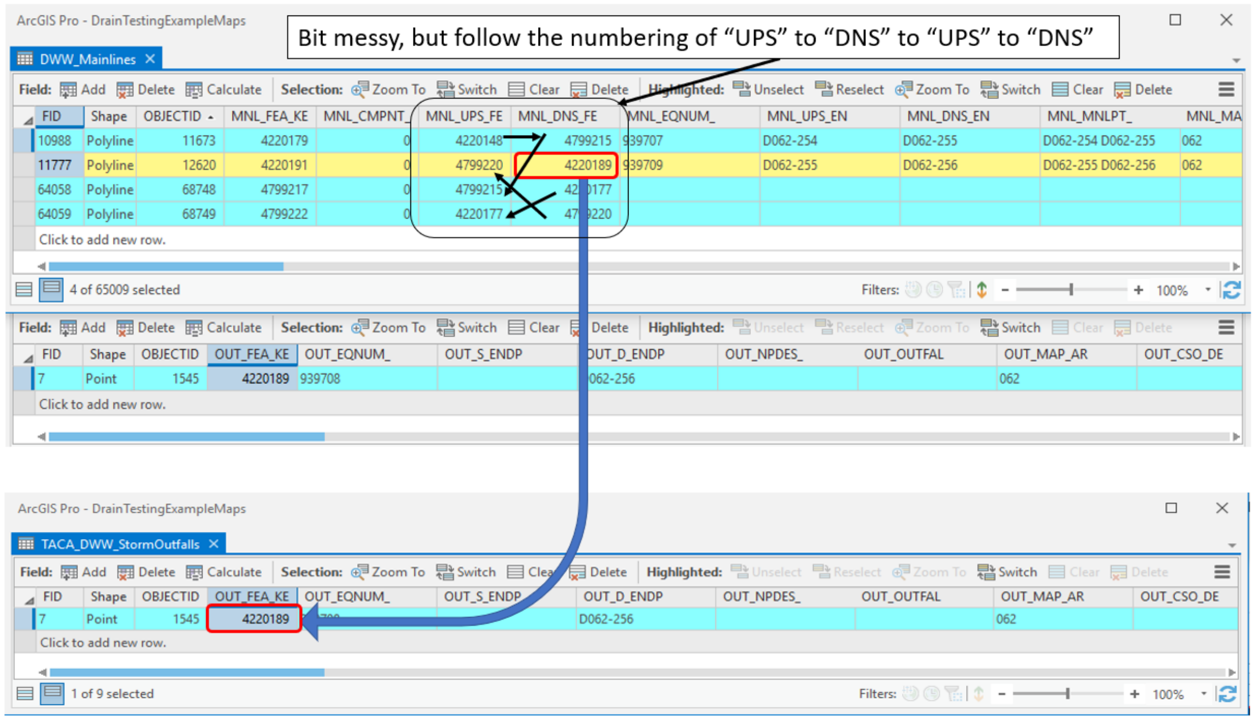
DWW_Mainlines

FID	Shape	OBJECTID	MNL_FEA_KE	MNL_CMPNT_	MNL_UPS_FE	MNL_DNS_FE	MNL_EQNUM_	MNL_UPS_EN	MNL_DNS_EN	MNL_MNLPT_	MNL_MAP_AR
32366	Polyline	34770	2169659	2080[illegible]	1711874	1711651	611307	D062-037	D062-036	D062-037 D062-036	062

Click to add new row.

1 of 65009 selected

The below image represents the same tabular walking pattern among a series of mainline pipes to the final outfall point location.



ArcGIS Pro - DrainTestingExampleMaps

Bit messy, but follow the numbering of "UPS" to "DNS" to "UPS" to "DNS"

DWW_Mainlines

FID	Shape	OBJECTID	MNL_FEA_KE	MNL_CMPNT_	MNL_UPS_FE	MNL_DNS_FE	MNL_EQNUM_	MNL_UPS_EN	MNL_DNS_EN	MNL_MNLPT_	MNL_MA
10988	Polyline	11673	4220179	0	4220148	4799215	939707	D062-254	D062-255	D062-254 D062-255	062
11777	Polyline	12620	4220191	0	4799220	4220189	939709	D062-255	D062-256	D062-255 D062-256	062
64058	Polyline	68748	4799217	0	4799215	4220177					
64059	Polyline	68749	4799222	0	4220177	4799220					

Click to add new row.

4 of 65009 selected

FID	Shape	OBJECTID	OUT_FEA_KE	OUT_EQNUM_	OUT_S_ENDP	OUT_D_ENDP	OUT_NPDES_	OUT_OUTFAL	OUT_MAP_AR	OUT_CSO_DE
7	Point	1545	4220189	939708		D062-256			062	

Click to add new row.

ArcGIS Pro - DrainTestingExampleMaps

TACA_DWW_StormOutfalls

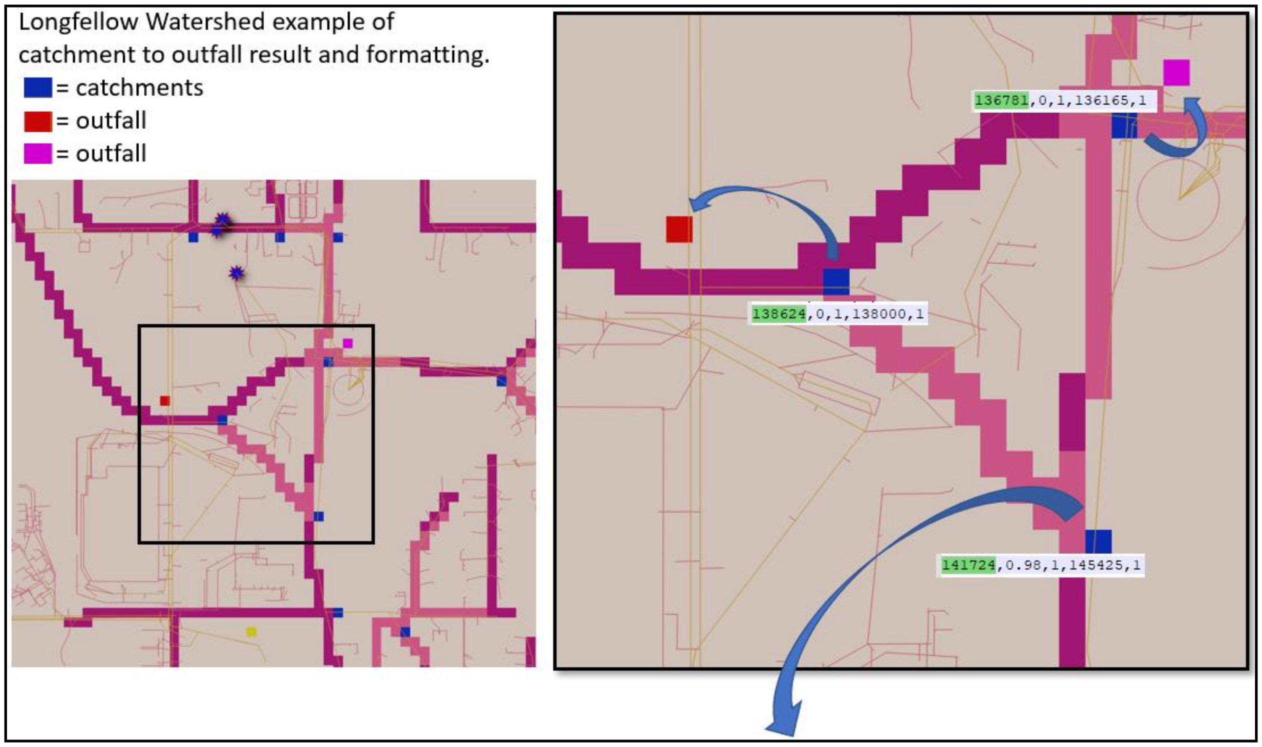
FID	Shape	OBJECTID	OUT_FEA_KE	OUT_EQNUM_	OUT_S_ENDP	OUT_D_ENDP	OUT_NPDES_	OUT_OUTFAL	OUT_MAP_AR	OUT_CSO_DE
7	Point	1545	4220189	[illegible]		D062-256			062	

Click to add new row.

1 of 9 selected

The output from the "VelmaEndPtFromDrainWalker.java" tool matches the format described above under section titled ***Creating a Drain Configuration Data File***. Below is a visual example of highlighting both the location of catchments to outfalls with the CSV file format.

Note: though both the above tabular walking and below visual are both within Longfellow watershed, the above stormwater features in the tabular data are not the locations represented below.



Curbed Streets

Street systems within an urban landscape can include curbs or drainage ditches that route water along roadways to curb stormwater drain inlets or catchment basins. This engineered system entirely removes the water from the original natural flow pathways and instead produces the water either:
unnaturally elsewhere within the watershed delineation (e.g., stream, detention pond, stormwater overflow vaults) shifting the hydrograph peak and timing, or

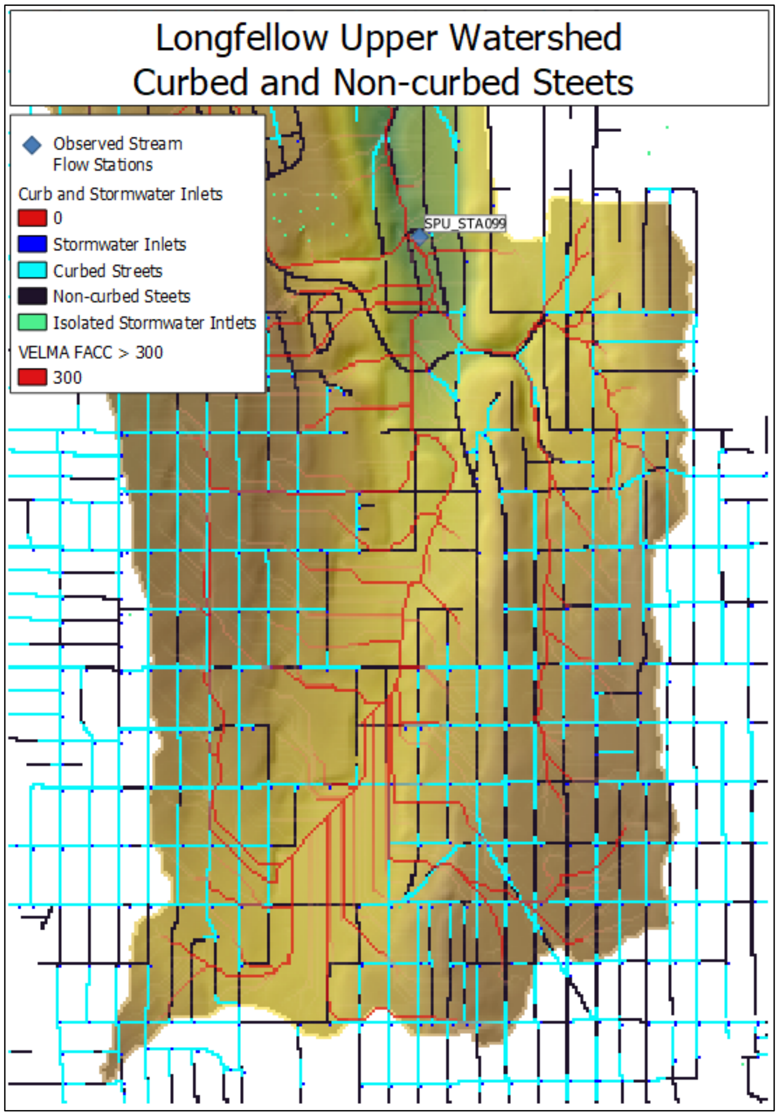
1. produces the water outside of watershed delineation which completely removes the water from the originating watershed's hydrograph.

In either situation, within an ecohydrology model the water must be properly routed within the constraints of the modeling framework. Once the "VelmaEndPtFromDrainWalker.java" tool solves all catchments to outfalls, each catchment is processed in relation to the road network that is provided as a ASCII map matching the VELMA simulations cell resolution and spatial extent. If a catchment cell is on or directly neighbors a road cell the road network is walked for all cells road cells uphill until reaching another catchment cell or the top of the hill. The result is a series of inlet-to-outlet, inlet-to-outlet, etcetera Water Drain Disturbance events that in VELMA route water among a roadway to stormwater catchment inlets. The formatting of these "curb" Water Drain Disturbance events is identical to the stormwater catchment to outfall described in under section titled ***Creating a Drain Configuration Data File***. These curb disturbances and stormwater system disturbances are all output in a single CSV by the "VelmaEndPtFromDrainWalker.java" tool.

The below example highlights the four possible cell types the "VelmaEndPtFromDrainWalker.java" tool would determine.

- Stormwater Inlets (blue): driven by GIS stormwater data and in proximity of road network or impervious parking lots.
- Isolated Stormwater Inlets (green): driven by GIS stormwater data but were not in vicinity of road network or impervious parking lots.

- Curbed Streets (aqua): these were locations that were road cells uphill of a stormwater catchment or contained a direct linkage in series to a catchment.
- Non-curbed (black): these were road cells not determined to have a water routing path directly to a catchment cell. Note: an optional impervious ASCII map could represent water transfer along these cells, but such water would not be routed to a stormwater catchment cell.


Longfellow Upper Watershed
Curbed and Non-curbed Steets
Observed Stream Flow Stations
Curb and Stormwater Inlets
0
Stormwater Inlets
Curbed Streets
Non-curbed Steets
Isolated Stormwater Intlets
VELMA FACC > 300
300
SPU_STA099

References

McKane, R.B., J.J. Halama, V. Phan, A. Brookes, K. Djang, E. Kolodziej, K. Peter. 2021. Model analysis and visualization of 6PPD-quinone fate and transport in Longfellow Creek watershed, Seattle, USA. 2021 International Emerging Contaminants (EMCON) Conference (virtual). U.S. Environmental Protection Agency clearance tracking number ORD-042888.

PowerPoint describing this work:
https://github.com/USEPA/VELMA_Public/blob/master/McKane%20et%20al%20EMCON_2021%20Presentation_9-13-21_v2%20with%20Notes.pptx

B.1 | Creating Flat-Processed DEM Data for the VELMA Simulator

Overview *(Tutorial B.1 – Create Flat-Processed DEM Data for VELMA)*

The VELMA Simulator requires elevation data in the form of a georeferenced Digital Elevation Model (DEM) for any simulation it runs.

This DEM contains elevation data in the form of a grid for the area simulated.

The area specified must contain no "sinks" or "flat spots", so that when VELMA simulates runoff, it must be able to flow "off" the map area in some way. Water is not allowed to pool anywhere within the simulation area.

In order to ensure this "no flat spots rule", DEM data used by the VELMA Simulator first needs to be "flat-processed" to remove any sinks or flat spots.

This document explains how to use JPDEM – an enhanced Java version of the “PDEM” program (Pan et al. 2012) – to (1) flat-process raw DEM data files, and (2) determine the location of a watershed outlet and delineate the boundary the watershed area contributing to that outlet.

How to Acquire Digital Elevation Model Data

Elevation and other common GIS data can be retrieved from many local, state, and federal GIS repository libraries. The USDA’s “Geospatial Data Gateway” provides easy access to multi resolution USGS National elevation data along with many other types of natural resource data. Higher resolution data, like LiDAR can many times be found from alternative public and private stakeholders such as the “Puget Sound Lidar Consortium”. Other popular data, such as STATSGO and newer 10m gridded SSURGO soils GIS layers can also be acquired from the “Geospatial Data Gateway”

References

Geospatial Data Gateway - https://datagateway.nrcs.usda.gov/

Puget Sound Lidar Consortium - http://pugetsoundlidar.ess.washington.edu/

How to Use the JPDEM Tool to Process DEM data for Use with VELMA

To ensure the "no flat spots rule", DEM data used by the VELMA Simulator is first "flat-processed" to remove any sinks or flat spots.

Use JPDEM to prepare DEM data for use by the VELMA Simulator.

Note
You must have a current version of Java (version 1.8 at this writing) installed on your computer to run JPDEM.

I. How to Create a Flat-Processed DEM

Start JPDEM

Open a Windows Command Line and enter the command:

```
C:\> java -Xmx1024m -jar C:\full\path\to\JPDEM.jar
```

The "`-Xmx1024m`" tells the Java Runtime Environment (JRE) to let JPDEM have 1024 megabytes (1 gigabyte) of working memory.

If your machine doesn't have that much memory, substitute a smaller value (e.g. "512" instead of "1024"). Alternately, if the .asc file of DEM data you intend to load is very large (and your machine has the memory) you may substitute a larger value (e.g. "2048" instead of "1024").

If the full path to the JPDEM.jar file contains whitespace characters, place double-quotes around the path.

Example:

```
"C:\full\path\with white space\to\JPDEM.jar"
```

Load the DEM Data into the JPDEM Tool

Load a "raw" DEM file into JPDEM (the file must be in standard ESRI Grid .asc format).

The easiest way to do this is to drag and drop a raw (unprocessed) DEM file into the open JPDEM window.

Alternatively, you can click the "File" menu, then select and click "Load DEM File".
The "Select DEM File to Load" dialog window opens.
Browse to the location of your DEM file, and click the name of the DEM file.
The DEM file's name appears in the "File Name:" text box.

Click "Load File" to load that file's data into JPDEM.
The file is loaded, and an image of the file appears in JPDEM's map display.
If the file is too small (has few rows and columns), click the Image Scale up-arrow button to "zoom in".

Flat Process the DEM Data Using JPDEM

Click the "Tools" menu, then select and click "Flat-Process DEM Data (Standard)". The drop-down menu currently provides a selection of methods to flat-process a DEM grid. There are currently four different JPDEM delineation algorithm implementations to choose from. We believe they all produce the same results (except possibly for cells on the edge of the DEM area -- which is an acceptable difference). However, they vary greatly in how long they take to run. We recommend using the "Experimental" algorithm for delineation because our experience is that it provides accurate results in a comparatively short amount of time.

The following is a quick summary of the alternate flat processing methods, including the strengths and weaknesses of each.

Standard

Feifei Pan's original PDEM algorithm and code, translated to Java.
Raises and lowers cell elevations.
May NOT completely process map -- has a "time-out" to keep it from running indefinitely.
UNSAFE for use with maps that have had border-mask and/or channel dredge applied beforehand.

Alternate

Feifei Pan's original algorithm, implemented to with some simple Java optimizations.
Raises and lowers cell elevations.
May NOT completely process map -- has a "time-out" to keep it from running indefinitely.
Is usually a bit faster than the Standard engine, but in a few cases has proven to run slower.
UNSAFE for use with maps that have had border-mask and/or channel dredge applied beforehand.

Experimental (recommended)

Allen Brookes' complete re-implementation of Feifei Pan’s original algorithm.
Only raises cell elevations.
Will always completely process map (not proven formally).
Is almost always much faster than either the Standard or Alternate engines.
Can be disproportionately slowed down by large, contiguous flat areas adjacent to edges of the map.
Can be safely used with maps that have had border-mask and/or channel dredge applied beforehand.

Enhanced Experimental

Allen Brookes' refinement of the Experimental algorithm.
Primarily raises cell elevations (but may lower them as well).
May NOT completely process map, but has a high likelihood of doing so.
Is almost always faster than the Experimental engine.
Can be disproportionately slowed down by large, contiguous flat areas adjacent to edges of the map.
Can be safely used with maps that have had border-mask and/or channel dredge applied beforehand.

Enhanced + Divide-and-Conquer Experimental

Allen Brookes' additional refinement of the Experimental algorithm
Primarily raises cell elevations (but may lower them as well).
May NOT completely process map, but has a high likelihood of doing so.
Is almost always faster than the Experimental Enhanced engine.
Can be disproportionately slowed down by large, contiguous flat areas adjacent to edges of the map.
Can be safely used with maps that have had border-mask and/or channel dredge applied beforehand.

After selecting (clicking) one of the preceding options, JPDEM will process the DEM, adjusting elevation data to remove flat spots or sinks within the DEM data's area.

If the processing takes more than a second or two, the display will periodically update, showing remaining flat spots in blue.

When the flat-processing algorithm finishes, the display shifts to a grey-scale representation, with darker-colored pixels representing cells with higher flow-accumulation values.

Save the Flat-Processed DEM Data from JPDEM To a File

After JPDEM finishes flat-processes DEM data, click the "File" menu, then click the "Save DEM As ..." submenu, and finally select and click the "Ascii Grid" menu item.
The "Save DEM Grid to File" dialog window opens.

Browse to the location you wish to save the data in, then type a name for the file in the "File Name:" text box.

Finally, click the "Save File" button to save the DEM data to the specified file name and location.

Notes:
Flat-processed DEM data may be loaded into JDPEM just like "raw" DEM data.

Flat-processed DEM date does not need to be flat-processed again after it is loaded into JPDEM.

However, to view the flow-accumulation grey-scale visualization of that flat-processed data, you must determine the flow data information for the flat-processed DEM values.

II. Watershed Delineation

Determining Flow Data for a Flat-Processed DEM

Load a flat-processed DEM .asc file into JPDEM, in the same way you would load a "raw" DEM file.

After loading the DEM data, click the "Tools" menu, then select and click the "Determine DEM Flow Data (Standard)" menu item.

JPDEM will then compute flow data (flow direction and accumulation values) for each cell in the DEM area.

Once the flow data is computed, JPDEM will shift the display to the flow-accumulation grey-scale visualization.

Note
Loading a "raw" DEM file and then immediately trying to determine DEM flow data should not crash JPDEM, but it won't result in any meaningful information either.

Note
The difference in results between the "(Standard)" and "(Alternate)" flow data algorithms should be zero or very small.

You may use either algorithm. The Alternate algorithm is somewhat faster, but often not much faster than the Standard algorithm.

Determining Watershed Delineation

Flat-processed, flow-accumulated DEM data in JDEM may be used to determine the delineation relative to a specified outlet cell (i.e. the set of cells that flow "to" a specified "outlet" cell).

In order to compute the delineation for an outlet cell, you must first specify the coordinates of the outlet cell.

You can do this either by entering the zero-based x (column) and y (row) coordinates of the outlet cell in the "Outlet X=" and "Y =" number boxes, above JPDEM's map display, or by using the mouse.

To select an outlet cell using the mouse, do the following:

Move the mouse pointer over the cell you wish to select.

Press down on either the right or left mouse button.

A context menu should pop-up, containing information about the cell you are pointing at.

Holding the mouse button down, drag the mouse pointer onto the menu item containing cell info (the info should highlight), then release the mouse button.

The menu goes away, and the x and y coordinates for the cell you selected appear in the "Outlet X = " and "Y = " number boxes above the map display.

Note
"Zoom in" on the map display by clicking the up-arrow button of the "Image Scale" to make selecting a specific cell easier. For example, at "Image Scale 3", for example, every cell is displayed by 3x3 pixels. However, on very large maps zooming too far in can crash the display, because it may run out of memory.

Once the coordinates of an outlet cell are specified, click the "Tools" menu, then select and click the "Delineate DEM Data (Experimental)" menu item.
JPDEM will then compute the delineation (i.e. the area of cells) for the currently specified "Outlet X =" and "Y =" values.

The resulting area is displayed in a logarithmic-coloring blue-red scale, superimposed over the grey-scale of the flow-accumulation display.

If the delineation looks incorrect, change the outlet cell coordinates and re-run the delineation.

When the delineation looks correct, record the "Outlet X =" and "Y =" values for future reference and use with JVELMA. (JPDEM does not "record" outlet values anywhere.)

References

Pan, F., Stieglitz, M., & McKane, R. B. (2012). An algorithm for treating flat areas and depressions in digital elevation models using linear interpolation. *Water Resources Research*, *48*(6).

B.2 | JPDEM Cell Highlighter

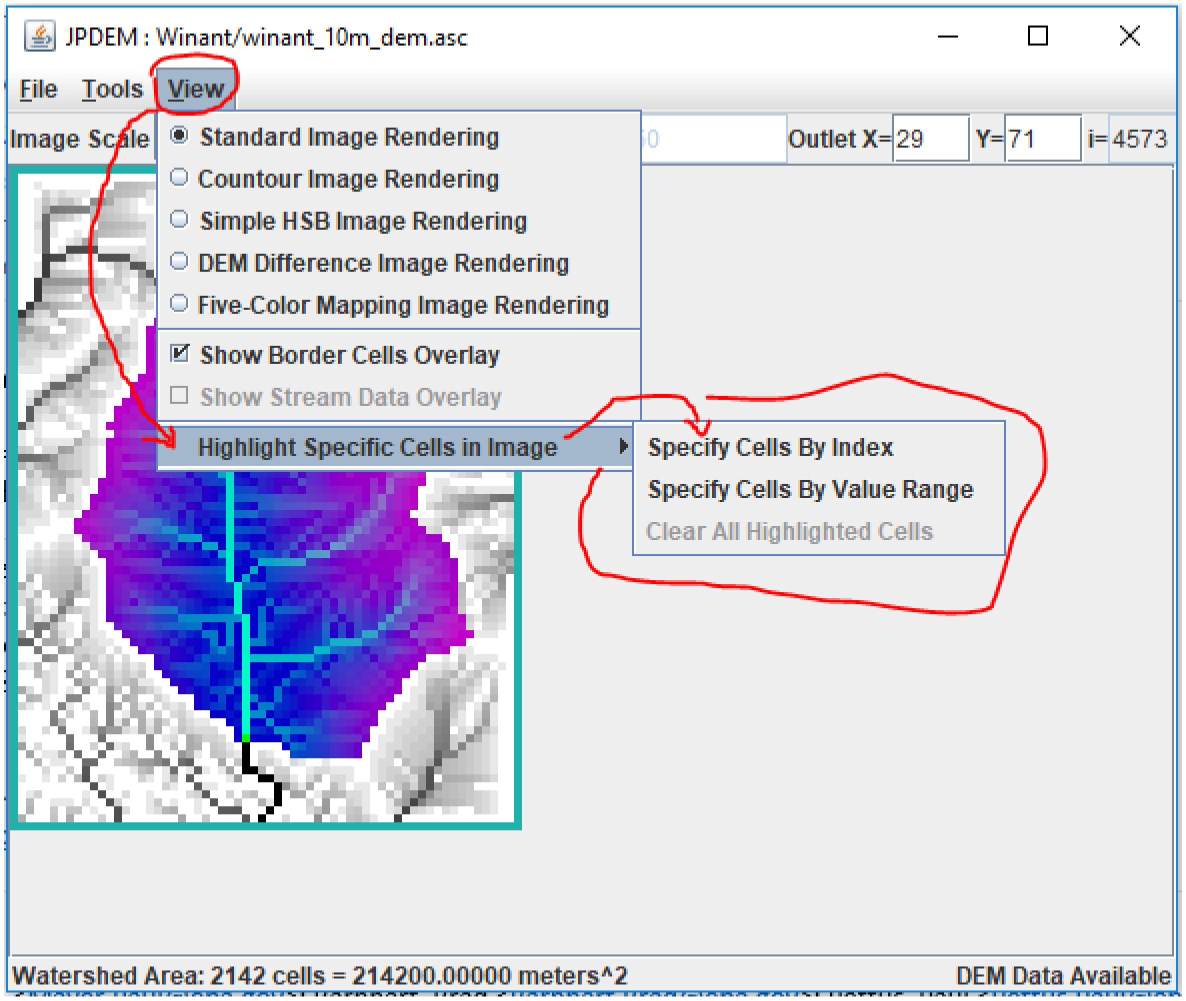
Overview *(Tutorial B.2 - JPDEM Cell Highlighter)*

This tutorial explains how to use JPDEM’s Cell Highlighter functions to select one or more specific cells within the currently-loaded map (DEM or other spatial data) and display them in a distinct color. The highlighter makes those cell’s locations easier to spot and keep track of.

Follow these steps to use JPDEM’s Cell Highlighter functions to highlight the locations of selected cells:

Load a flat-processed DEM into JPDEM (see Section 1.1).

The Cell Highlighter controls are items in the View Menu:


JPDEM : Winant/winant_10m_dem.asc

Outlet X= 29 Y= 71 i= 4573

Watershed Area: 2142 cells = 214200.00000 meters^2

DEM Data Available

File
Tools
View
Image Scale
Standard Image Rendering
Countour Image Rendering
Simple HSB Image Rendering
DEM Difference Image Rendering
Five-Color Mapping Image Rendering
Show Border Cells Overlay
Show Stream Data Overlay
Highlight Specific Cells in Image
Specify Cells By Index
Specify Cells By Value Range
Clear All Highlighted Cells

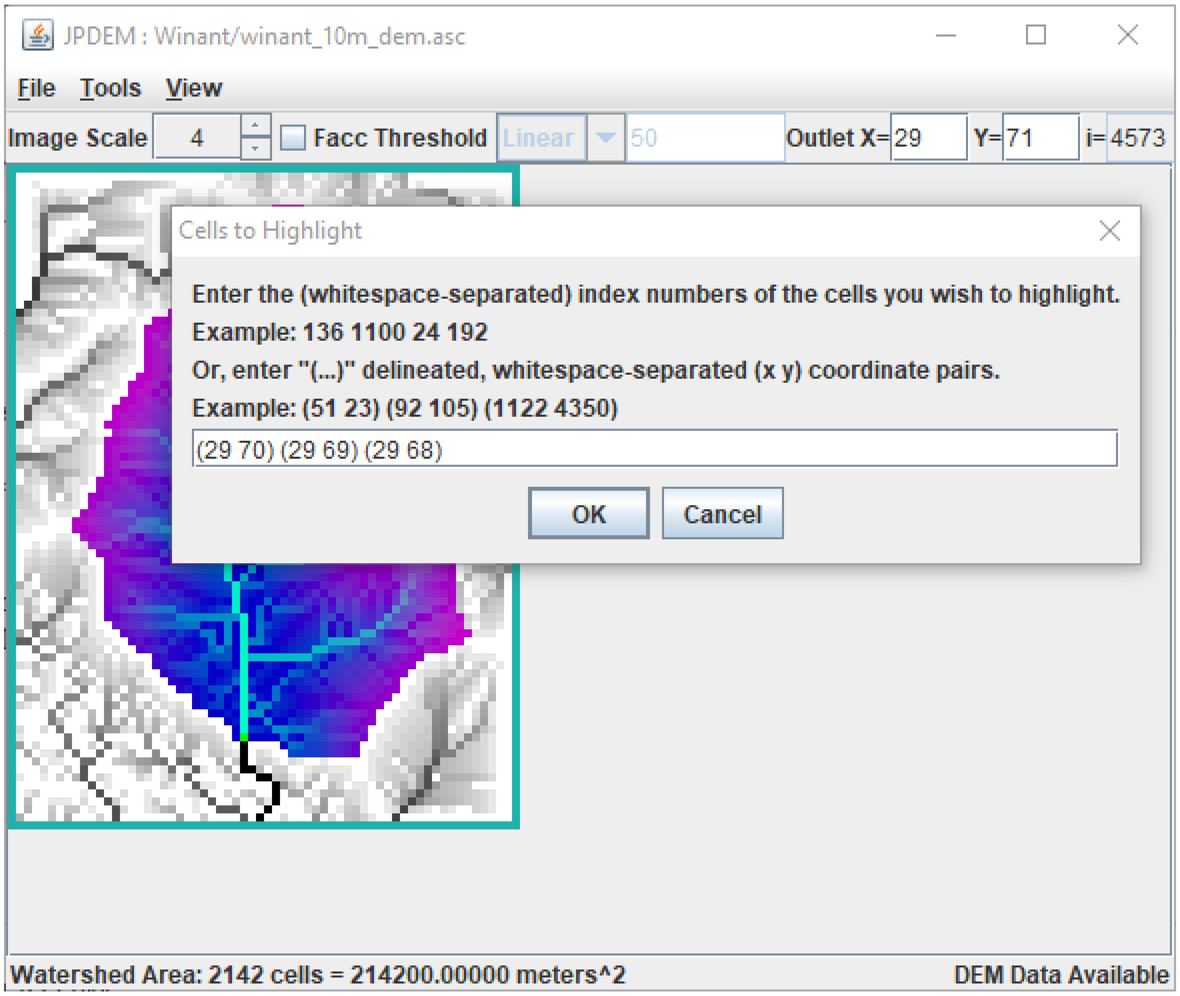
The “Clear All Highlighted Cells” item does exactly what it says, clearing the highlighting from any cells specified for highlighting. It’s only enabled when one or more cells *are* highlighted.

The two “Specify Cells …” options allow you to specify which cells to highlight.
“By Index” means either by listing the linear indices of the cells, or by their (x y) coordinate pairs.

“By Value” means by specifying a range of cell values. All cells within the map whose values are within the specified range are highlighted.

Highlighting Cells by Index

Click the View -> Highlight Specific Cells in Image -> Specify Cells By Index menu item.
The “Cells to Highlight” dialog box opens; note the instructions.
Here is an example that specifies the 3 cells above the outlet location, using the (x y) coordinate pairs notation option. *Important! Notices that there is NO COMMA between the x and y values.*

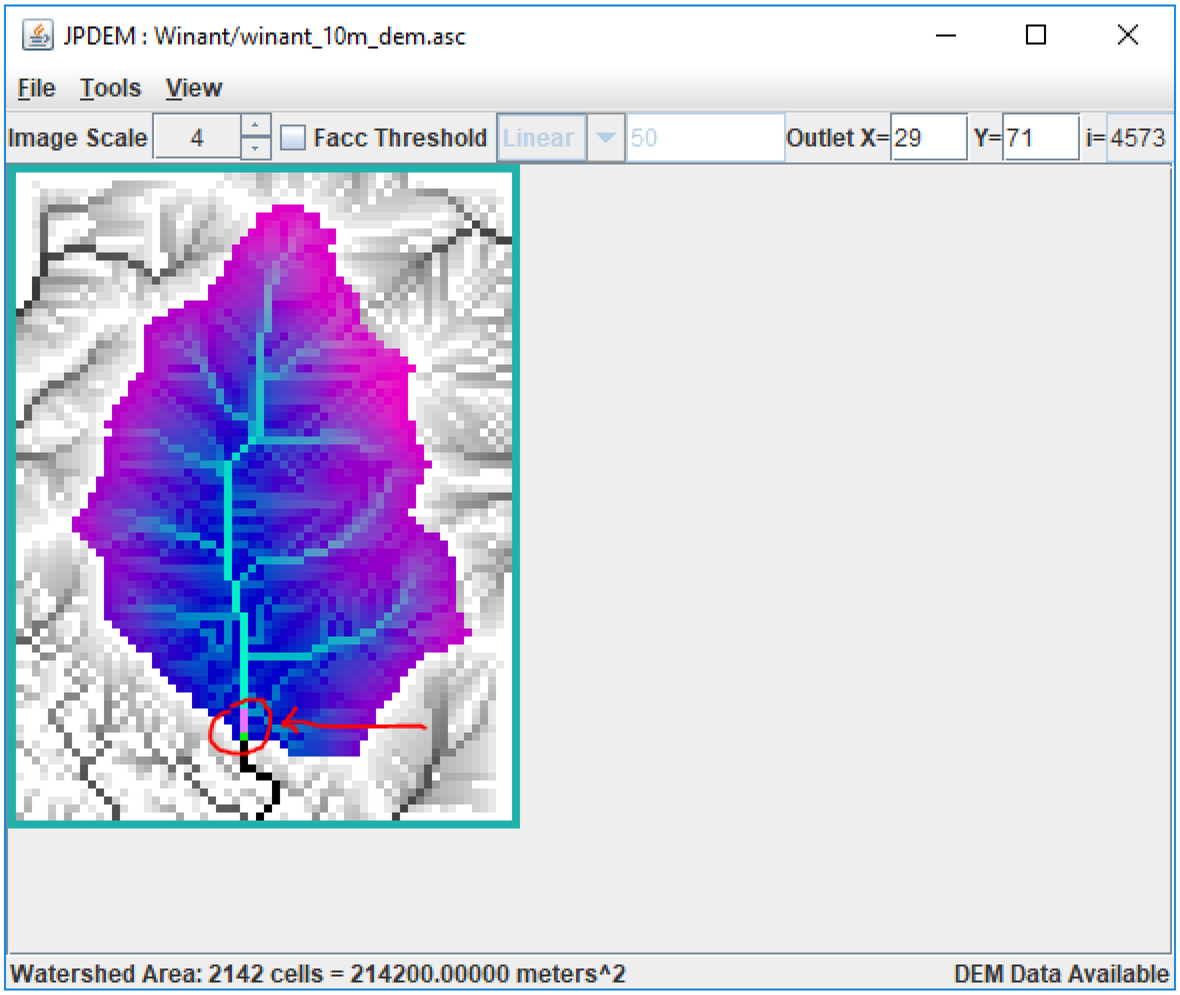


Enter the (whitespace-separated) index numbers of the cells you wish to highlight.
Example: 136 1100 24 192
Or, enter "(...)" delineated, whitespace-separated (x y) coordinate pairs.
Example: (51 23) (92 105) (1122 4350)

(29 70) (29 69) (29 68)

Watershed Area: 2142 cells = 214200.00000 meters^2 DEM Data Available

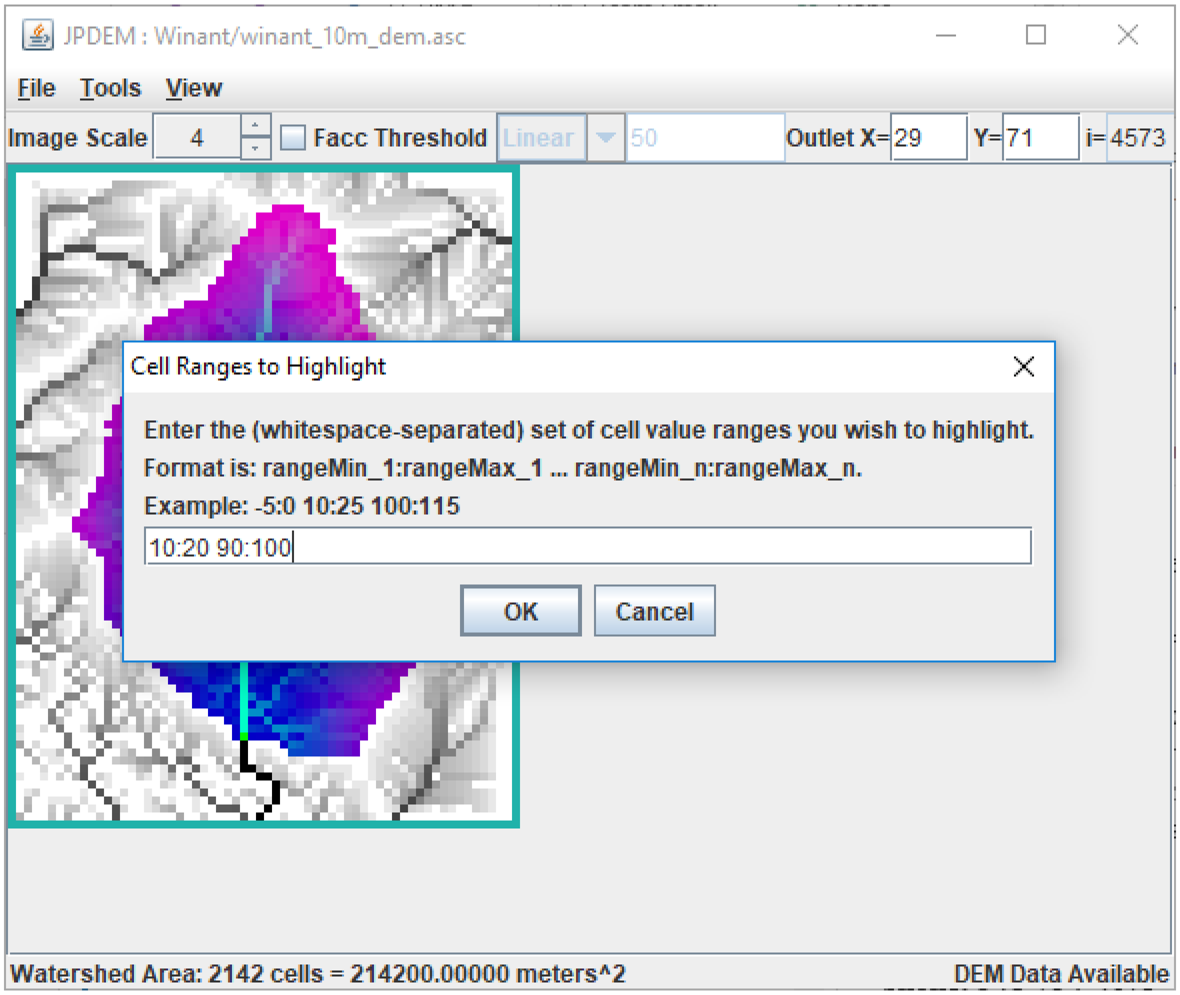
After entering the (x y) coordinate pairs and clicking OK, JPDEM highlights the specified cell locations over the current display/view option:



Watershed Area: 2142 cells = 214200.00000 meters^2 DEM Data Available

Highlighting Cells by Value

Click the View -> Highlight Specific Cells in Image -> Specify Cells By Value Range menu item.
The “Cell Ranges to Highlight” dialog box opens; note the instructions.
Here is an example that specifies all cells with values between 10 and 20, and between 90 and 100:



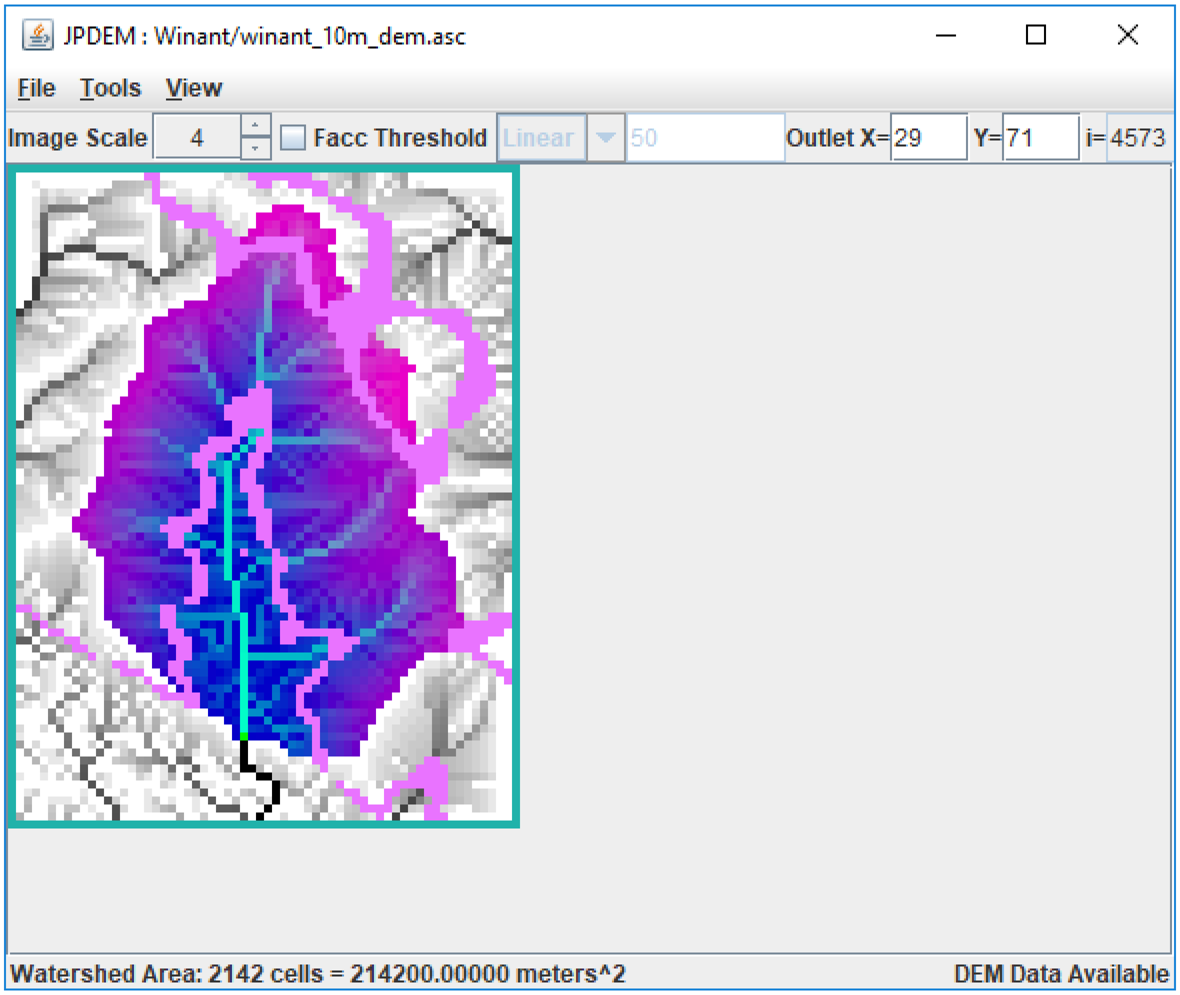
Enter the (whitespace-separated) set of cell value ranges you wish to highlight.
Format is: rangeMin_1:rangeMax_1 ... rangeMin_n:rangeMax_n.
Example: -5:0 10:25 100:115

```
10:20 90:100
```

Watershed Area: 2142 cells = 214200.00000 meters^2
DEM Data Available

Notice that the notation differs from the Cell By index dialog, but that whitespace, NOT COMMAS, is still the delimiter in use.

After entering the low:high coordinate pairs and clicking OK, JPDEM highlights the specified cell locations over the current display/view option:



Watershed Area: 2142 cells = 214200.00000 meters^2

Notes and Limitations

Cell Highlighting (when specified) is visible for all the rendering modes selectable in the View menu. However, depending upon the selected View mode, the highlight color may be "pepto-pink", cyan, or white:

View Rendering Mode	**Highlighter Color**
Standard Image Rendering	Pepto-Pink
Contour Image Rendering	Pepto-Pink
Simple HSB Image Rendering	White
DEM Difference Image Rendering	Cyan
Five-Color Mapping Image Rendering	Pepto-Pink

The Cell Highlighter does not permit specifying both cell index and cell values at the same time. You must choose one or the other, and changing from one to the other clears the previous cell-highlighting.

B.3 | JPDEM Flow Accumulation (Facc) Threshold Display

Overview *(Tutorial B.3_JPDEM Facc Threshold Display)*

This document explains how to use JPDEM’s flow accumulation threshold display to analyze and prepare a DEM to:

- Identify primary flow paths
- Select an outlet cell
- Delineate the watershed boundary for a selected outlet

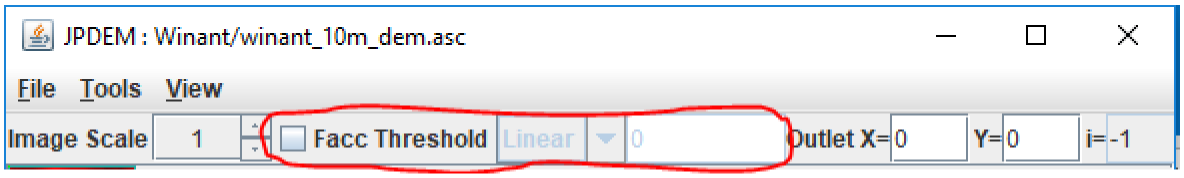
In JPDEM, a DEM cell’s flow accumulation (“Facc”) is the number of upstream cells that flow into that cell. Cells with high Facc values are cells that have high water flow, and cells with high water flow are likely to contain channels.

The JPDEM application’s Facc Threshold display control provides a way to quickly identify and visualize the cells in a (flat-processed) DEM whose Facc values are at or above a specified threshold. You can use the Facc Threshold display to get an idea of which areas and cells of a DEM are likely its main water channels.

JPDEM must have Facc values available for the currently-loaded DEM before the Facc Threshold display is used. JPDEM calculates the Facc values for a DEM during flat-processing, or when the “Determine DEM Flow Data” Tools menu item is run for a previously flat-processed DEM map.
Perform either activity before using the Facc Threshold display controls.

Sidenote: JDPEM and VELMA both compute “flow accumulation” values for the cells of their specified DEM map, and both refer to these as the “Facc” values. However, VELMA calculates Facc values differently than JPDEM. JDPEM and VELMA Facc values for a given cell, while similar, are unlikely to be identical.

The Facc Threshold display controls are located in the center of JPDEM’s upper toolbar:



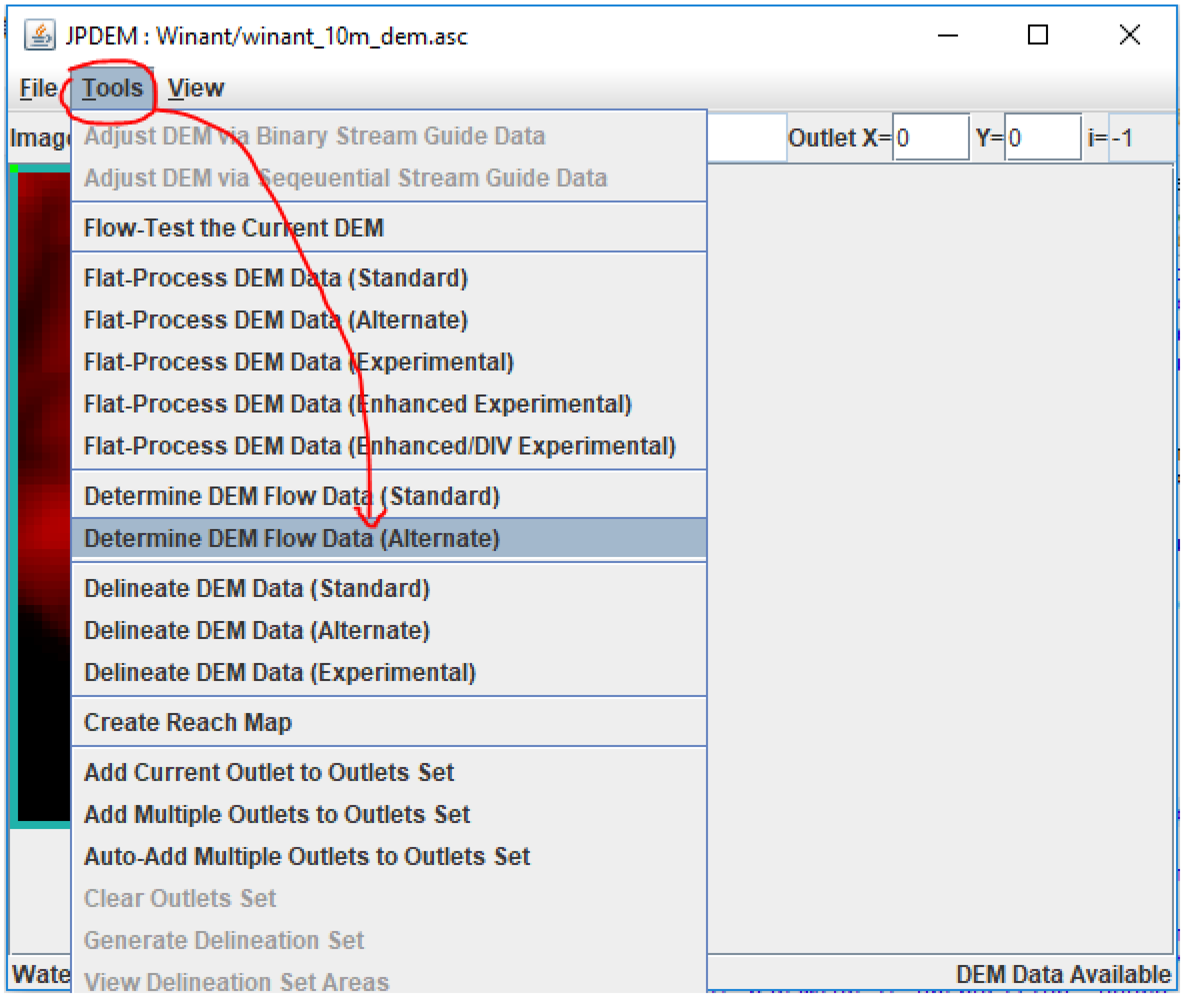
From left to right, the controls are: a checkbox that toggles the display on and off, a drop-down selector for specifying the threshold value’s type, and a number-entry field for specifying the threshold value. All three controls are disabled until a map file is loaded into JPDEM, and the type selector and number-entry field are disabled whenever the checkbox is unchecked.

Using the Facc Threshold Display by Example

In this example, we start with a DEM map that was previously flat-processed, then saved to a new .asc file (see Tutorial B.1 – Creating Flat-Processed DEM Data). We could also start with an unprocessed DEM map – the only difference being that our first step would change from Finding the Flow Data, to Flat-Processing the map.

Step 1

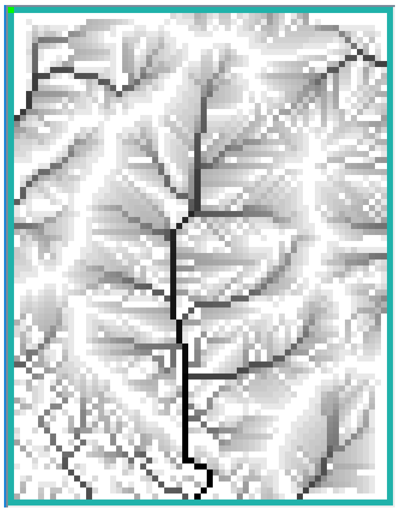
Load the (flat-processed) DEM map from its .asc file, then click the Tools -> “Determine DEM Flow Data (Alternate)” menu item:


JPDEM : Winant/winant_10m_dem.asc
File
Tools
View
Imag
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Seqeuential Stream Guide Data
Flow-Test the Current DEM
Flat-Process DEM Data (Standard)
Flat-Process DEM Data (Alternate)
Flat-Process DEM Data (Experimental)
Flat-Process DEM Data (Enhanced Experimental)
Flat-Process DEM Data (Enhanced/DIV Experimental)
Determine DEM Flow Data (Standard)
Determine DEM Flow Data (Alternate)
Delineate DEM Data (Standard)
Delineate DEM Data (Alternate)
Delineate DEM Data (Experimental)
Create Reach Map
Add Current Outlet to Outlets Set
Add Multiple Outlets to Outlets Set
Auto-Add Multiple Outlets to Outlets Set
Clear Outlets Set
Generate Delineation Set
View Delineation Set Areas
Outlet X= 0
Y= 0
i= -1
Wate
DEM Data Available

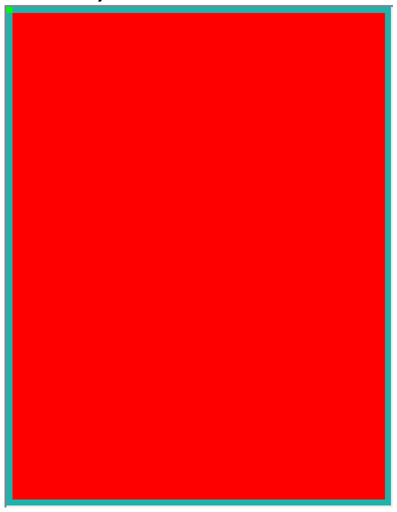
You can use either the “(Standard)” or “(Alternate)” Determine DEM Flow Data item – but the Alternate algorithm is usually quicker.

Step 2
Click the Flow Threshold checkbox control to enable the threshold display. The entire map turns red. Don’t panic! The default Facc threshold value is zero, and the display is showing you all the cells with Facc values > 0.0. Change the number-entry field value to 25.0 and press the Tab key. The display now shows all cells with Facc values > 25.0 in red.

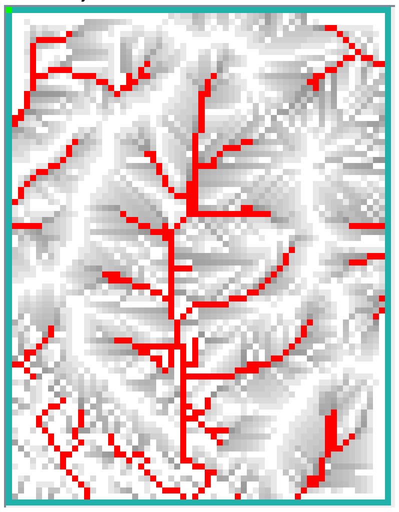
Facc threshold OFF



ON, with threshold = 0



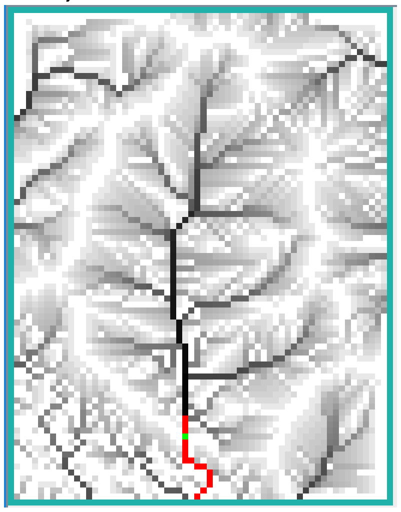
ON, with threshold = 25.0



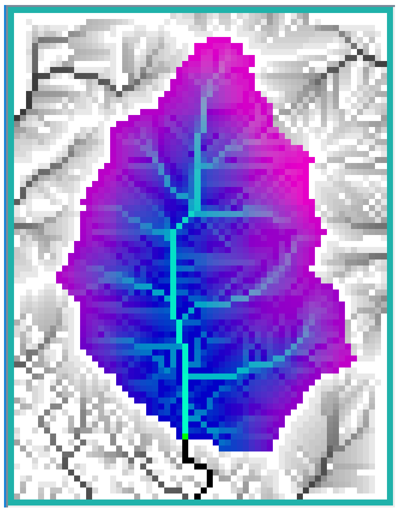
Try changing the threshold (always follow changing the value by pressing the Tab key) to successively larger values. Watch how the number of red cells that meet that Facc threshold dwindles. These are cells with high waterflow.

Step 3
Set the threshold to a large value (2000.0), select one of the highlighted cells as an outlet (green cell, below), and delineate the watershed for that outlet (per Tutorial B.1 – Creating Flat-Processed DEM Data).

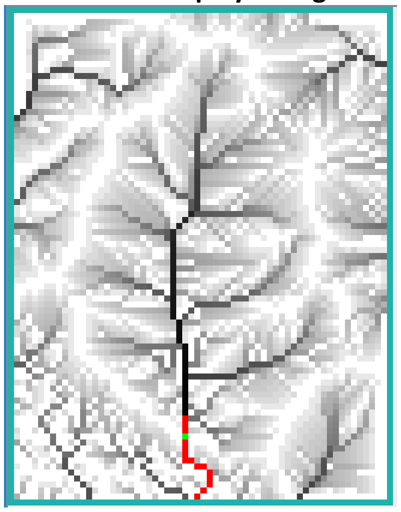
ON, with threshold = 2200.0



After Delineation



Threshold display ON again



Delineation automatically toggles the threshold display OFF, because delineation and threshold data cannot be displayed at the same time. Click the Facc Threshold checkbox to re-enable the display.

Other Options and Some Limitations

You can change the type of the threshold value from linear to log using the drop-down selector. When the Log type is selected, the threshold display highlights cells whose log(Facc) value meets or exceeds the specific value.

As mentioned before, you cannot display threshold and delineation information at the same time, you must toggle between the two. Also, the threshold display is invisible (but not disabled) when you set the View option to any Image Rendering option apart from Standard.

B.4 | JPDEM Flow Test Tool

Overview *(Tutorial B.4_JPDEM Flow Test Tool)*

This document demonstrates how to use JPDEM's flow test tool to perform quality assurance on an existing DEM. The tool will report whether the DEM meets all the criteria needed to support a VELMA simulation.

Use JPDEM's flow test tool to determine:

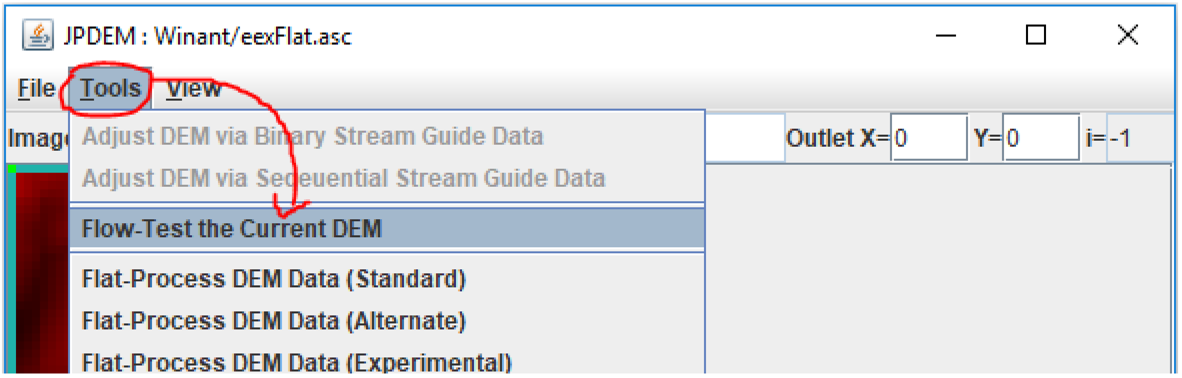
- Whether or not a DEM data map needs to be flat-processed or not.
  Load the DEM .asc file in question into JPDEM and run the flow test tool. The tool will report whether the DEM contains any cells that do not have a flow path to the edge of the map. If the DEM passes the flow test, it does not require flat-processing, and can be used as-is with VELMA.

- Whether or not a flat-processing run of a DEM data map succeeded.
  Some of the flat-processing algorithms (e.g. "Standard" and "Alternate") do not have deterministic halting conditions. To prevent them from possibly running endlessly, they have arbitrary timeout limits. For extremely large maps, the timeout limit may occur before the map is completely flat-processed.
  After flat-processing a map, run the flow test tool if you suspect the flat-processor did not complete its task. If the flow test succeeds, the map was completely flat-processed.

- Basic information about a DEM data map.
  In addition to success or failure, the flow test tool reports some basic information about the map grid (total number of cells, total number of border cells, etc.)

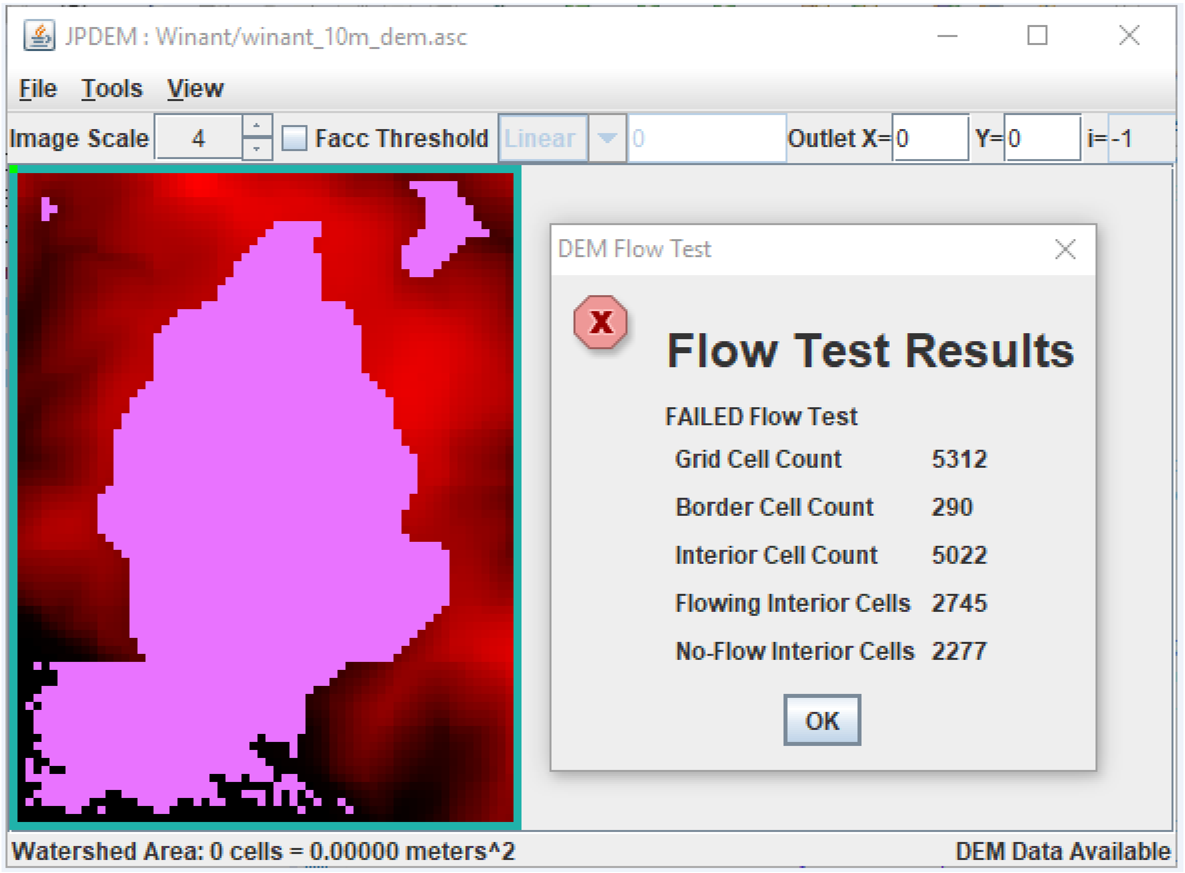
The flow test tool is only enabled when JPDEM has map data loaded. The flow test tool does not expect or require any other preliminary steps; however, it can be run after additional processing (flat-processing, flow determination, etc.) have been run for JPDEM's current data map.

To run a flow test, load a map into JPDEM, and click Tools -> "Flow Test the Current DEM".


JPDEM : Winant/eexFlat.asc
File
Tools
View
Imag
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Sequential Stream Guide Data
Flow-Test the Current DEM
Flat-Process DEM Data (Standard)
Flat-Process DEM Data (Alternate)
Flat-Process DEM Data (Experimental)
Outlet X= 0
Y= 0
i= -1

The flow test tool reports its results in a “DEM Flow Test” pop-up dialog window. (Unfortunately, the contents of the result window cannot be highlight-copied to the system clipboard. If you need to remember them, use the Windows 10 Snipping Tool to screen capture the window.)

Here are the results of running a flow test on a DEM file that has not be flat-processed:



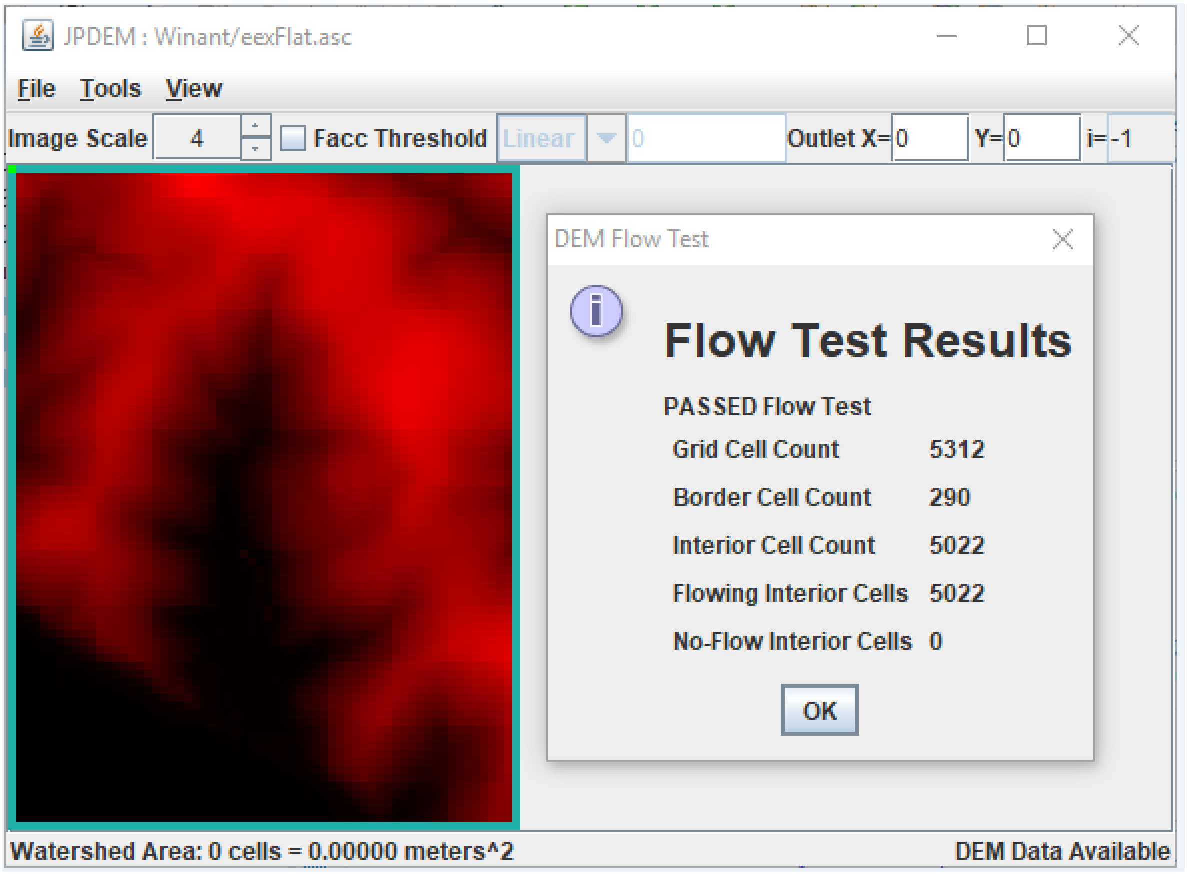
Flow Test Results

FAILED Flow Test

Grid Cell Count	5312
Border Cell Count	290
Interior Cell Count	5022
Flowing Interior Cells	2745
No-Flow Interior Cells	2277

In addition to reporting failure and the number of cells that do not flow to the border of the map, the flow test tool sets Cell Highlighting for each “no-flow” interior cell.
The no-flow cells remain highlighted until you explicitly clear the highlighting.
To clear the highlighted cells, click the View -> “High Specific Cells in Image” -> “Clear All Highlighted Cells” menu item.

In contrast here are the results of running a flow test on a flat-processed version of the same DEM file:



Flow Test Results

PASSED Flow Test

Grid Cell Count	5312
Border Cell Count	290
Interior Cell Count	5022
Flowing Interior Cells	5022
No-Flow Interior Cells	0

Watershed Area: 0 cells = 0.00000 meters^2

Here the flow test reports that all cells passed the test, and no cells are highlighted.

B.5 | JPDEM Saving Data to Files

Overview *(B.5_JPDEM Saving Data to Files)*

This document explains how to use JPDEM's file menu to save different types of data files and formats useful for various spatial analysis and visualization purposes.

JPDEM's File menu contains multiple "Save" items, each of which allows you to save a specific type of data to a file (or set of files):

Save Item	Summary
Velma Simulator Fileset	Saves the current DEM, Flow Direction and Flow Accumulation data in 3 separate .asc files.
DEM Data	Saves the current DEM data -- file type options available.
Flow Direction Data	Saves the current Flow Directions ("fdir") data – file type options available.
Flow Accumulation Data	Saves the Flow Accumulation ("facc") data – file type options available.
Highlighted Cells	Saves the current set of highlighted cells into an .asc file.
Current Image	Saves JPDEM's current display as an image -- file type options available. Unique to this item: save the current image as a .gif, .jpg or .png file.
JPDEM State	Save's JPDEM's entire data state into a .csv file. Provided for JPDEM developers, and not generally useful to others.

Many of the "Save" items also have sub-items that allow you to specify the format of the data file that is written:

File Type Option	Description
Data Array	Writes data values to a .txt file, one value per line, tab-character suffixed. (This is a legacy format, now not commonly used.)
ASCII Grid	Writes data values to an .asc file with dimensions matching the currently-loaded DEM data grid.
Delineated ASCII Grid	As above, but data values for cells outside the current delineation are written as the ASCII Grid "nodata_value" (i.e. "-9999").
Delineated Threshold ASCII Grid	As above, but data values for cells lying below the specified "Facc Threshold" value are written as the ASCII Grid "nodata_value" (i.e. "-9999").
Image (.gif, .jpg, .png)	Only available for the "Save Current Image" menu item.

Additional Details

Save VELMA Simulator Fileset

This item writes three separate .asc files for DEM, fdir, and facc data. The files all share the same name, but have the prefixes "DEM_", "FDIR_" and "FACC_" prepended when they are written.

Note that although the item name declares this as a "VELMA" fileset, currently the facc values written are JPDEM values, not VELMA values (see the comments in the section "Save Flow Accumulation Data" above for more on why this distinction is important).

Save DEM Data

Selecting this item allows you to write JPDEM's current DEM data to a file. The "current DEM data" means whatever JPDEM has currently loaded, in its current state. For example, if you click File -> "Load Map File Data" -> "Load as DEM File", and then immediately click File -> "Save DEM Data As" -> "ASCII Grid", the fil you save will be a duplicate of the file you loaded (apart from any filename differences you introduce). However, if you load a DEM file, flat-process it, and then "Save DEM Data As", the files are likely to contain different data.

Save Flow Direction Data

When JPDEM flat-processes a map, it generates a flow-direction ("fdir") value for each cell in that map. Flow direction data is also generated when the "Tools" -> "Determine Flow Data" item is run.

The flow direction value of a cell is a number between 1 and 8, uniquely identifying one of the cell's eight adjacent cells. A flow direction value <= 0 indicates that flow direction data has not been computed yet.

The directional values for cell "C" are show below:

	6	7	8	
	5	C	1	
	4	3	2	

So, for example, if C's fdir = 2, C's flow direction is to the cell below and to the right of it.

Flow direction indicates only the primary direction that water will flow from a cell to its adjacent cells. Water may flow to adjacent cells in addition to the cell specified by the flow direction value.

Flow direction data may be saved as either an .asc grid file, or a delineated .asc grid file.
The delineated .asc grid file option reports the "nodata_value" (i.e. -9999) for any cell that is not part of the current delineation. If no delineation has been performed yet, the entire .asc file will be nodata_value values.

Save Flow Accumulation Data

When JPDEM flat-processes a map, it generates a flow-accumulation (“facc”) value for each cell in that map. Flow direction data is also generated when the “Tools” -> “Determine Flow Data” item is run.

In JPDEM, the flow accumulation value of a cell is the number of “uphill” cells that flow into that cell. The term “flow into” means that a chain of fdir values leads from the uphill cell to the cell in question, i.e. primary flow directly is the only consideration when calculating a cell’s flow accumulation count.

Note that JPDEM and VELMA both contain flow accumulation data, but VELMA facc values differ from JPDEM’s. While both JPDEM and VELMA’s facc values contain the notion of “cells whose primary flow direction leads to me”, VELMA’s facc value additionally include a scaling component based upon the grid’s cell size, and whether flow direction is across an edge or corner cell.

JPEM fdir values give a fair idea of the high-vs-low flow areas within a flat-processed DEM and watershed, but they *cannot* be used as VELMA facc data. VELMA computes its own facc values from the (JPDEM-produced) flat-processed DEM it is provided with at initialization.

Flow accumulation data may be saved as an .asc grid file, or a delineated .asc grid file.
The delineated .asc grid file option reports the “nodata_value” (i.e. -9999) for any cell outside the current delineation. If no delineation has been performed yet, the entire .asc file will be nodata_value values.

Flow accumulation data may also be saved as a delineated threshold .asc grid file: in this case, any cell that is not part of the current delineation, OR is below the specified Facc Threshold value is set to the nodata_value. Again, if no delineation has been performed, or no Facc Threshold has been set, the entire .asc file will be nodata_value values.

Save Highlighted Cells

When selected, this item writes an .asc file in which each cell has one of three possible values:

- Negative One (“-1”) indicates a cell that is not highlighted, and not part of the current delineation.
- Zero (“0”) indicates a cell that is not highlighted, but that is part of the current delineation.
- One (“1”) indicates a cell that is highlighted, regardless of whether it is part of the current delineation or not.

If no delineation has been performed, no cells will have value zero.
If no cells have been highlighted, no cells will have value one.

Note: even when no cells are highlighted, if a delineation *has* been performed, saving a highlighted cells .asc file will provide a binary-coded (-1, 0) map of the delineated area. This is a legitimate, albeit sneaky, use of the functionality.

Save Current Image

The “current image” is whatever JPDEM is displaying when this item is selected/clicked. JPDEM will write the map image (*not* a screen capture of the entire JPDEM app window) to the file type specified (.gif, .jpg, or .png).

Note: Use the Windows 10 Snipping Tool if you want a screen capture of the entire JPDEM window.

Save JPDEM State

Selecting this item writes JPDEM’s current spatial data state to a comma-separated values (.csv) file. This function is provided for JPDEM developers. General users may find the contents difficult to interpret.

B.6 | JPDEM User-Specified Map Borders

Overview (*Tutorial B.6 – JPDEM User-Specified Map Borders)*

JPDEM’s experimental (“EXP”) and enhanced experimental (“EEX”) flat-processing algorithms experience significant runtime slowdown when the DEM map they are applied to contains large waterbodies or other uniform-elevation areas of cells along one or more edges of the map.

This tutorial explains how to create a border mask that specifies these areas as being part of the map border (i.e. ignored during flat-processing and off-limits for delineation thereafter) can considerably speed up the flat-processing process.

Any DEM grid .asc file loaded into JPDEM must have its computational domain set prior to flat-processing. This is normally done automatically, when you use JPDEM’s “Load Map File Data” --> “Load As DEM File menu item”.

Setting the computational domain involves the following:

- The loaded DEM grid is prohibited from containing any NODATA_value(s).
- A default, single-cell-width border is imposed on the DEM grid.

However, you can instead load a DEM (or any) grid .asc file using JPDEM’s “Load Map File Data” --> “Load As Raw Map File” menu item, and in this case the grid .asc file is loaded as-is, without performing the steps to set its computational domain. A DEM grid .asc file loaded via the “Raw” menu item can subsequently be flat-processed if its computation domain is set, but to do this, the user must load a valid Border Mask Map .asc file.

The BorderMask Map .asc file must have the same column and row dimensions as the loaded DEM grid .asc map, and its cell data must consist solely of integer “1” and “0” values. Border cells must have the value “0” and interior cells must have the value “1”.

Running the “Load Map File Data” --> “Load Border Mask” menu item will load border mask data from the specified file, then set the computational domain of the currently-loaded DEM grid .asc file based on the bitmask of border/interior cells from the border mask map file.

Why Are User-Specified Borders Useful?

JPDEM’s experimental (“EXP”) and enhanced experimental (“EEX”) flat-processing algorithms experience significant runtime slowdown when the DEM map they are applied to contains large, uniform-elevation areas of cells along one or more edges of the map.

Creating a border mask that specifies these areas as being part of the map border (i.e. ignored during flat-processing and off-limits for delineation thereafter) can considerably speed up the flat-processing process.

Important Limitations

- You can only load a border mask map file for a DEM map that was raw-loaded.
- Your border mask map file must (at a minimum) specify the top and bottom rows, and the left-most and right-most columns of the mask as border cells.
- The border mask must encode border cell values as "0", all other cells as "1".
- You cannot load a second border mask map file for a given DEM file.
  Once a DEM map's computational domain is set, it cannot be set again.
  To use a new border mask, re-load the original DEM file and then load the new border mask.

An Overview / Example of Using a Border Mask During Flat-Processing

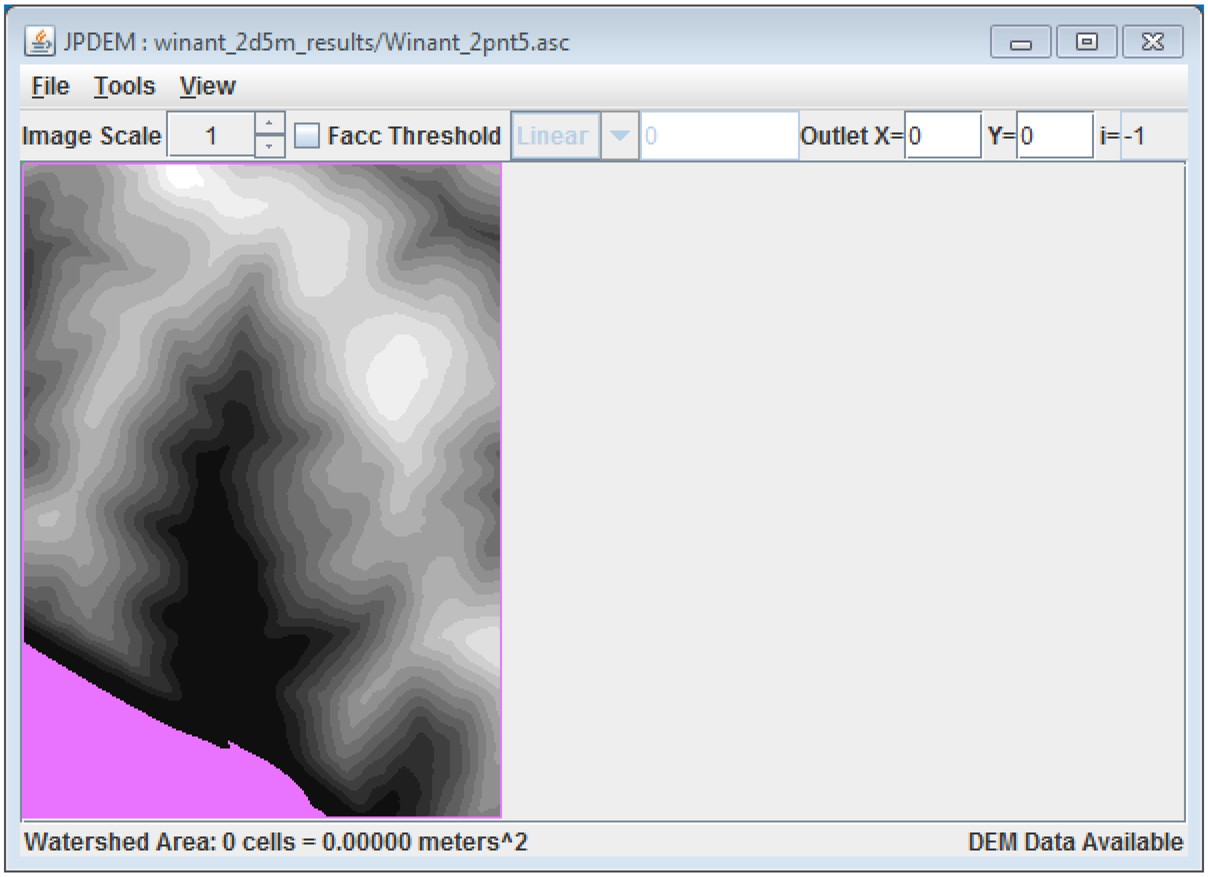
For the following overview, the example data is:

- A DEM file ("Winant_2pnt5.asc")
- A prepared border mask ("Winant_2pnt5_dmMask.asc")
- A stream dredge channel guide ("Winant_2pnt5m__watershedDredgeMap_Corrected.asc")

Step 1:
Load the DEM file as Raw Data via the "Load Map File Data" --> "Load As Raw Map File" menu item.

Step 2:
Load the Border Mask via the "Load Map File Data" --> "Load Border Mask" menu item.

At this point, we can see the user-specified border by highlighting cells that are < 0 meters in elevation. (For this particular map, the lowest elevation prior to setting the computation domain was zero meters. After loading the border mask and setting the computation domain, all border cells will have been "pushed down" below 0 meters in elevation.) In the image below, border cells are highlighted "pepto-pink", while interior cells are shown with contour-rendering greyscale to make the contrast clear:



Notice that the top and bottom rows, and the left-most and right-most columns are part of the border. This was specified in the border mask file (“Winant_2pnt5_dmMask.asc”) and is required (per the prior Limitations list).

Step 3: (Optional)
Load a stream burn-in dredge channel guide map via the “Load Stream File Data” menu item

This is not a necessary step, however, if you plan to perform a stream channel burn-in prior to flat-processing, you must load the data and run the channel-cutter on paired (DEM + border mask) + stream channel guide at this point (i.e. after raw-loading the DEM and loading the border mask, and before running the flat-processor).

Step 4:
Run the EXP or EEX flat-processing algorithm from the Tools menu.
(i.e. “Tools” --> “Flat-Process DEM Data (Experimental)” or “Tools” --> “Flat-Process DEM (Enhanced Experimental)”

Note: If you have performed a stream burn-in, do not attempt to flat-process the DEM with either the Standard or Alternate flat-processing algorithms.

If you have not performed a burn-in (Step 3), then it is possible -- but still not recommended -- to run either the Standard or Alternate flat-processing algorithms, however it is unlikely that they will produce better results than either the EXP and EEX algorithms.

Here’s what our example looks like after flat-processing using the EEX algorithm.
Cells colored red meet the Facc Threshold indicated on the GUI, Cells colored green are channel guide cells that do not meet the Facc Threshold, Cells colored yellow-orange are channel guide cells that meet the Facc threshold.

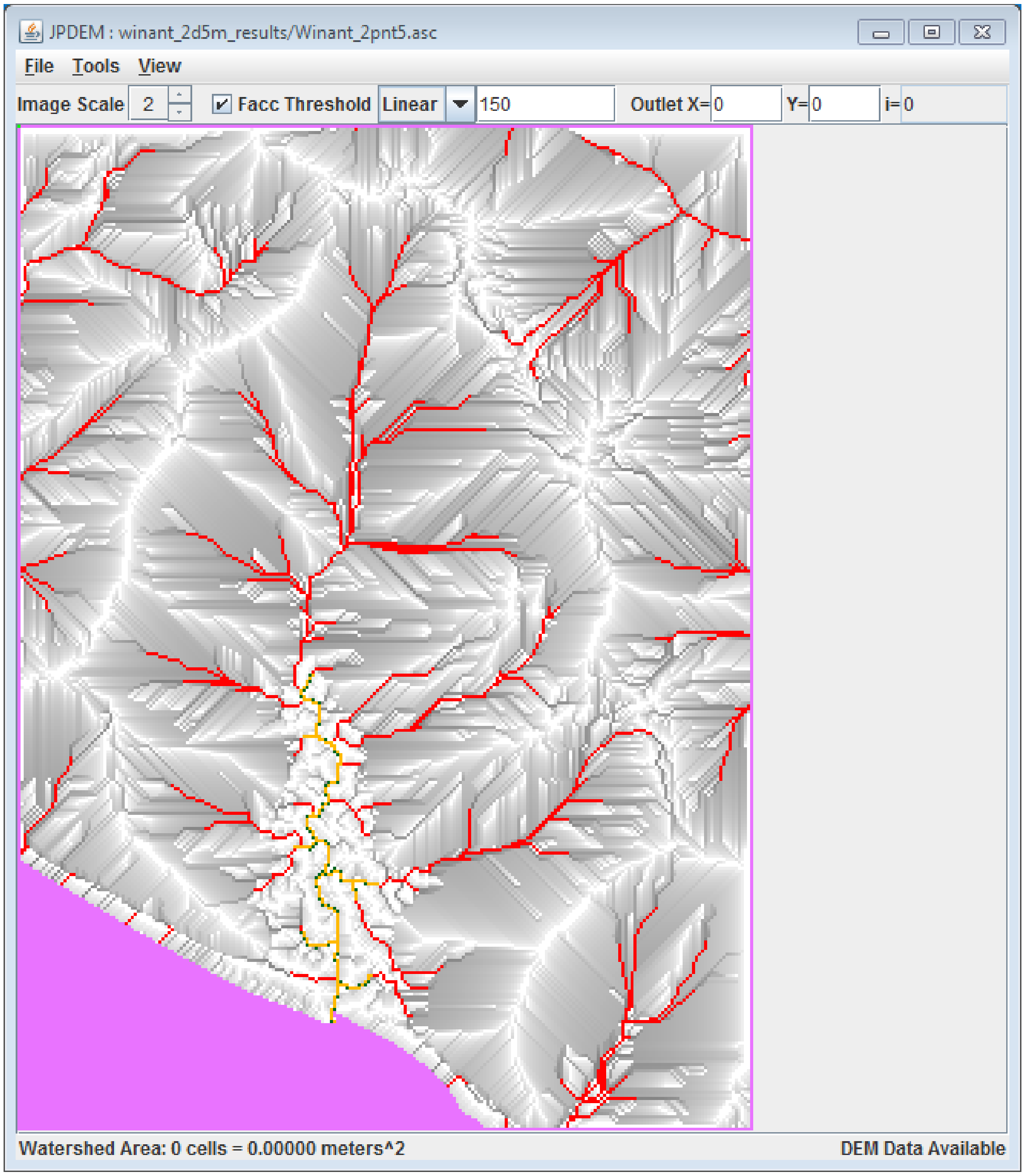

JPDEM : winant_2d5m_results/Winant_2pnt5.asc

Image Scale 2 ✓ Facc Threshold Linear 150 Outlet X= 0 Y= 0 i= 0

Watershed Area: 0 cells = 0.00000 meters^2 DEM Data Available

NOTE: Please see JPDEM tutorial B.7, “JPDEM User-Specified Border Mask Example”, which provides a more detailed example of how to create a DEM border mask that encompasses large waterbodies at the edge of a DEM. By including these flat features as part of the map border, JPDEM will ignore them during flat-processing of raw DEMs, considerably speeding up the flat-processing process.

B.7 | JPDEM User-Specified Border Mask

Overview *(Tutorial B.7 – JPDEM User-Specified Border Masks)*

This JPDEM tutorial (B.7) builds on Tutorial B.6, “JPDEM User-Specified Map Borders”, specifically to provide a more detailed example of how to create a DEM border mask that encompasses large waterbodies at the edge of the DEM. By including these flat features as part of the map border, JPDEM will ignore them during flat-processing of raw DEM maps, considerably speeding up the flat-processing process.

Example

This is an example of how to flat-process a large DEM grid in JPDEM in a reasonable amount of time, by taking advantage of JDPEM’s user-specifiable border-mask functionality.

We start with the original “raw” DEM ASCII grid (.asc) file we wish to flat process:
`thornton_mask_v2_10m.asc`

The header information for this file is:

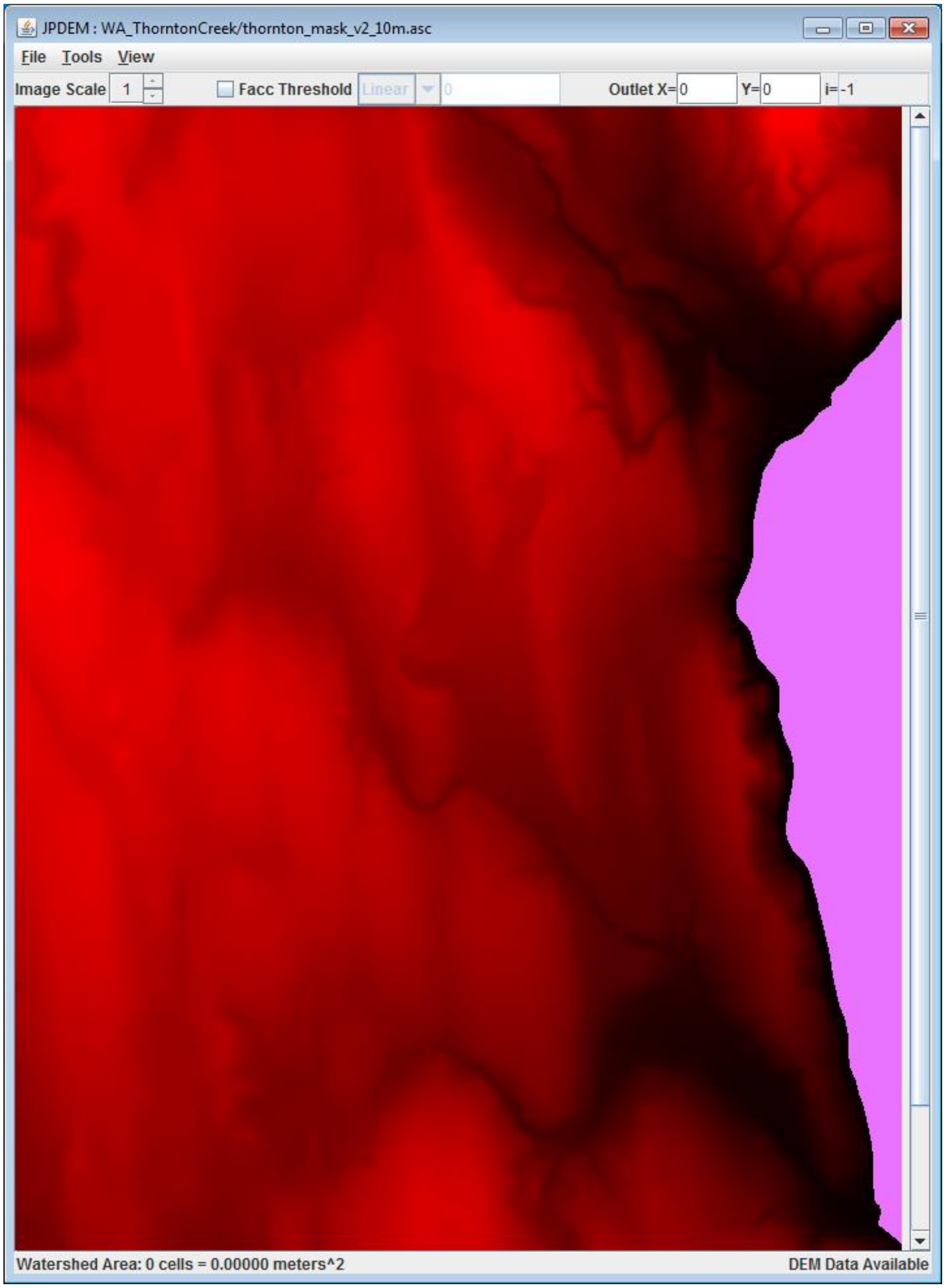
```
ncols         681
nrows         990
xllcorner     548019.221
yllcorner     5281283.0124
cellsize      10
NODATA_value  -9999
```

Initial efforts to flat-process this grid with JPDEM’s Standard (STD) and Alternate (ALT) flat-processing algorithms failed.

The grid is large enough, and has enough “flat spots” that these algorithms reach hard-wired, “unable to complete” timeout limits before all flat areas have been processed.

Attempts to flat-process this grid with the Experimental (EXP) and Enhanced Experimental (EEX) algorithms likewise failed. Unlike the STD and ALT algorithms, EXP and EEX do not have mandatory timeout limits, but neither algorithm was able to complete the flat-processing task after running continuously for more than 35 days.

Here is a screen-capture of the raw DEM file, loaded into JPDEM, with cells that have value == 0.0 highlighted in "pepto-pink":



Watershed Area: 0 cells = 0.00000 meters^2

DEM Data Available

The highlighted, "all cells are zero" area on the right side of the raw DEM has a drastic impact on all the flat-processing algorithms, but because that area of uniform-value cells lies against an edge of the grid, the EXP and EEX algorithms in particular spend an inordinate amount of their processing time ensuring flow through that area and to the grid edge.

We can tell JPDEM to ignore the problem “all cells are zero” area by providing it with a border mask map that marks all the zero value cells as part of “off-limits” border of the grid. JPDEM *always* requires a border for flat-processing, and only flat-processes cells that lie within the border (“interior cells”). Loading an .asc file into JPDEM “as DEM” automatically designates the top and bottom rows, and the left and right-most columns of cells as the border. However, when an .asc file is loaded into JPDEM “raw”, no border is created, and the “Load Border Mask” menu item can subsequently be used to specify an arbitrary border that is applied to the raw-loaded DEM values.

A border mask .asc file’s grid values must be either “0” (for border cells) or “1” for (interior cells). The values must also be integers (i.e. “0.0” or “1.0” is not allowed).

A border mask must always include the default border as border cells; i.e. the top and bottom rows and left-most and right-most columns must be marked as border cells.

GIS tools like ArcMap can be used to create border mask .asc files of arbitrary complexity and detail, but for the thornton_mask_v2_10m.asc file, we can take advantage of the fact that the border we wish to create is either a cell in the top or bottom row, left or right-most column OR has a cell value of “0.0”.

We can encode this relatively simple set of rules into a small Awk script and run it in a Cygwin Bash command shell. A Python script would work as well, and we could run that under a DOS command prompt or Windows PowerShell console, but the Awk code is more succinct.

Note that the Awk script developed in this example is not a general-purpose script; it takes advantages of various specifics of the thornton_mask_v2_10m.asc file’s contents.

Here is the Awk script in its entirety:

```
# TOP of MaskMaker.awk
BEGIN {
    IGNORECASE = 1;
    headRows = 6;
}
# get the number of data rows from the header information
/^nrows/  {
    dataRows = $2;
}
NR <= headRows {
    print;
}
# handle data rows, transforming and emitting them on-the-fly
NR > headRows {
    if (NR == (headRows + 1) || (NR - headRows) == dataRows) {
        for (i=1; i<=NF; ++i) {
            printf("%s%s",(i==1?"":" "),"0");
        }
        printf("\n");
    } else {
        for (i=1; i<=NF; ++i) {
            s = "0";
            if (i > 1 && i < NF) s = $i == 0 ? "0" : "1";
            printf("%s%s",(i==1?"":" "),s);
        }
        printf("\n");
    }
}
# BOT of MaskMaker.awk
```

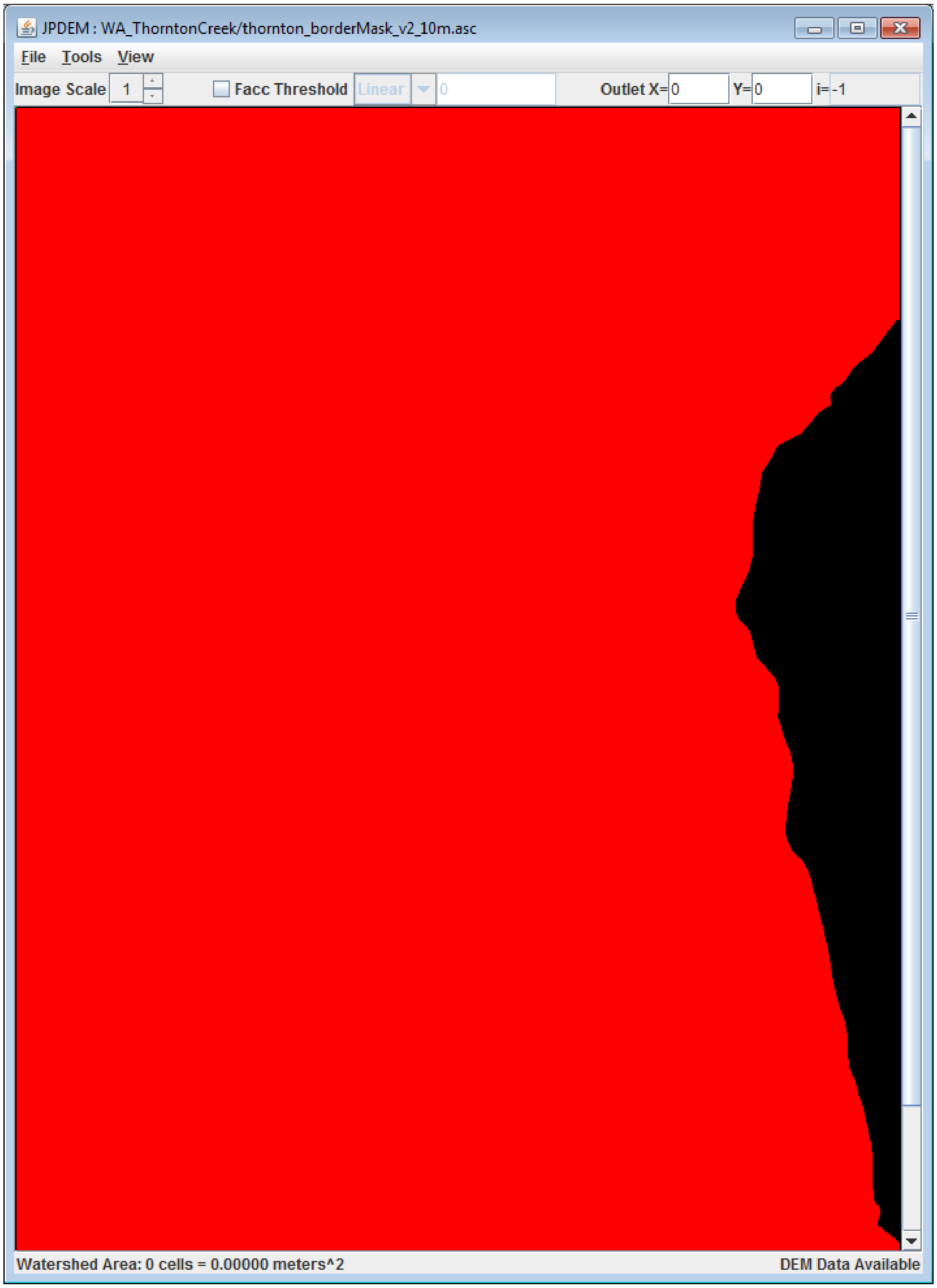
We can generate a border mask .asc file for the thornton_mask_v2_10m.asc file by running the following command line in a Cygwin Bash console:

```
$ awk -f MaskMaker.awk thornton_mask_v2_10m.asc > thornton_borderMask_v2_10m.asc
```

Depending upon the mode of the Cygwin Base console, we may also need to translate the newline characters of the resulting mask file to run under Windows:

```
$ unix2dos thornton_borderMask_v2_10m.asc
unix2dos: converting file thornton_borderMask_v2_10m.asc to DOS format...
```

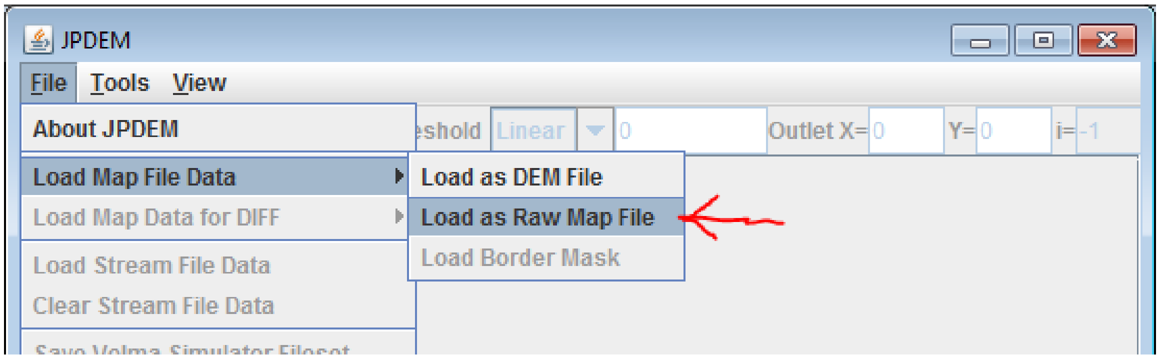
Here’s the resulting thornton_borderMask_v2_10m.asc file, “as Raw Map File” into JPDEM. Red (interior) cells have value 1, and black (border) cells have value 0:



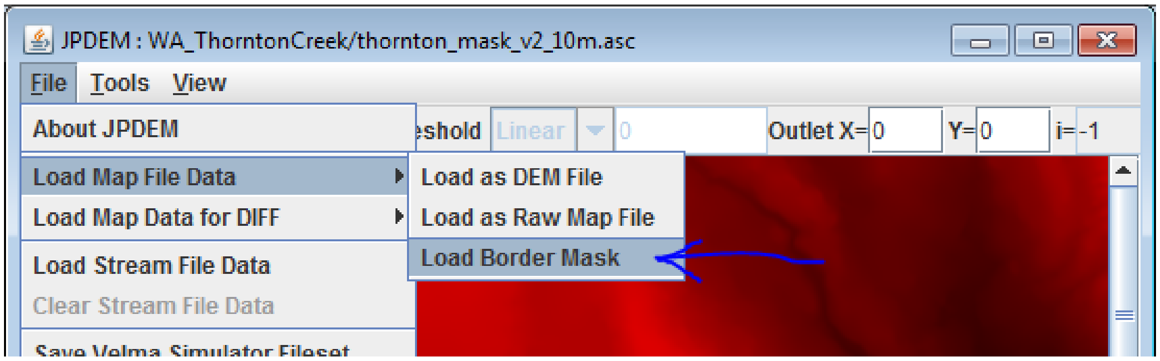
Watershed Area: 0 cells = 0.00000 meters^2

Now that we have a raw DEM map, and a border mask to overlay onto it, it's simply a matter of…

Loading the DEM file "**as Raw Map File**" data . . .

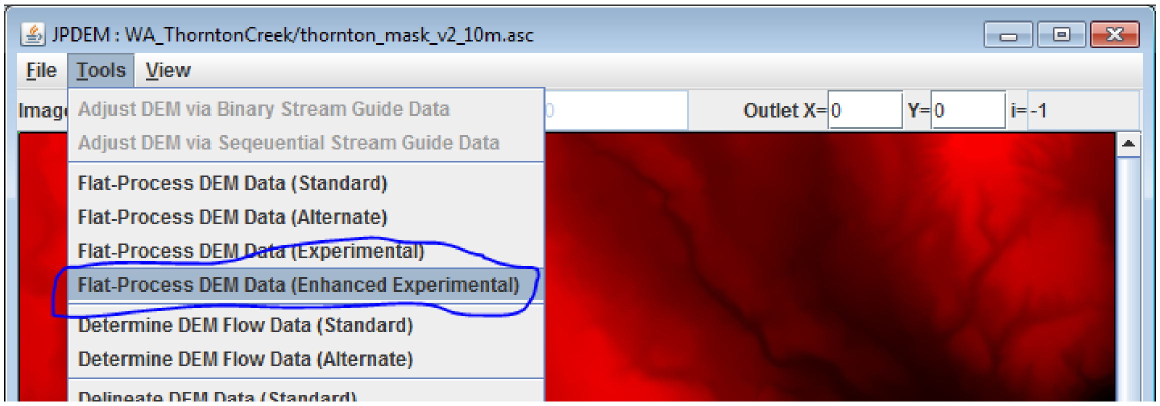


Loading the border mask file as "**Border Mask**" data . . .

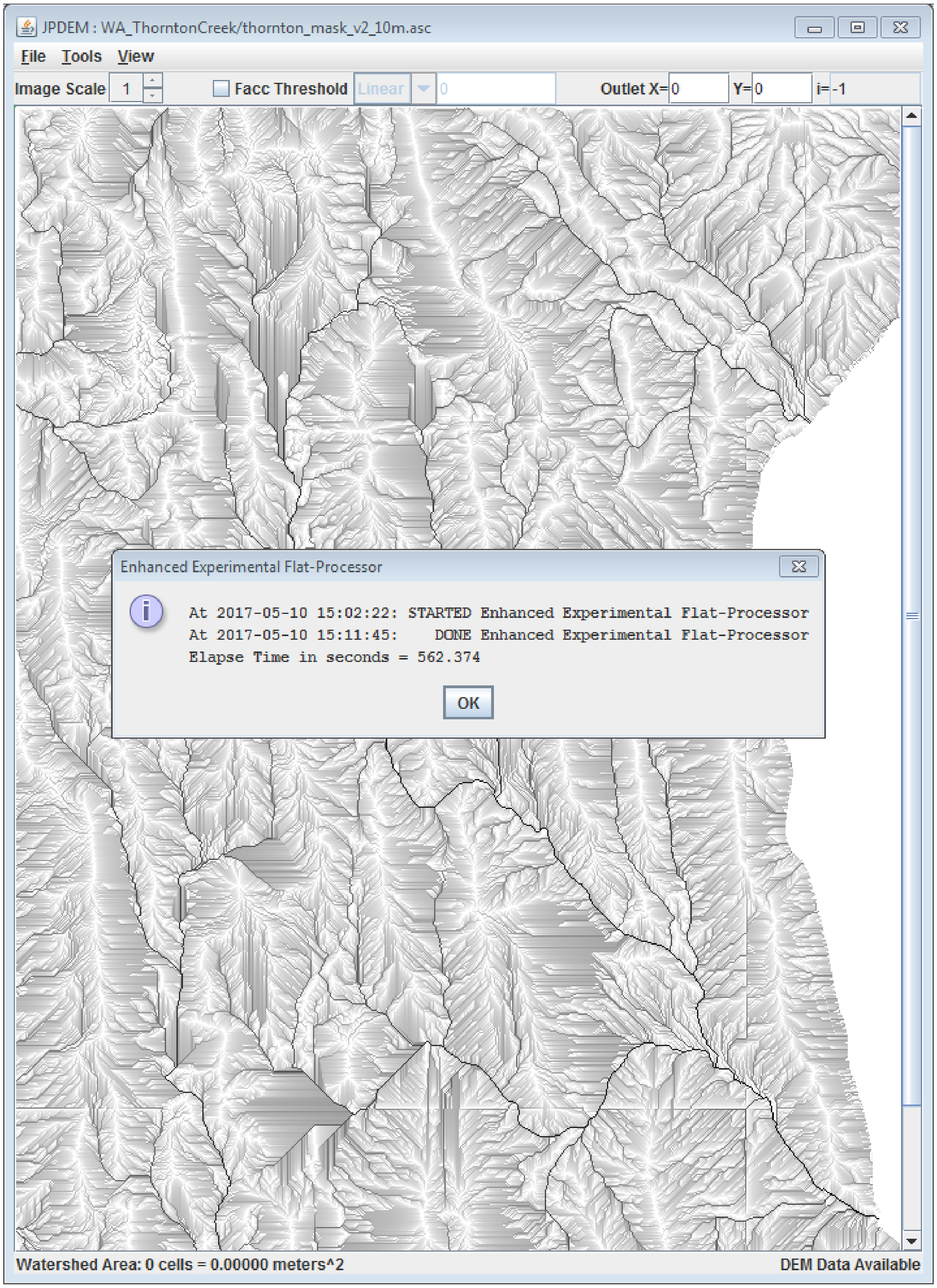


and finally, running a flat-processing algorithm on the combined DEM + custom border.

The STD and ALT flat-processors are *not* recommended for custom-border situations. Instead, the best bet is the EEX flat-processor (the EXP *may* work, but is not ideal) . . .


JPDEM : WA_ThorntonCreek/thornton_mask_v2_10m.asc
File
Tools
View
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Seqeuential Stream Guide Data
Flat-Process DEM Data (Standard)
Flat-Process DEM Data (Alternate)
Flat-Process DEM Data (Experimental)
Flat-Process DEM Data (Enhanced Experimental)
Determine DEM Flow Data (Standard)
Determine DEM Flow Data (Alternate)
Outlet X=0
Y=0
i=-1

Here is the EEX flat-processed map, showing the completion message:


JPDEM : WA_ThorntonCreek/thornton_mask_v2_10m.asc
File Tools View
Image Scale 1
Facc Threshold Linear 0
Outlet X=0 Y=0 i=-1
Enhanced Experimental Flat-Processor
OK

```
At 2017-05-10 15:02:22: STARTED Enhanced Experimental Flat-Processor
At 2017-05-10 15:11:45:    DONE Enhanced Experimental Flat-Processor
Elapse Time in seconds = 562.374
```

Watershed Area: 0 cells = 0.00000 meters^2 DEM Data Available

Here are the console messages from the sequence of steps: Load/raw, Load/border, EEX:

```
LoadPropertyFilesAction:
file="D:\Users\kdjang\Velma_SiteData\WA_ThorntonCreek\thornton_mask_v2_10m.asc"

hasData=true ncol=681 nrow=990 xll=548019.221 yll=5281283.0124 isLlCenter=false
cell=10.0 ndata=-9999.0 zmin=0.0 zmax=167.9841 delx=0 dely=0 minLogFacc=0.0
maxLogFacc=0.0

LoadDmMaskFileAction:
file="D:\Users\kdjang\Velma_SiteData\WA_ThorntonCreek\thornton_borderMask_v2_10m.asc"

At 2017-05-10 15:02:22: Launching Enhanced Experimental Flat-Processor thread ...

Experimental Flat-Processor: processed 622821 cells

Experimental Flat-Processor: largest set processed = 4234

Internal Data State Report

Map Summary: Total Cols=681 Total Rows=990 Total Cells=674190

NoData cells: Actual=0

Buffer Cells: Expected=3338 Actual=51369 Suspect=0 !!! WARNING !!!

Interior Cells: Expected=670852 Actual=622821 Suspect=0 !!! WARNING !!!

At 2017-05-10 15:02:22: STARTED Enhanced Experimental Flat-Processor

At 2017-05-10 15:11:45:    DONE Enhanced Experimental Flat-Processor

Elapse Time in seconds = 562.374
```

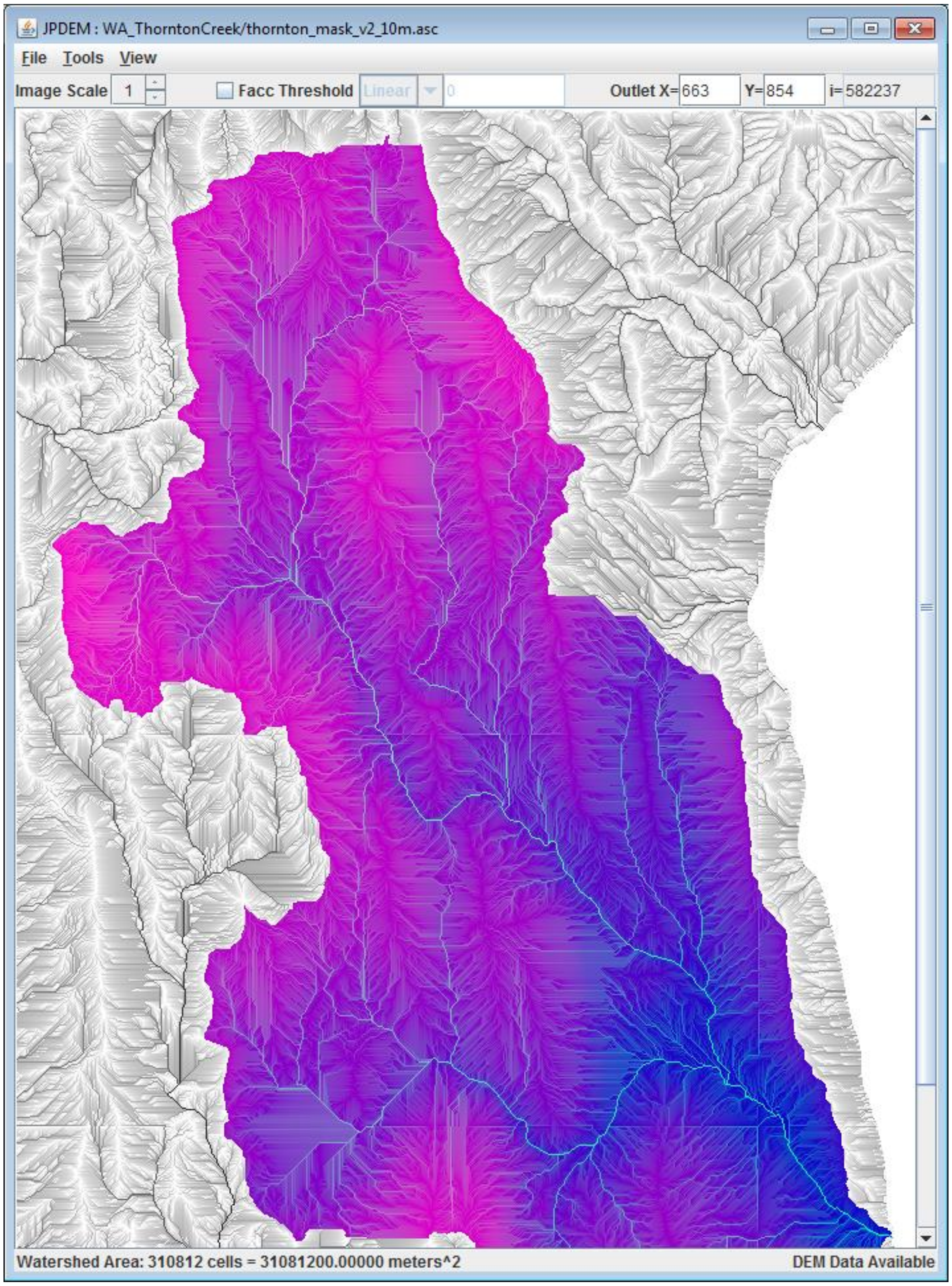
Some observations:

- The running time has dropped from many weeks to just under 10 minutes: a truly significant improvement in running time.
- The “!!! WARNING !!!” messages are triggered because the border is not the default, single-edge-cells border; future versions of the JPDEM code may be made “smarter” about issuing this warning.
- “Buffer Cells” refers to border cells.

What About Stream Channel Cutting/Burn-In?

To include a channel cut/burn-in in the above example, we would simply have performed the sequence of steps to load a stream channel guide file and run the cutter after the border mask load, and before running the EEX flat-processor.

Here is a delineation of the flat-processed map. The delineation does not actually touch the bottom grid edge as it appears to – the grid is simply taller than the JPDEM window:



Outlet X=663 Y=854 i=582237

Watershed Area: 310812 cells = 31081200.00000 meters^2

B.8 | JPDEM for Viewing Map Differences

Overview *(B.8 – JPDEM Viewing Map Differences)*

This tutorial explains how to use JPDEM to compare elevation differences within a DEM before and after it is flat-processed. This quality assurance (QA) step allows users to assess the veracity of JPDEM’s flat-processing procedure. That is, where and to what extent did JPDEM alter the elevations of cells within the DEM? Note that this procedure is but a first step in a QA process that may include comparisons of a flat-processed DEM to other types of data not included in this tutorial. For example, does the flat-processed DEM provide flow routing (Facc values – see Tutorial B.3) consistent with mapped stream locations represented by the National Hydrography Dataset (NHD; https://www.usgs.gov/core-science-systems/ngp/national-hydrography/nhdplus-high-resolution)?

The JPDEM application’s File menu contains a “Load Map Data For DIFF” item that provides a way to compare two DEM files across all cells (the maps must have the same dimensions). Although its’ abilities are rudimentary relative to comparisons you could perform with true GIS systems (e.g. ESRI’s tools) JPDEM’s DIFFerencing function is a quick way to – for instance – compare a flat-processed map against the original, raw DEM map. (This is, in fact, precisely the use-case for which the DIFF function was added.)

The DIFF function allows you to load a second DEM .asc file after loading a first .asc file.
The DIFF function replaces the first DEM file’s cell data with the difference between the second and the first file’s cell data, at every cell. After the DIFF function is completed, JPDEM’s cell data contains the difference values, and the display automatically shifts to the “DEM Difference Image Rendering Mode” – cells that differ should be relatively easy to spot in this mode.

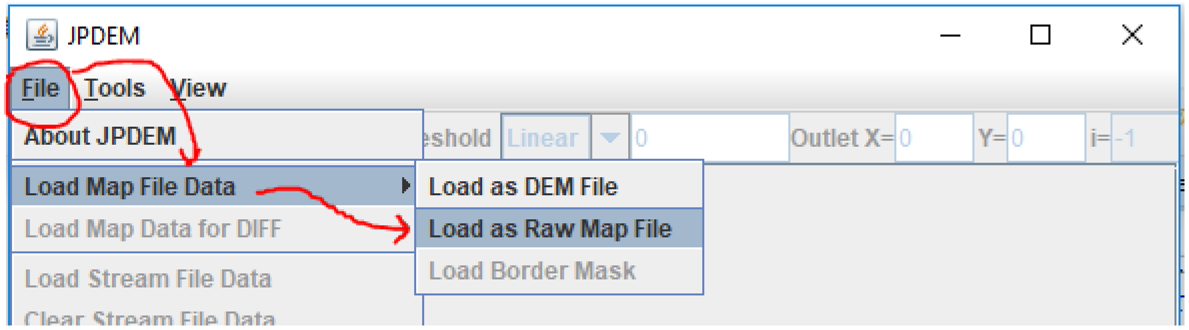
A DIFF Example

Starting Assumption: e=we have both an original, raw DEM map .asc file, *and* have already flat-processed that DEM map in JPDEM and saved the results into a second .asc file that is also available.

The DIFF function calculates per cell differences as New Value – Current Value, where “New Value” refers to the file loaded for DIFF. To express the differences between an original .asc DEM file and a flat-processed .asc version of that DEM file such that a positive change in the processed .asc is expressed as a positive value, we should load the original DEM file first, then load the processed DEM file as the DIFF file.

(continued)

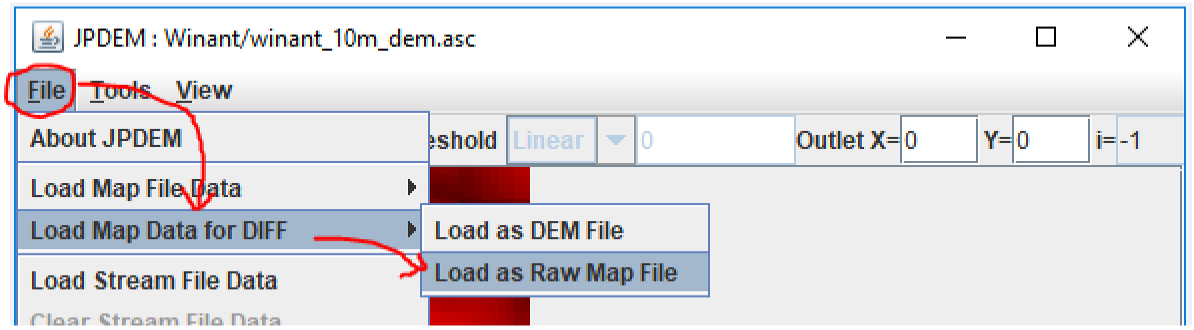
Step 1
Load the original DEM file, using File -> “Load Map File Data” -> “Load as Raw Map File”.



It *is* possible to use “Load as DEM File”, but when possible, load the original and the diff file as “raw” data.

In the “Select DEM File to Load” dialog that opens, find the original raw DEM .asc file, select it, and click “Load File” to load it into JPDEM as the current map data.

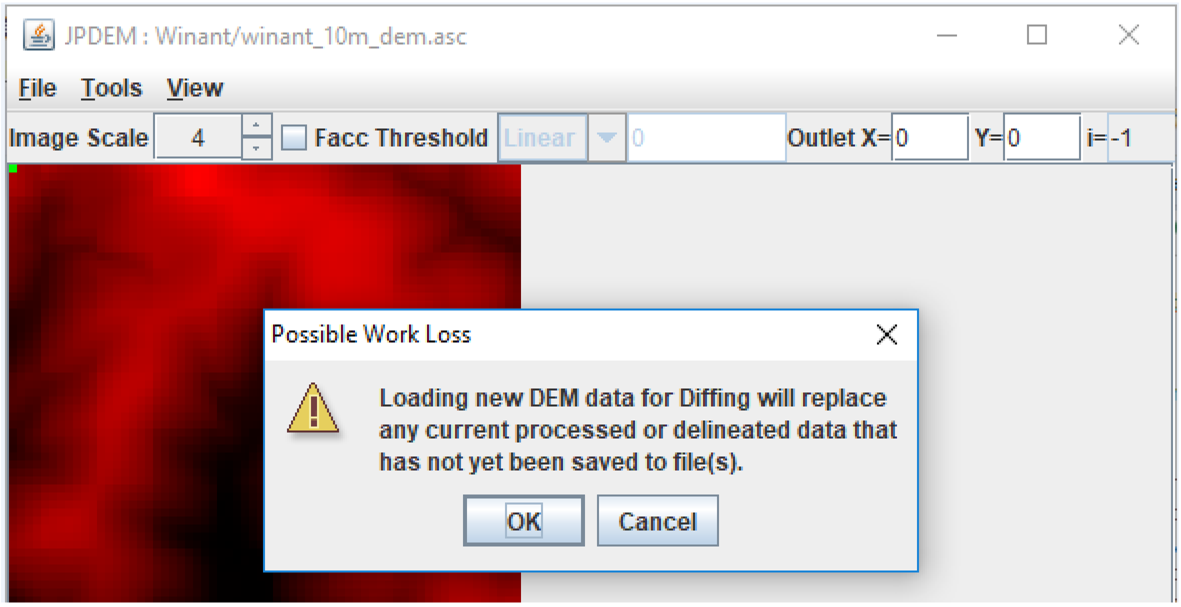
Step 2
Load the flat-processed DEM file, using File -> “Load Map Data for DIFF” -> “Load as Raw Map File”.



The flat-processed .asc file takes on the role as the “diff” map, the original .asc file is the current map.

Because the DIFF function replaces JPDEM’s current map data with difference values, a warning dialog opens before the difference map file finder dialog:

(continued)



Possible Work Loss

Loading new DEM data for Diffing will replace any current processed or delineated data that has not yet been saved to file(s).

Click OK to proceed to the “Select Raw File to Load for DIFF” dialog and “Load File” load the diff file.

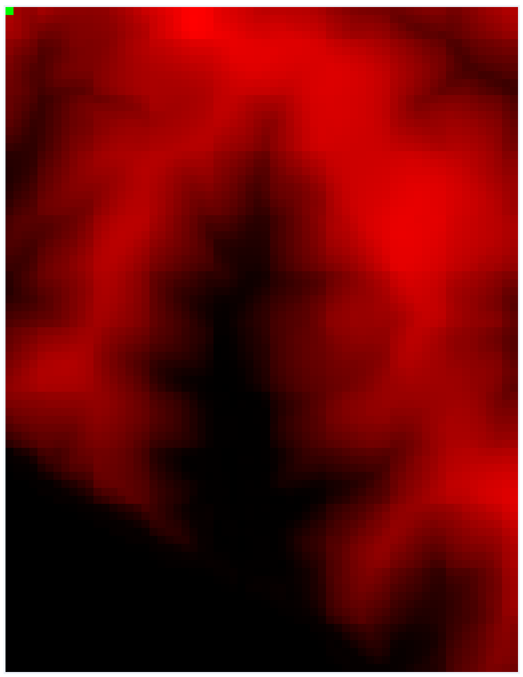
After the cell differences are calculated, JPDEM will contain (and display) the differences for each cell in the grid. The DEM Difference Image Rendering view is active:

- diff file cell value < map cell value = RED (darker -> more negative)
- diff file cell value = map cell value = WHITE
- diff file cell value > map cell value = GREEN (darker -> more positive)
- The current Outlet cell = GRAY
- Highlighted cells = CYAN

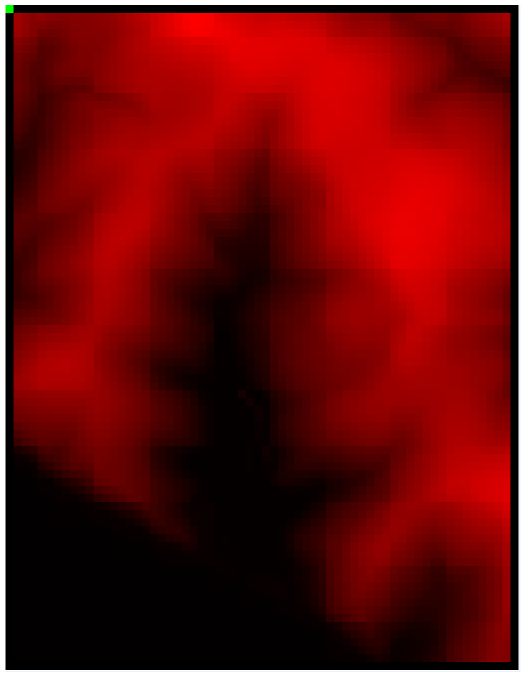
(continued)

Here is a comparison of our original raw DEM, the flat-processed DEM, and the difference display:

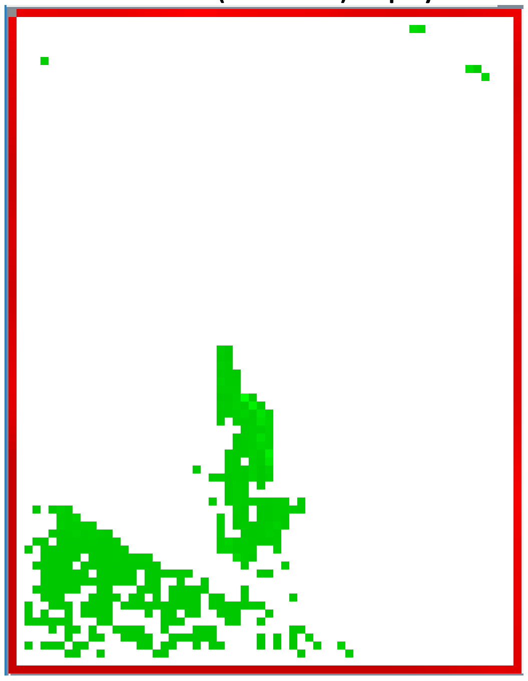
Original DEM (raw loaded)



Flat-Processed DEM (raw loaded)



Difference (Flat – Raw) Display



The DIFF display clearly indicates the areas of the map that have been altered by flat processing.

Step 3

After the DIFF function has completed, JPDEM's current map data is the set of cell difference values. You can view an individual cell's values by clicking on a cell with the mouse right-button. The Facc and fdir values, if nonzero, are invalid, but the difference value is listed as the cell's "dem=" value:

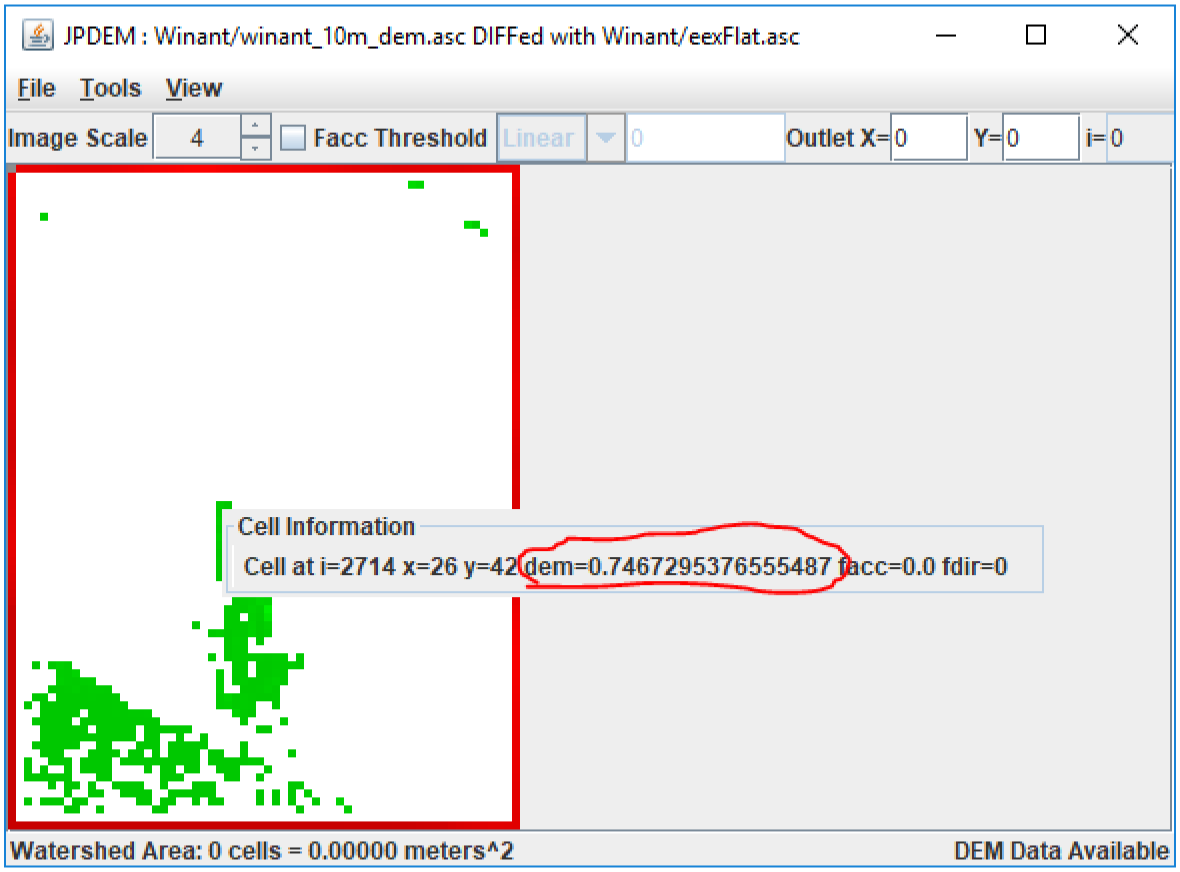

JPDEM : Winant/winant_10m_dem.asc DIFFed with Winant/eexFlat.asc

Image Scale 4 Facc Threshold Linear 0 Outlet X=0 Y=0 i=0

Cell Information

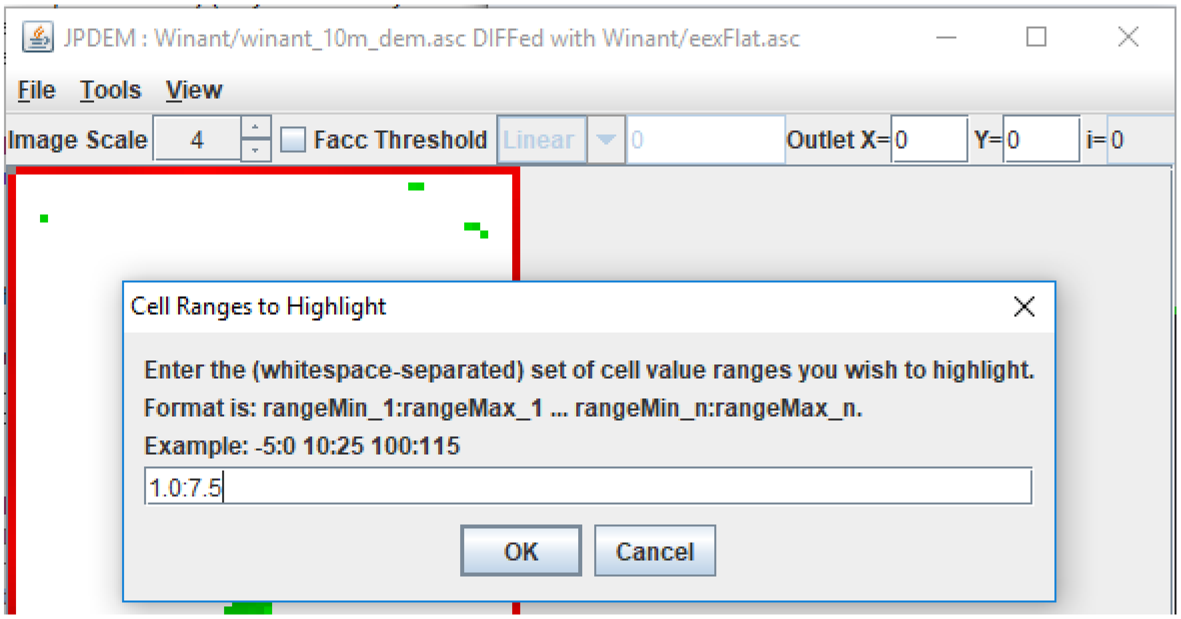
Cell at i=2714 x=26 y=42 dem=0.7467295376555487 facc=0.0 fdir=0

Watershed Area: 0 cells = 0.00000 meters^2 DEM Data Available

Use the View -> "Highlight Specific Cells in Image" -> "Specify Cells by Value Range" function to find differences within a specific range of difference.

Here we want to find all cells raised between 1.0 and 7.5 meters relative to the original DEM:

(continued)

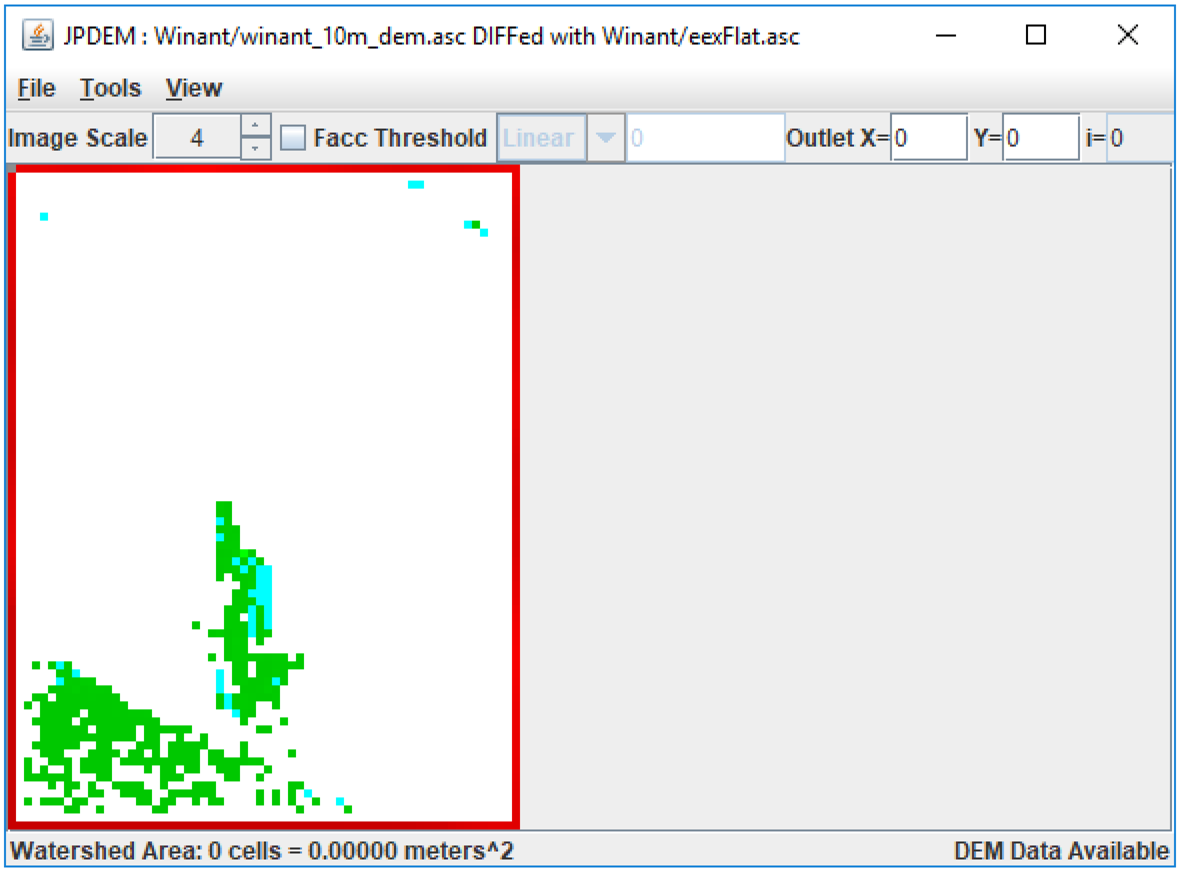


Enter the (whitespace-separated) set of cell value ranges you wish to highlight.
Format is: rangeMin_1:rangeMax_1 ... rangeMin_n:rangeMax_n.
Example: -5:0 10:25 100:115

```
1.0:7.5
```

(continued)

The result highlights the specified cells in cyan, rather than the usual "pepto-pink":


JPDEM : Winant/winant_10m_dem.asc DIFFed with Winant/eexFlat.asc
File
Tools
View
Image Scale
4
Facc Threshold
Linear
0
Outlet X=
0
Y=
0
i=
0

Watershed Area: 0 cells = 0.00000 meters^2 DEM Data Available

As noted in the Overview (page 1), this difference mapping procedure is but a first step in a QA process that may include comparisons of a flat-processed DEM to other types of data not included in this tutorial. For example, does the flat-processed DEM provide flow routing (Facc values – see Tutorial B.3) consistent with mapped stream locations represented by the National Hydrography Dataset (NHD; https://www.usgs.gov/core-science-systems/ngp/national-hydrography/nhdplus-high-resolution)?

B.9 | JPDEM Sub-Reach Delineations

Overview *(Tutorial B.9 – JPDEM Sub-Reach Delineations)*

Large watersheds can include many sub-watersheds, each draining to their respective sub-reach outlets. VELMA users have the option of simulating the overall watershed using a single outlet, or breaking up the watershed into multiple delineated sub-watersheds (each with a sub-reach outlet) and simulating them in parallel. The latter option can be much faster computationally, especially for very large watersheds.

This tutorial describes how to use JPDEM to establish a set of sub-reach outlets, in preparation for using VELMA to simulate, in parallel, daily streamflow for a designated set of sub-watersheds / sub-reach outlets.

Note: Additional tutorials (C.1 – C.3) explain how to run VELMA in parallel mode. When running in parallel mode, VELMA is programmed to collect simulated daily streamflow information for each designated sub-reach outlet. VELMA uses that information to compute, for each sub-reach outlet, the sum of flow contributions from (1) all upstream grid cells draining to the sub-reach outlet, and (2) only those cells falling between the sub-reach outlet and the next upstream sub-reach outlet (thereby excluding flow from all other upstream cells).

Thus, aside from potential gains in computational efficiency, sub-reach delineations for running VELMA in parallel mode can help watershed managers tease apart streamflow responses to local and more distant upstream changes in land cover and land use.

Warning: The functionality describe here is an initial draft that will be refined when the development schedule permits.

The JPDEM application provides a collection of functions that, when applied in concert, allow you to determine a set of outlets for sub-reaches of an overall watershed's area.

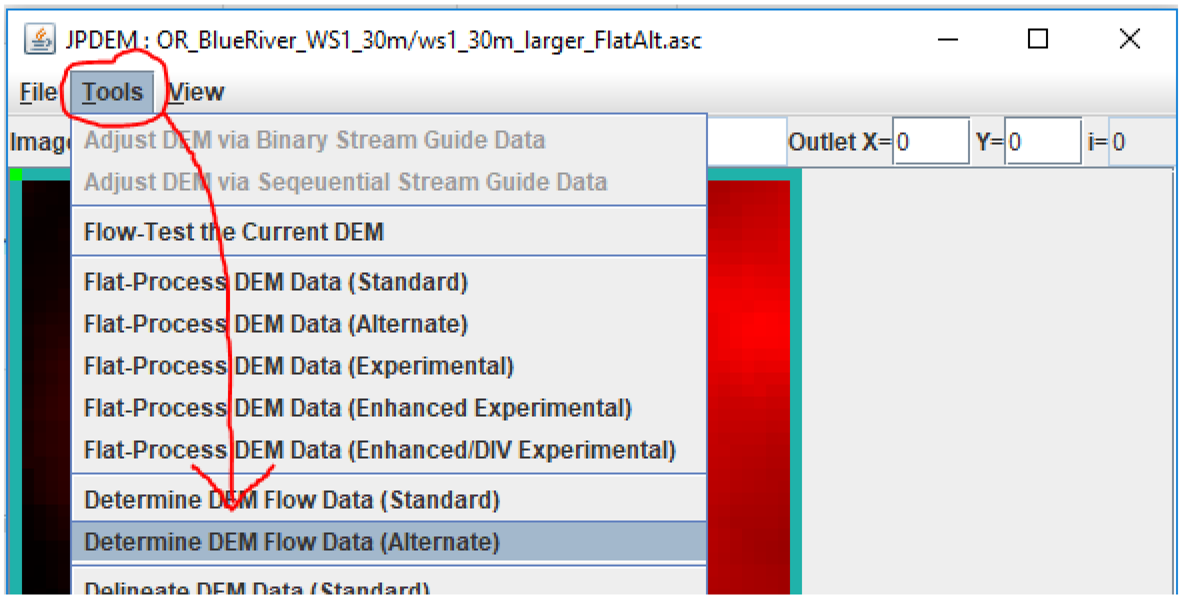
Once identified, a set of outlets can be used to configure a JVelma simulation so that runoff information is generated for each of the sub-reach outlets.

JPDEM Must Have Flat-Processed DEM and Flow Data Available for Delineation

Use JPDEM to flat-process the map containing your overall watershed before you use JPDEM to determine the sub-reach outlet locations. Flat-process the raw DEM .asc file, save it to a new .asc file, and then close JPDEM, restart it, and load the new, flat-processed .asc file. If you already have a flat-processed .asc file available, you can simply start JPDEM and load it.

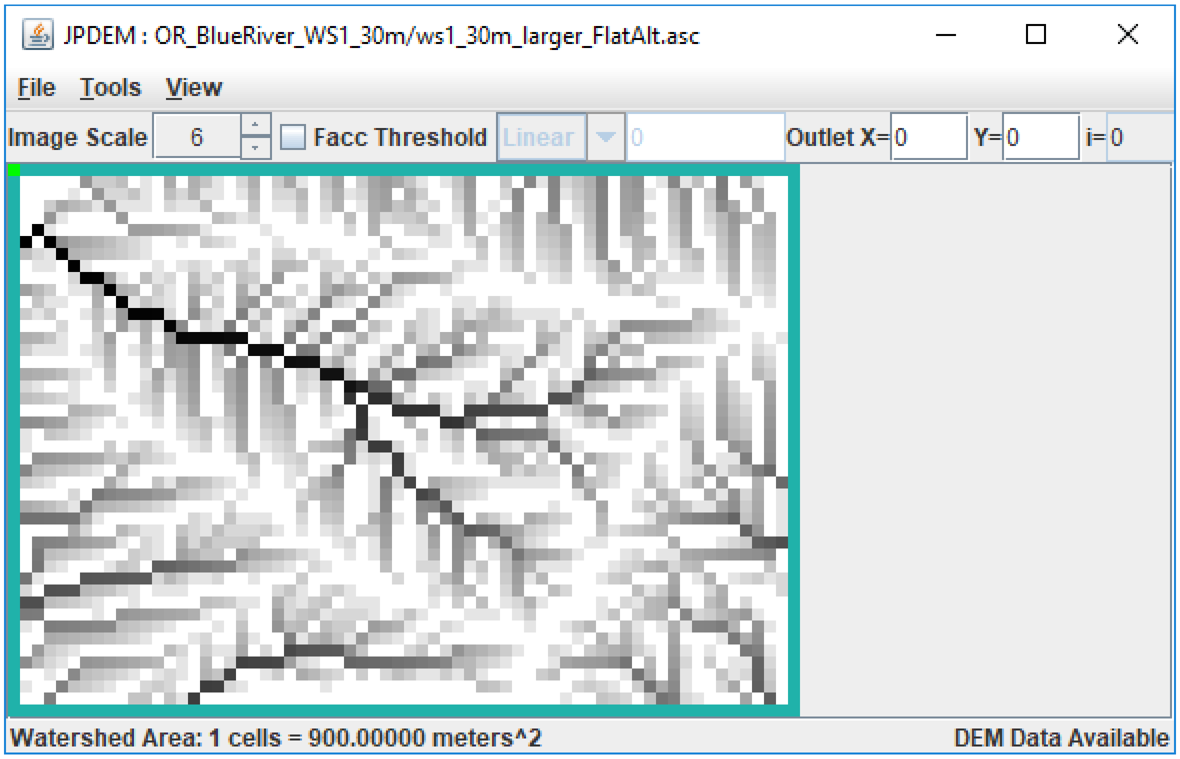
Once you have flat-processed DEM map loaded in JPDEM, use the "Determine DEM Flow" tool to generate flow and direction information. (The delineation tools need this information to figure how which cells flow to which reach outlets.)

Click the "Tools" -> "Determine DEM Flow Data (Alternate)" menu item, as shown in the following image:

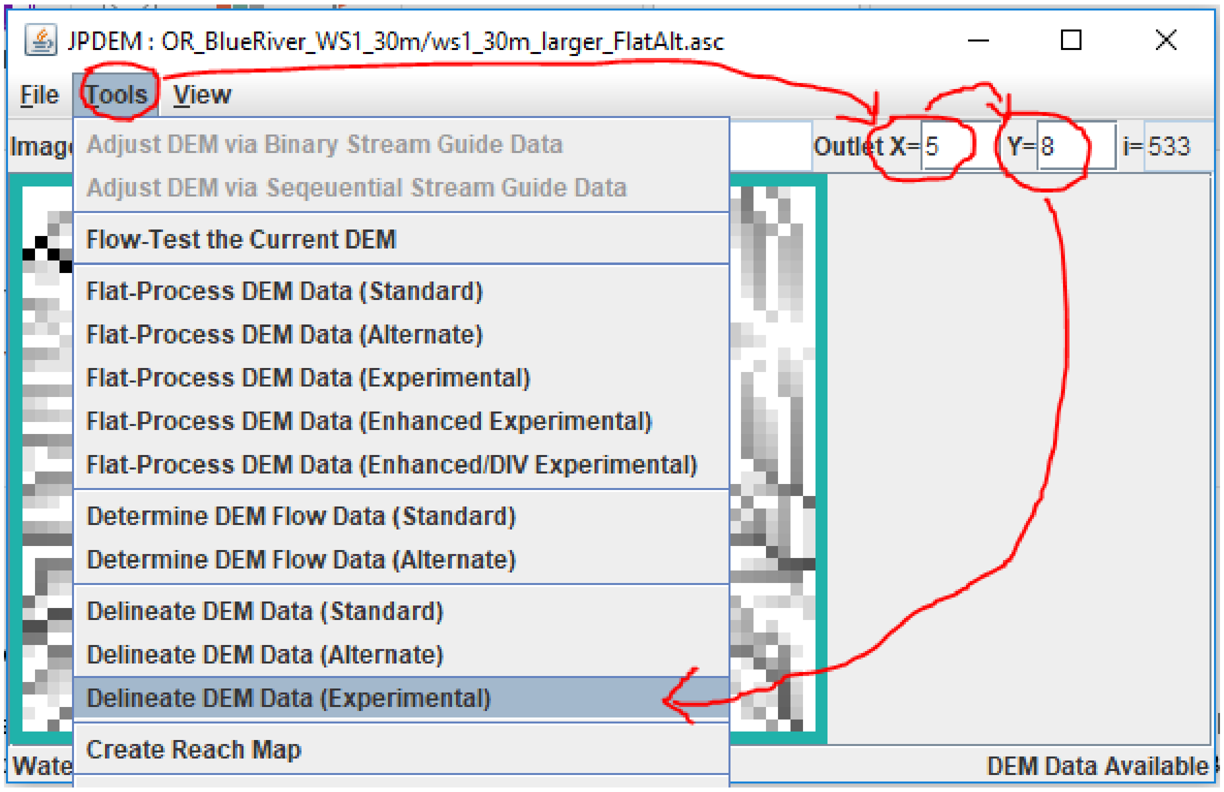

JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc
File
Tools
View
Image
Outlet X= 0
Y= 0
i= 0
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Seqeuential Stream Guide Data
Flow-Test the Current DEM
Flat-Process DEM Data (Standard)
Flat-Process DEM Data (Alternate)
Flat-Process DEM Data (Experimental)
Flat-Process DEM Data (Enhanced Experimental)
Flat-Process DEM Data (Enhanced/DIV Experimental)
Determine DEM Flow Data (Standard)
Determine DEM Flow Data (Alternate)
Delineate DEM Data (Standard)

Delineate the Overall Watershed First

After JPDEM determines the flat-processed DEM’s flow data, the map display should so a grayscale image like this:


JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc
File
Tools
View
Image Scale
6
Facc Threshold
Linear
0
Outlet X= 0
Y= 0
i= 0
Watershed Area: 1 cells = 900.00000 meters^2
DEM Data Available

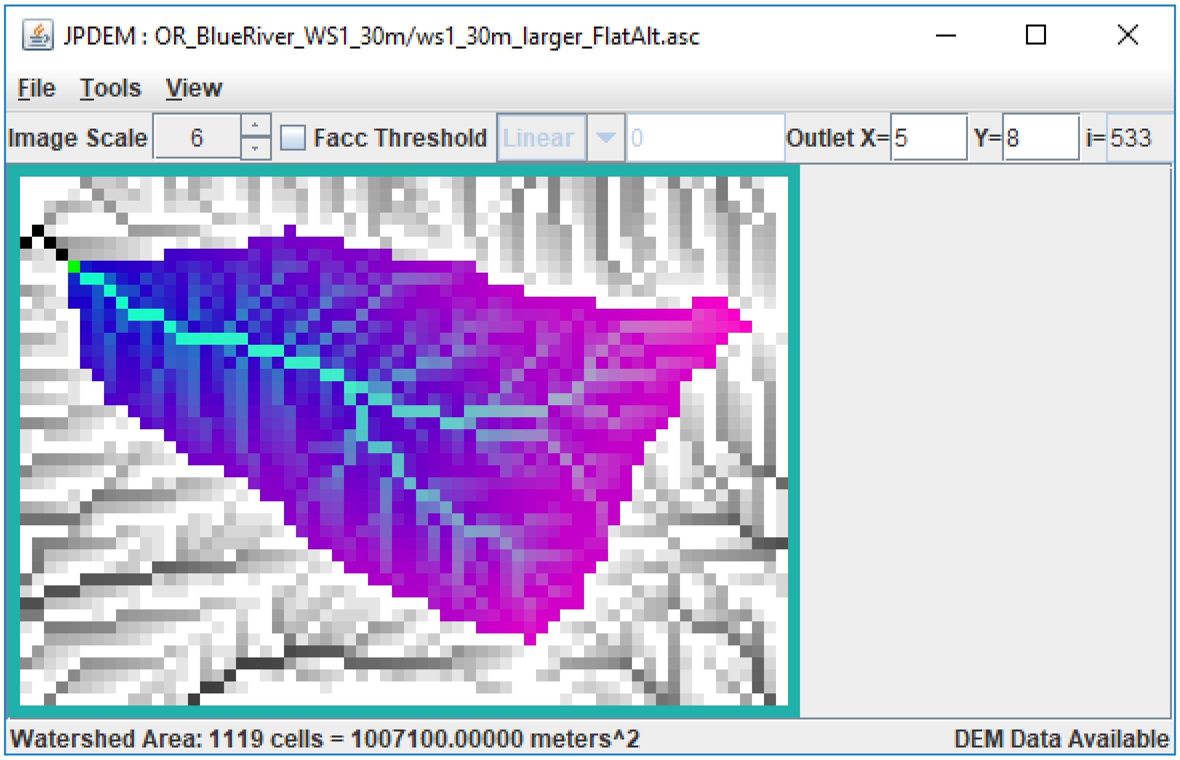
Set the Outlet X= and Y= fields to the x,y coordinates of the primary outlet for your watershed (i.e. the cell location of the pour point for the entire watershed), then click the "Tools" -> "Delineate DEM Data (Experimental)" menu item.


JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc
File
Tools
View
Imag
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Seqeuential Stream Guide Data
Flow-Test the Current DEM
Flat-Process DEM Data (Standard)
Flat-Process DEM Data (Alternate)
Flat-Process DEM Data (Experimental)
Flat-Process DEM Data (Enhanced Experimental)
Flat-Process DEM Data (Enhanced/DIV Experimental)
Determine DEM Flow Data (Standard)
Determine DEM Flow Data (Alternate)
Delineate DEM Data (Standard)
Delineate DEM Data (Alternate)
Delineate DEM Data (Experimental)
Create Reach Map
Outlet X= 5
Y= 8
i= 533
Wate
DEM Data Available

(continued)

Display the Overall Delineation, and the High-Flow Paths

Once it is delineated, the watershed is clearly marked in JPDEM’s map display:



Outlet X=5 Y=8 i=533

Watershed Area: 1119 cells = 1007100.00000 meters^2

Usually, you’ll want to choose sub-reach outlet locations at forks in the highest-flow paths of the watershed. You can see those paths by clicking the “Facc Threshold” checkbox, then typing a cell count in the numeric field to its right.

In the example below, the Facc Threshold is set to 75, which means that any cell colored red has 75 or more cells flowing into it.

(continued)

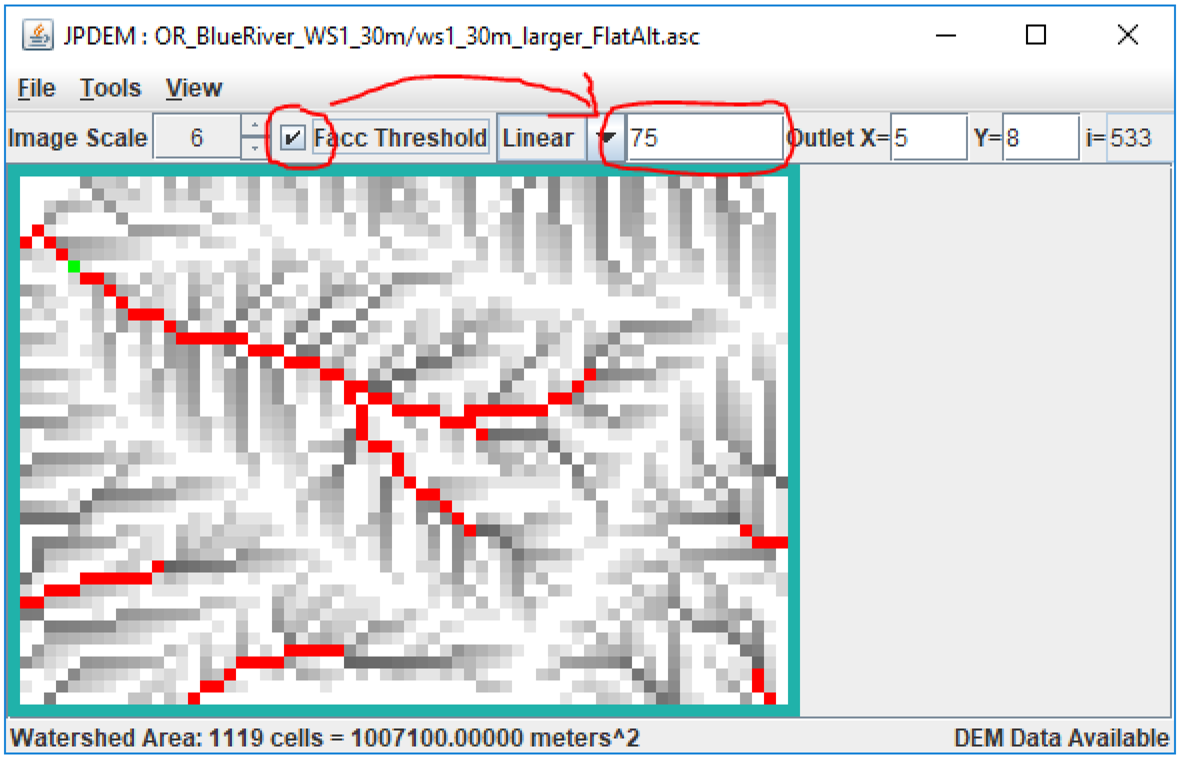

JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc

Image Scale 6 ✓ Facc Threshold Linear 75 Outlet X= 5 Y= 8 i= 533

Watershed Area: 1119 cells = 1007100.00000 meters^2 DEM Data Available

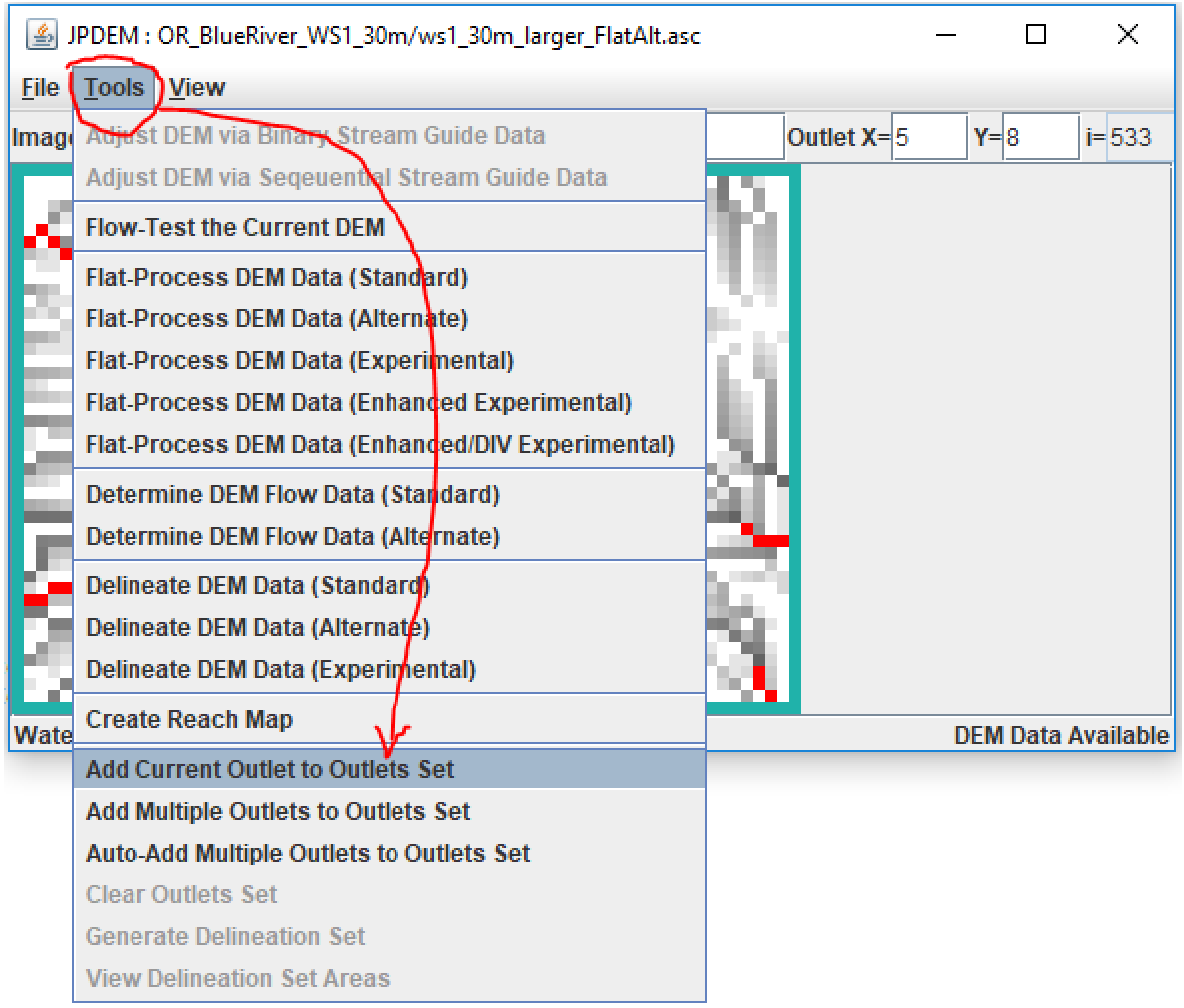
Note that when the Facc Threshold indicator is checked, the watershed area is not shown. You can switch back and forth between the two display modes by repeatedly clicking the Facc Threshold checkbox.

Determining Sub-Reach Outlets Manually

Using the Delineation and Facc Threshold displays, you can see which cells are in the overall watershed, and which of those in-watershed cells form the highest flow paths. The process of determining the cell locations of sub-reaches is as follows:

First, with the Outlet X and Y values set to the overall watershed's outlet location, click the "Tools" -> "Add Current Outlet to Outlets Set":

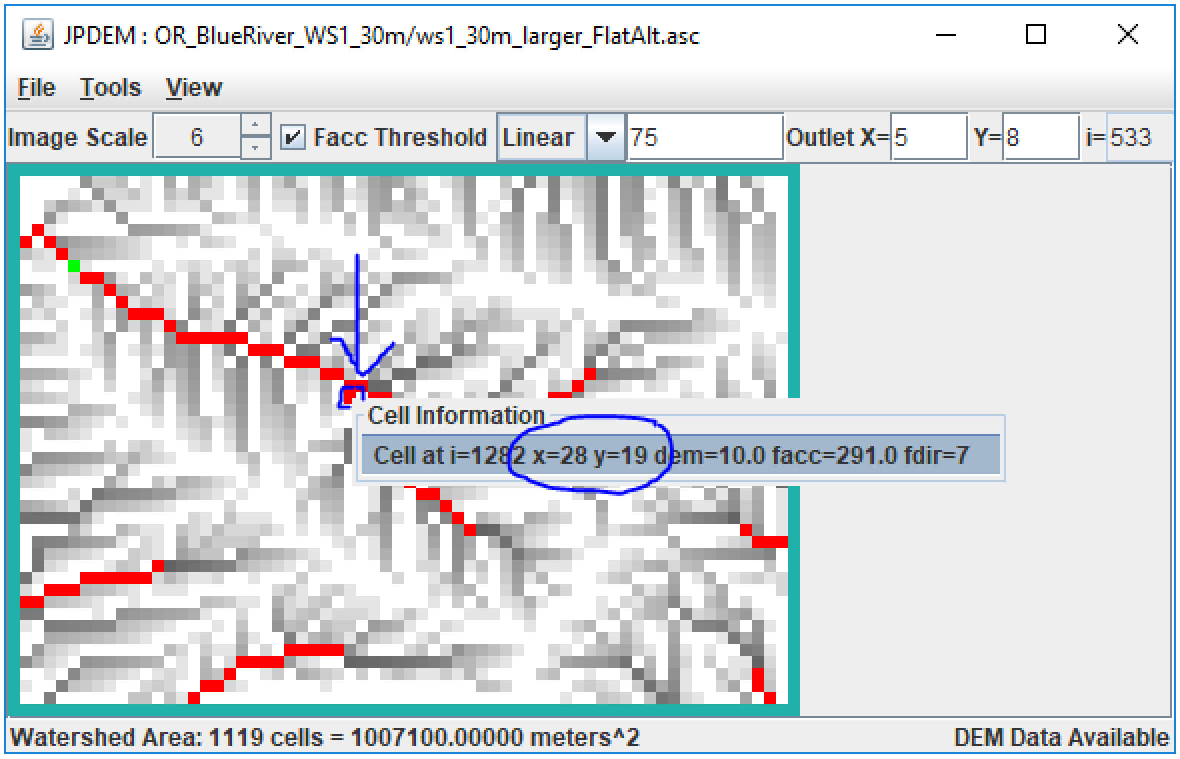
(continued)


JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc
File
Tools
View
Outlet X= 5
Y= 8
i= 533
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Seqeuential Stream Guide Data
Flow-Test the Current DEM
Flat-Process DEM Data (Standard)
Flat-Process DEM Data (Alternate)
Flat-Process DEM Data (Experimental)
Flat-Process DEM Data (Enhanced Experimental)
Flat-Process DEM Data (Enhanced/DIV Experimental)
Determine DEM Flow Data (Standard)
Determine DEM Flow Data (Alternate)
Delineate DEM Data (Standard)
Delineate DEM Data (Alternate)
Delineate DEM Data (Experimental)
Create Reach Map
Add Current Outlet to Outlets Set
Add Multiple Outlets to Outlets Set
Auto-Add Multiple Outlets to Outlets Set
Clear Outlets Set
Generate Delineation Set
View Delineation Set Areas
DEM Data Available

The “current outlet” is the outlet whose X and Y coordinates are specified in upper right numeric fields of JPDEM’s status bar.

Once the overall watershed’s outlet is recorded, adding other, sub-reach outlets is simply a matter of changing the current outlet X and Y values, and click the “Add Current Outlet …” menu item again. You can directly enter the X and Y value of each new sub-reach outlet into the Outlet X=, Y= fields, but if you don’t know the exact coordinates, you can left-click, then hold-left-button + drag-to-select “Cell at …” – when you un-hold the left mouse button, the current outlet will change to the selected “Cell at …” and you can then add it using the “Add Current Outlet …” menu item.

(continued)



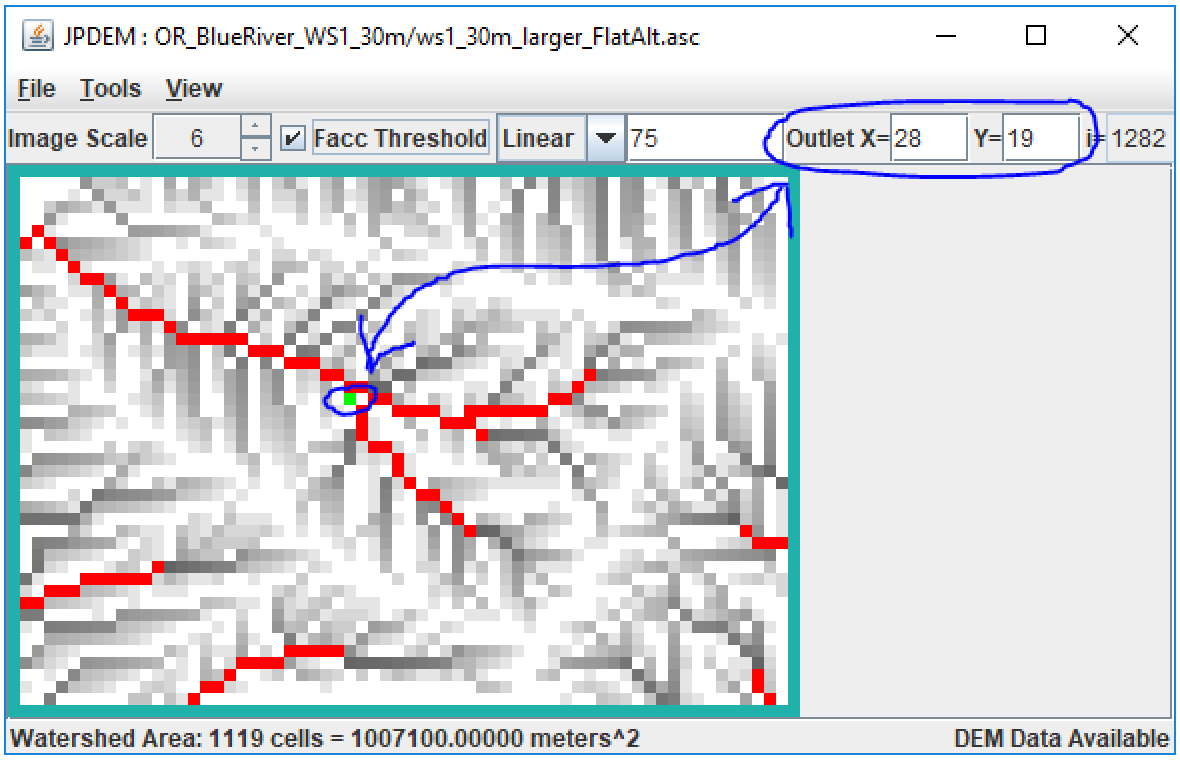
JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc

Image Scale 6 ☑ Facc Threshold Linear 75 Outlet X=5 Y=8 i=533

Cell Information

Cell at i=1282 x=28 y=19 dem=10.0 facc=291.0 fdir=7

Watershed Area: 1119 cells = 1007100.00000 meters^2 DEM Data Available



JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc

Image Scale 6 ☑ Facc Threshold Linear 75 Outlet X=28 Y=19 i=1282

Watershed Area: 1119 cells = 1007100.00000 meters^2 DEM Data Available

Unfortunately, if you add a cell to the outlet set by mistake, there’s no way to undo that particular cell’s addition. You must click the “Tools” -> “Clear Outlets Set” menu item, and the start over, re-adding the overall watershed outlet, and then all the sub-reach outlet cell’s in turn.

JPDEM doesn’t currently provide any confirmation when you “Add Current Outlet …”, however it does echo a statement to the console window it was started from.

Here’s an example, showing the addition of the overall watershed outlet (i=533) and three other cells as sub-reaches:

```
[ . . . ]
After clear, outlets set is {}
After add, outlets set is {533}
After add, outlets set is {1282, 533}
After add, outlets set is {1217, 1282, 533}
After add, outlets set is {1217, 1282, 1491, 533}
[ . . . ]
```

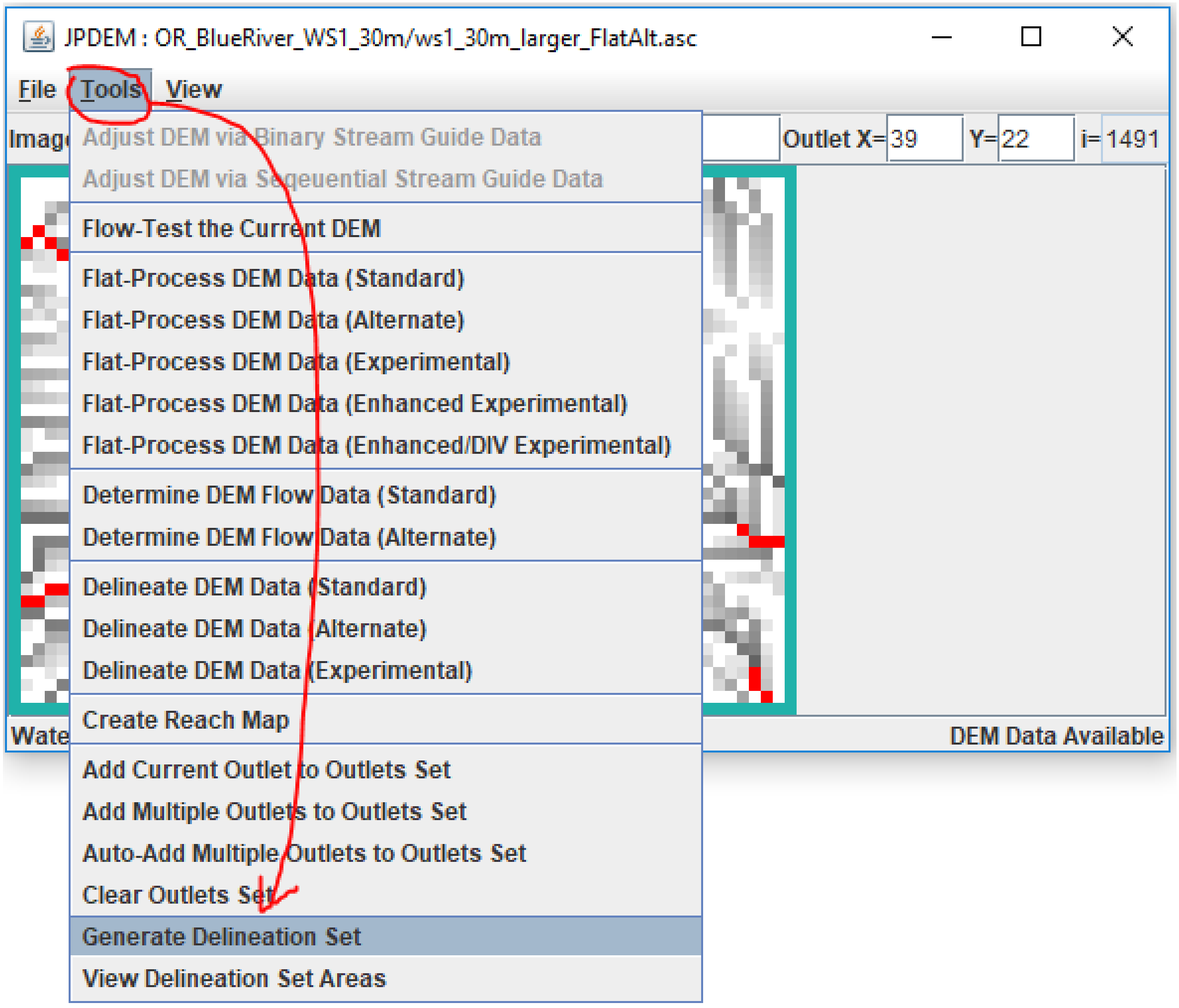
Note that the console status statement identifies cells by their linear indices, not their x,y coordinates. A cell’s linear index is its index counting left to right, across rows and down columns starting from the upper-left cell in the grid and ending with the lower-right cell. (For example, in a tiny 10x10 grid, the upper-left cell’s index is 0, and the lower-right cell’s index is 99.)

The highlighted cell indices above are exactly the values that JVelma needs to track runoff for the sub-reaches you’ve specified. You can copy them from the console window to the system clipboard (highlight them with the mouse, then press return), or simply write the numbers down on a piece of paper for reference when you configure your simulation in JVelma.

However, if you started JPDEM by double-clicking the .jar file, there is no console log and the above output won’t be available. Also, the above doesn’t provide the X, Y coordinates, which are handy if you want to set Cell Data Writers in JVelma for each sub-reach.

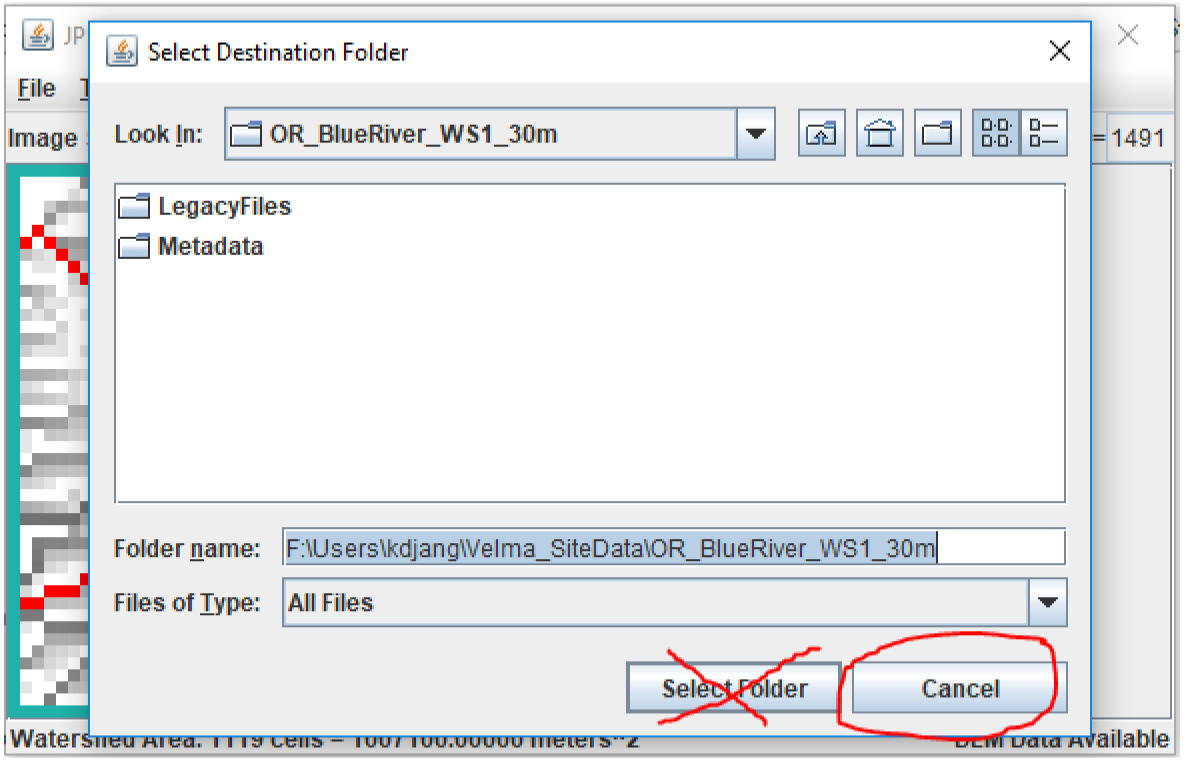
(continued)

JPDEM does provide a way to view the Outlet Set, but it is a bit confusing.
Click the “Tools” -> “Generation Delineation Set” menu item:


JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc
File
Tools
View
Imag
Outlet X= 39
Y= 22
i= 1491
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Seqeuential Stream Guide Data
Flow-Test the Current DEM
Flat-Process DEM Data (Standard)
Flat-Process DEM Data (Alternate)
Flat-Process DEM Data (Experimental)
Flat-Process DEM Data (Enhanced Experimental)
Flat-Process DEM Data (Enhanced/DIV Experimental)
Determine DEM Flow Data (Standard)
Determine DEM Flow Data (Alternate)
Delineate DEM Data (Standard)
Delineate DEM Data (Alternate)
Delineate DEM Data (Experimental)
Create Reach Map
Add Current Outlet to Outlets Set
Add Multiple Outlets to Outlets Set
Auto-Add Multiple Outlets to Outlets Set
Clear Outlets Set
Generate Delineation Set
View Delineation Set Areas
Wate
DEM Data Available

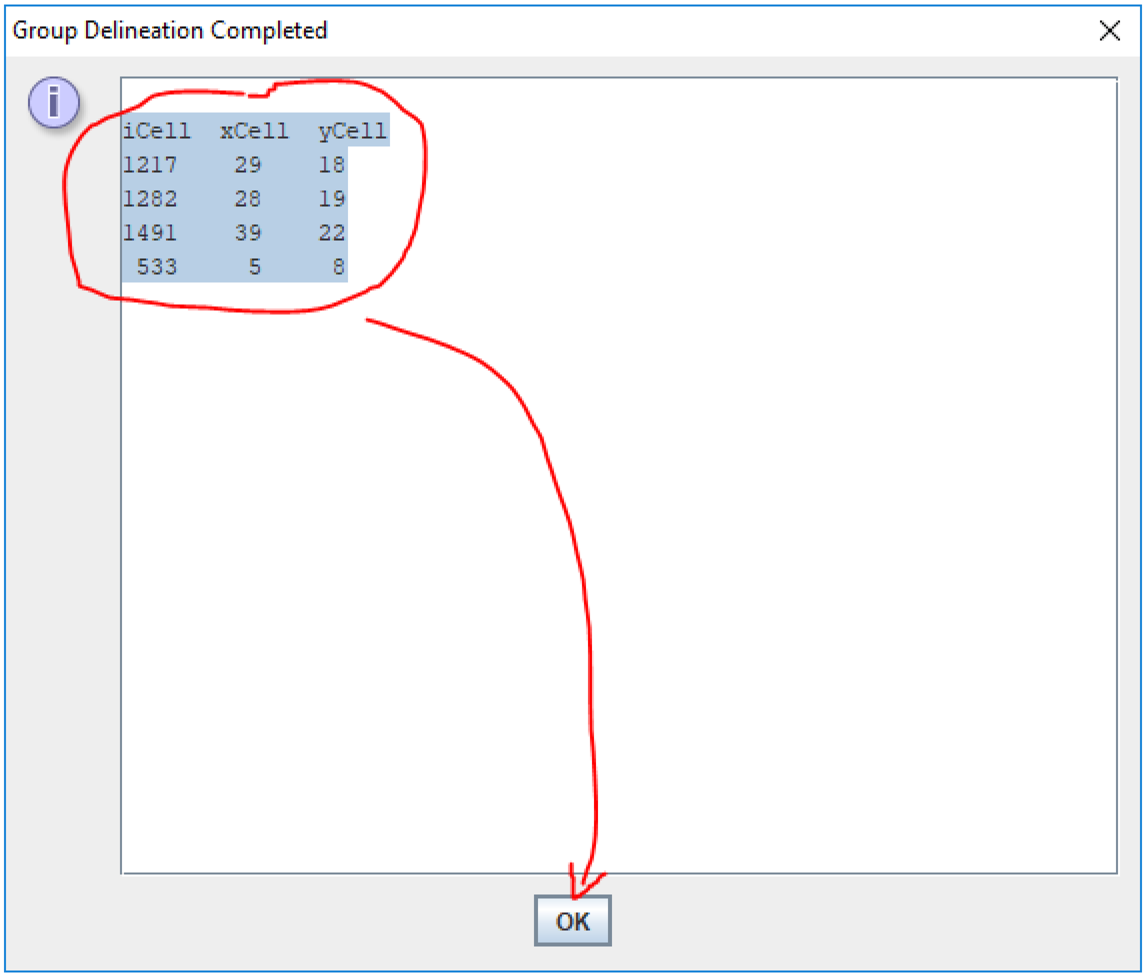
(continued)

This pops up a File Dialog – however, YOU’RE NOT ACTUALLY GOING TO GENERATE A DELINEATION SET. Simply Click the “Cancel” button:



Clicking the cancel button brings up a summary dialog window, containing the details of the set of outlets, ***without*** actually performing the “Group Delineation” that it claims is completed:

(continued)



```
iCell  xCell  yCell
1217     29     18
1282     28     19
1491     39     22
 533      5      8
```

You can highlight-select-copy the text in this window.
We recommend paste/saving it to a Notepad .txt file for later reference.

(continued)

You can also view the sub-reaches by clicking the "Tools" -> "View Delineation Set Areas" menu item, but the Image Scale is fixed at "1", which may make small maps hard to read:

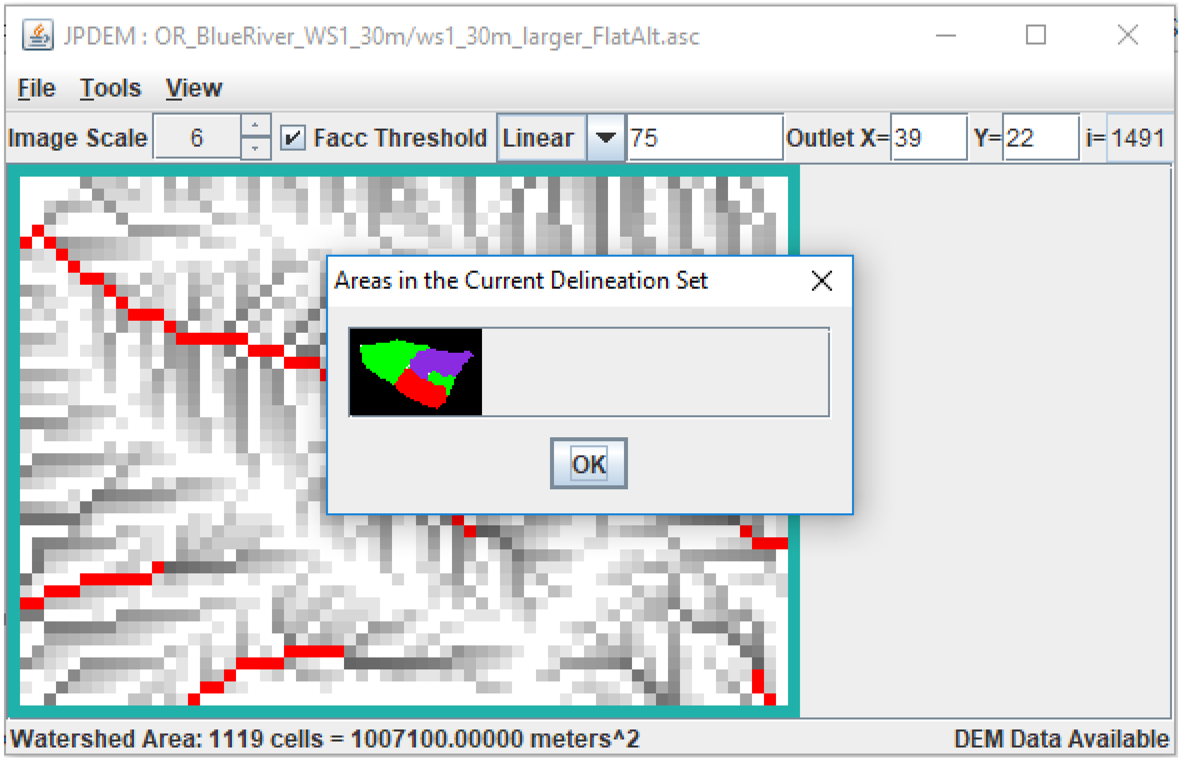

JPDEM : OR_BlueRiver_WS1_30m/ws1_30m_larger_FlatAlt.asc
File Tools View
Areas in the Current Delineation Set
OK

Image Scale 6 ✓ Facc Threshold Linear 75 Outlet X=39 Y=22 i=1491

Watershed Area: 1119 cells = 1007100.00000 meters^2 DEM Data Available

Using the Sub-Reach Delineation Outlets with JVelma

Once you've found a set of sub-reach delineations that identify the cells for which you want to determine runoff, you'll need to configure a VELMA simulation configuration (using JVelma) to use those locations.

Start JVelma, and either create a new configuration using the DEM you found sub-reaches for, or load an existing .xml configuration file that uses that DEM.

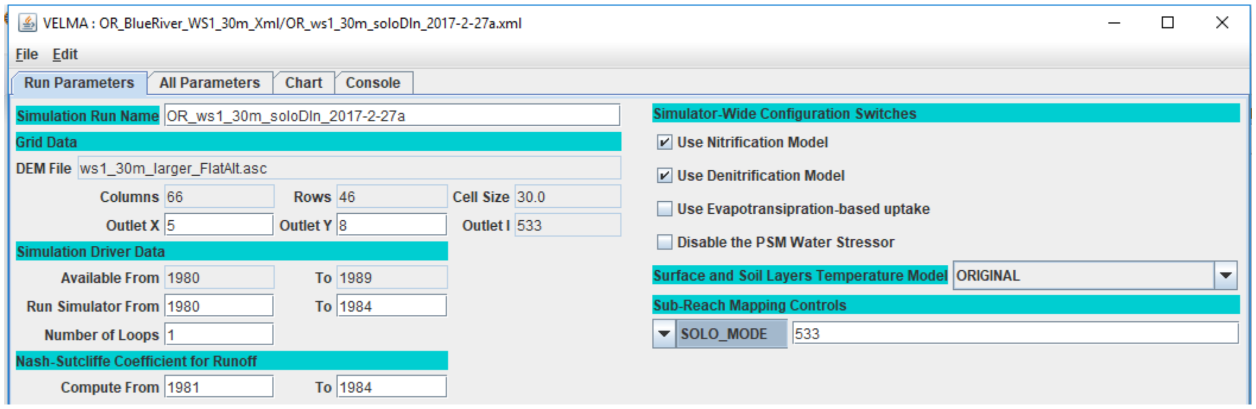
In the "Run Parameters" Tab, under "Sub-Reach Mapping Controls", click the drop-down selector and choose "MULTI_MODE", then type the overall watershed outlet index value and the other sub-reach outlet index values into the numeric field to the right of the mode selector.

The following example shows an existing .xml configuration file being changed from solo mode to multi.

(continued)

T

The original configuration:


VELMA : OR_BlueRiver_WS1_30m_Xml/OR_ws1_30m_soloDln_2017-2-27a.xml
File
Edit
Run Parameters
All Parameters
Chart
Console

Simulation Run Name: OR_ws1_30m_soloDln_2017-2-27a

Grid Data
DEM File: ws1_30m_larger_FlatAlt.asc
Columns: 66 Rows: 46 Cell Size: 30.0
Outlet X: 5 Outlet Y: 8 Outlet I: 533

Simulation Driver Data
Available From: 1980 To: 1989
Run Simulator From: 1980 To: 1984
Number of Loops: 1

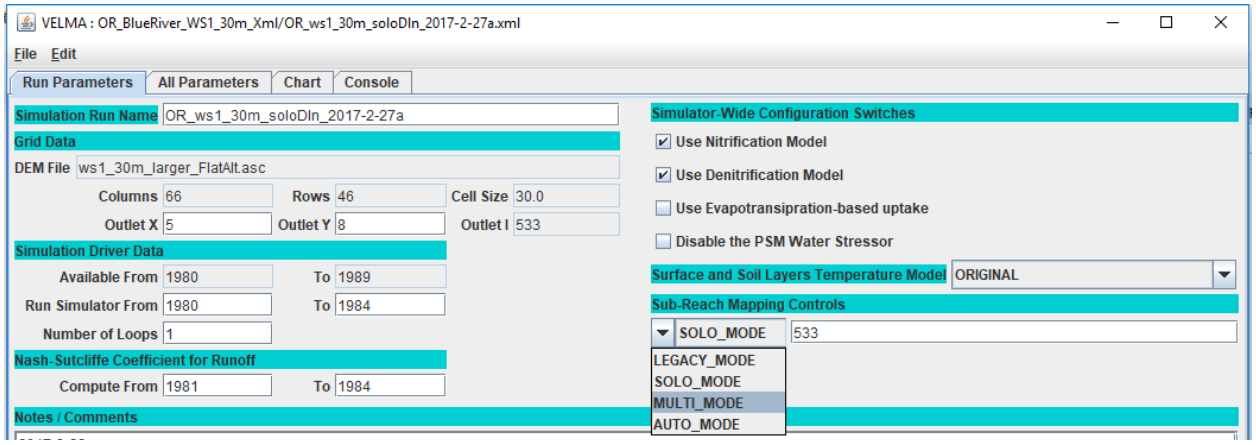
Nash-Sutcliffe Coefficient for Runoff
Compute From: 1981 To: 1984

Simulator-Wide Configuration Switches
☑ Use Nitrification Model
☑ Use Denitrification Model
☐ Use Evapotranspiration-based uptake
☐ Disable the PSM Water Stressor

Surface and Soil Layers Temperature Model: ORIGINAL

Sub-Reach Mapping Controls
SOLO_MODE 533

Setting the mode:



Simulation Run Name: OR_ws1_30m_soloDln_2017-2-27a

Grid Data
DEM File: ws1_30m_larger_FlatAlt.asc
Columns: 66 Rows: 46 Cell Size: 30.0
Outlet X: 5 Outlet Y: 8 Outlet I: 533

Simulation Driver Data
Available From: 1980 To: 1989
Run Simulator From: 1980 To: 1984
Number of Loops: 1

Nash-Sutcliffe Coefficient for Runoff
Compute From: 1981 To: 1984

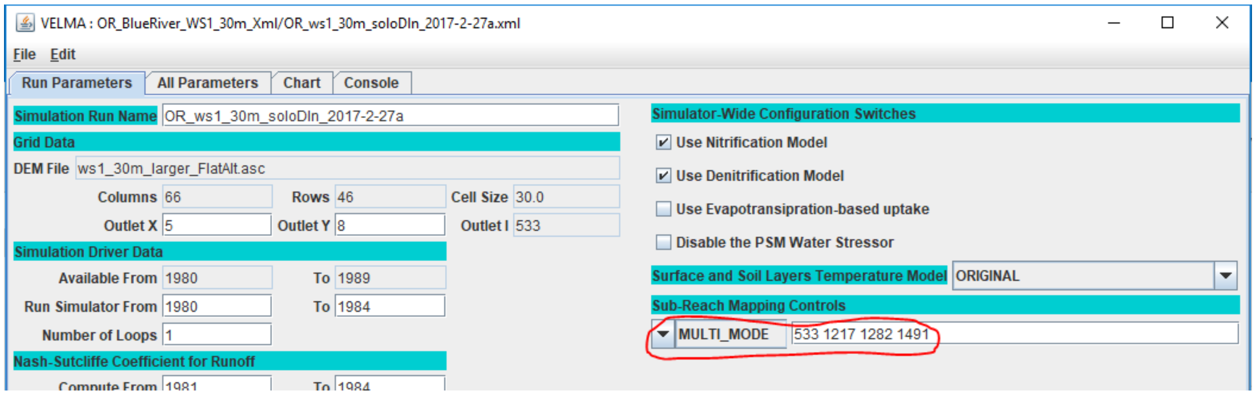
Notes / Comments

Simulator-Wide Configuration Switches
☑ Use Nitrification Model
☑ Use Denitrification Model
☐ Use Evapotranspiration-based uptake
☐ Disable the PSM Water Stressor

Surface and Soil Layers Temperature Model: ORIGINAL

Sub-Reach Mapping Controls
SOLO_MODE 533

Adding the sub-reach indices:


VELMA : OR_BlueRiver_WS1_30m_Xml/OR_ws1_30m_soloDln_2017-2-27a.xml
File
Edit
Run Parameters
All Parameters
Chart
Console

Simulation Run Name: OR_ws1_30m_soloDln_2017-2-27a

Grid Data
DEM File: ws1_30m_larger_FlatAlt.asc
Columns: 66 Rows: 46 Cell Size: 30.0
Outlet X: 5 Outlet Y: 8 Outlet I: 533

Simulation Driver Data
Available From: 1980 To: 1989
Run Simulator From: 1980 To: 1984
Number of Loops: 1

Nash-Sutcliffe Coefficient for Runoff
Compute From: 1981 To: 1984

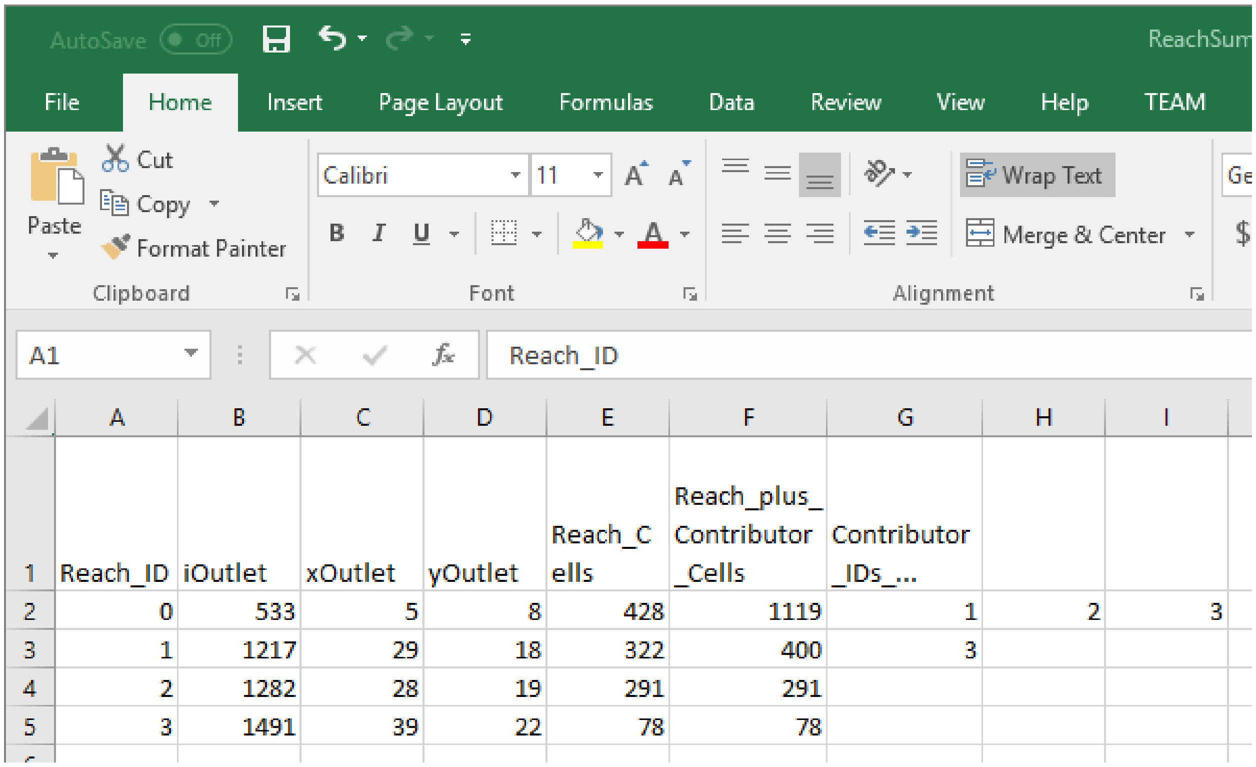
Simulator-Wide Configuration Switches
☑ Use Nitrification Model
☑ Use Denitrification Model
☐ Use Evapotranspiration-based uptake
☐ Disable the PSM Water Stressor

Surface and Soil Layers Temperature Model: ORIGINAL

Sub-Reach Mapping Controls
MULTI_MODE 533 1217 1282 1491

When this simulation configuration is started, the VELMA simulation engine notes the MULTI_MODE setting and the list of outlets, and reports runoff for each of the specified sub-reaches. It also writes an .asc file mapping the cells of the overall watershed to their specific reach-id values (which are listed in the ReachSummary.csv file).

After running the above simulation via JVelma, here is the ReachSummary.csv file’s contents, opened using Excel:



Reach_ID

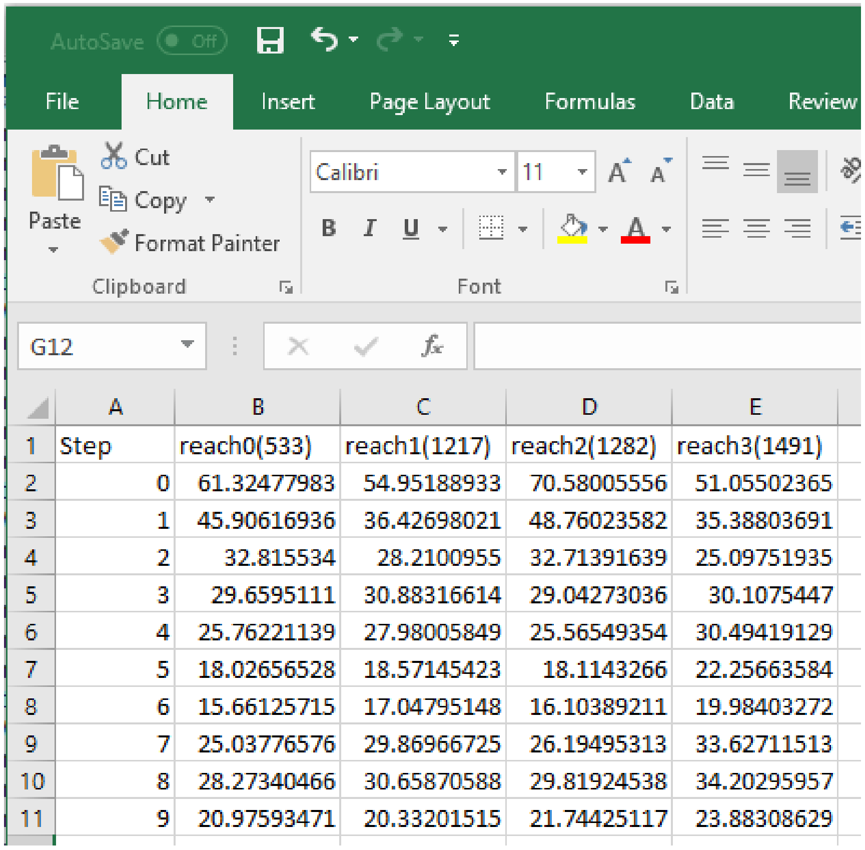
	A	B	C	D	E	F	G	H	I
1	Reach_ID	iOutlet	xOutlet	yOutlet	Reach_Cells	Reach_plus_Contributor_Cells	Contributor_IDs_...		
2	0	533	5	8	428	1119	1	2	3
3	1	1217	29	18	322	400	3		
4	2	1282	28	19	291	291			
5	3	1491	39	22	78	78			

The ReachSummary.csv contains 1 row for each sub-reach, and lists the linear index and X,Y coordinates of the each reach's outlet, plus the number of cells in the reach, and a list of the other reaches that flow into it.

The daily runoff values for each reach are listed in the ReachFlowResults.csv and ReachFlowContributions.csv files.

(continued)

Here are the first 10 simulation days from the ReachFlowResults.csv file:



	A	B	C	D	E
1	Step	reach0(533)	reach1(1217)	reach2(1282)	reach3(1491)
2	0	61.32477983	54.95188933	70.58005556	51.05502365
3	1	45.90616936	36.42698021	48.76023582	35.38803691
4	2	32.815534	28.2100955	32.71391639	25.09751935
5	3	29.6595111	30.88316614	29.04273036	30.1075447
6	4	25.76221139	27.98005849	25.56549354	30.49419129
7	5	18.02656528	18.57145423	18.1143266	22.25663584
8	6	15.66125715	17.04795148	16.10389211	19.98403272
9	7	25.03776576	29.86966725	26.19495313	33.62711513
10	8	28.27340466	30.65870588	29.81924538	34.20295957
11	9	20.97593471	20.33201515	21.74425117	23.88308629

AN IMPORTANT CONSIDERATION:
The contents of the ReachFlowResults.csv and ReachFlowContributions.csv files depend upon how you run the VELMA simulation.

When you use JVelma or VelmaSimulatorCmdLine to run your simulation, the contents of the ReachFlowResults.csv and ReachFlowContributions.csv files will show the same values.
These values are the runoff values at each specified sub-reach outlet INCLUDING runoff from upstream reaches. The runoff values are in mm/day/number-of-cells -- where the number-of-cells is the cell count for the sub-reach PLUS the cell counts of upstream reaches.

When you use VelmaParallelCmdLine to run your simulation, ReachFlowResults.csv continues to report the runoff values as described above, but ReachFlowContributions.csv contents change: they now represent the runoff for ONLY the sub-reach area, in mm/day/number-of-cells -- and now, the number-of-cells value is the cell count for ONLY the sub-reach.

Additionally, VelmaParalleCmLine reports each sub-reach's results in a separate subdirectory under the main output directory. There is one "./Results_* subdirectory for each specified sub-reach outlet (including the primary watershed outlet), and each ./Results_* subdirectory contains a ReachFlowResults.csv and ReachFlowContributions.csv file specific to that sub-reach.

B.10 | Flow Path Modification with the JPDEM Channel Cutter

Overview *(Tutorial B.10 – JPDEM Channel Cutter)*

Digital elevation models (DEMs) need to be flat-processed using the JPDEM tool before VELMA can accurately simulate flow paths within a watershed. However, JPDEM's flat-processing procedure cannot account for obscuring landscape features such as bridges or buried streams, leading to potentially large errors in estimating actual flow paths.

Therefore, JPDEM includes a procedure that allows users to make elevation adjustments (a.k.a. "dredging" or "cutting") to reroute specified channels within a DEM. Performing channel cutting prior to flat-processing a DEM can improve the fidelity of flow paths in the subsequent flat-processed map relative to the actual terrain it represents.

JPDEM's "channel cutter" can also be used to re-engineer existing DEM flow paths, for example, to support "what if" green infrastructure (GI) and low impact development (LID) simulations. Or, VELMA users might want to explore how rerouting stormwater from established gray infrastructure flow paths (e.g., streets equipped with storm drains) to proposed rain gardens, detention ponds, engineered wetlands, etc.

This document describes the basics of using JPDEM's channel cutter to accomplish such objectives.

The Channel Cutter mechanism reduces the elevations of specified sequences of cells ("channel cells") relative to their adjacent cells ("bank cells"). The elevation reduction is performed so that the channel cells guaranty a continuous "step down" from the upper most channel cell ("headwater cell") to the edge of the DEM map grid.

The Channel Cutter mechanism requires a channel guide map that defines the channel cells it should adjust, including a built-in coded sequence representing the order of adjustment to be made on the DEM map data. Producing the channel guide map comprises most of the effort for the overall task of modifying ("dredging") a map's channels (flow paths). Appendix A????? provides instructions for using ArcMap and associated utility tools for establishing a channel guide map and performing the dredging process.

> ***Important Note:*** *JPDEM's "Standard" and "Alternate" flat-processing methods should <u>NOT</u> be used for flat-processing DEMs when the Channel Cutter mechanism is being used. Reason is these two methods both raise and lower cells during flat-processing. The other methods ("Experimental" and "Enhanced Experimental") only raise cell elevations during flat-processing adjustments. Since the Channel Cutter mechanism enforces flow to the edge of the map, these cells are guaranteed correct, therefore are not modified when using the "Experimental" and "Enhanced Experimental"*

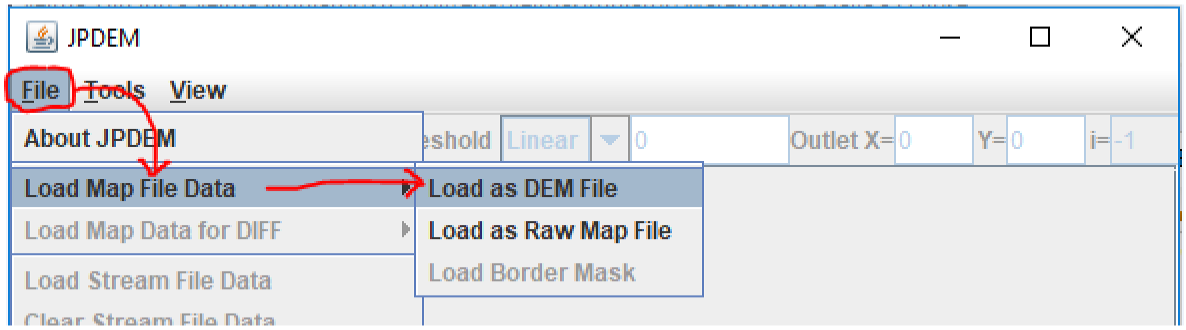
methods. That behavior cannot be guaranteed when using the “Standard” and “Alternate” flat-processing methods.

Using the Channel Cutter

This document assumes that you already have a raw DEM (i.e. a DEM that has *not* been flat-processed) that you wish to dredge (i.e. adjust channel cell elevations for) and a correctly-formatted stream channel sequential map to guide the channel cutter.

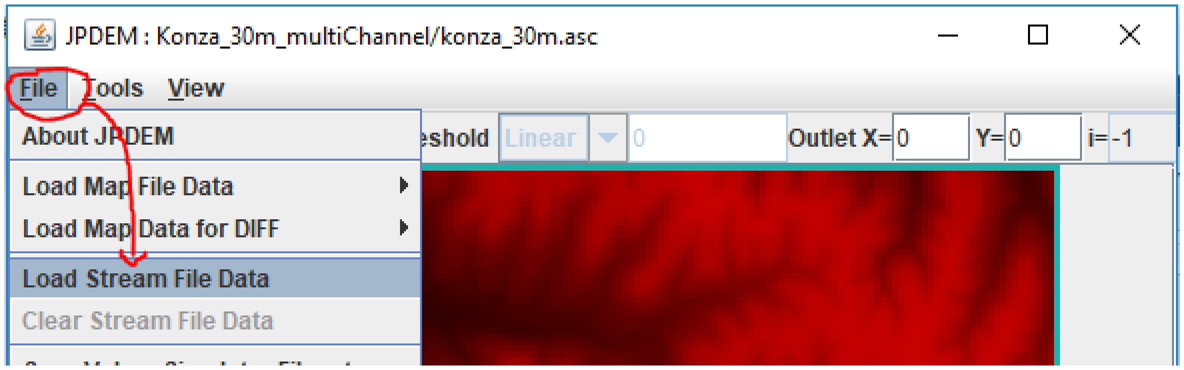
Step 1: Load the DEM Map

Start JPDEM and load the un-flat-processed DEM file, via the File -> “Load Map File Data” -> “Load as DEM File” menu item:

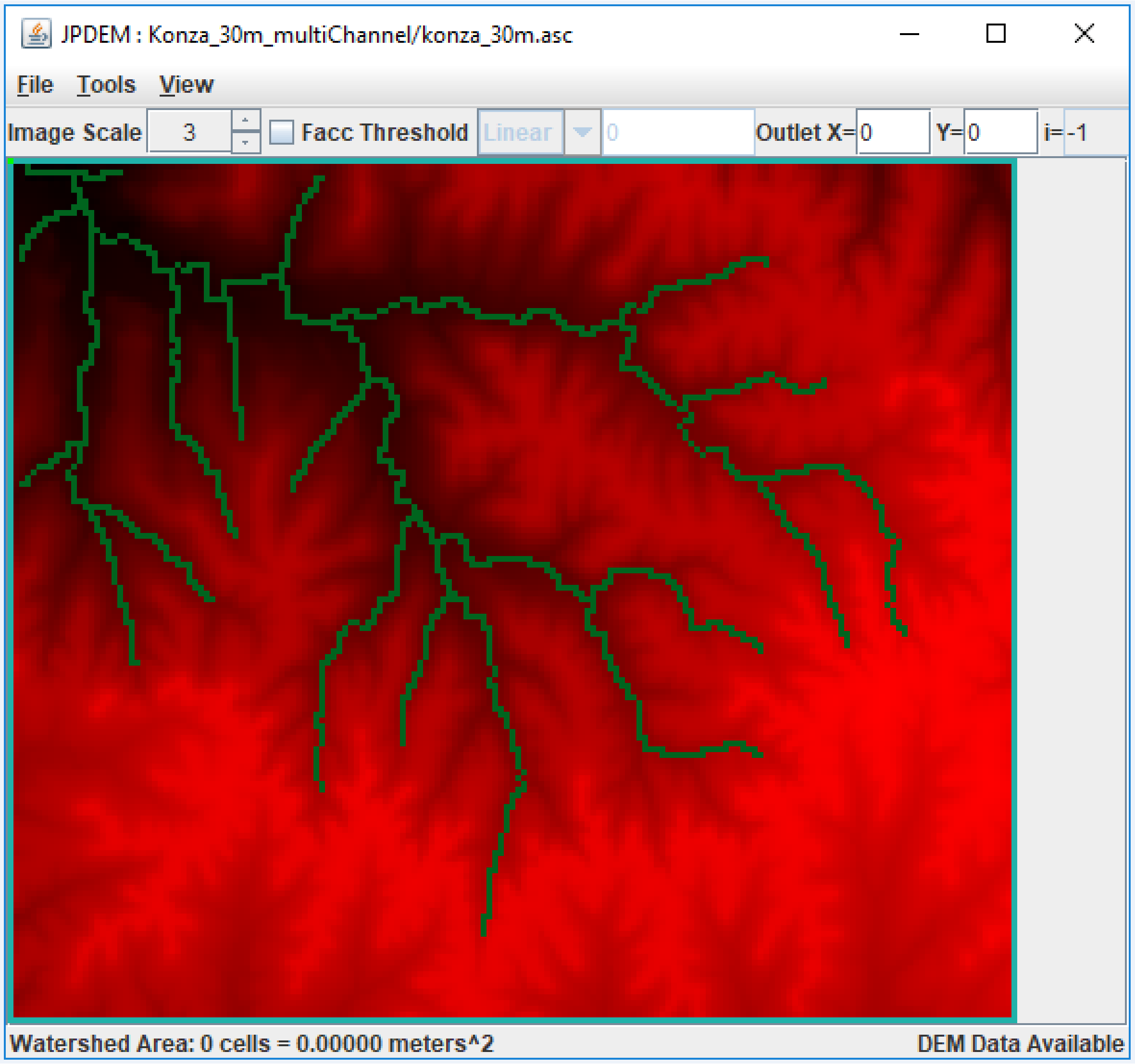


Step 2: Load the Channel/Stream Guide Map

Load the sequential guide map via the File -> “Load Stream File Data”



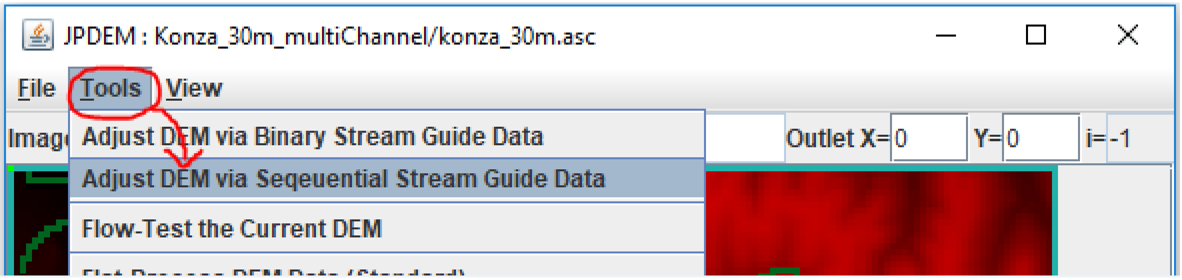
After the sequential guide (a.k.a. Stream File Data) map has been loaded, JPDEM will display the cell from the guide map in green:



If you find that you have loaded the wrong channel/stream guide data, click the File -> "Clear Stream File Data", and then "File -> "Load Stream File Data" to load the correct file.

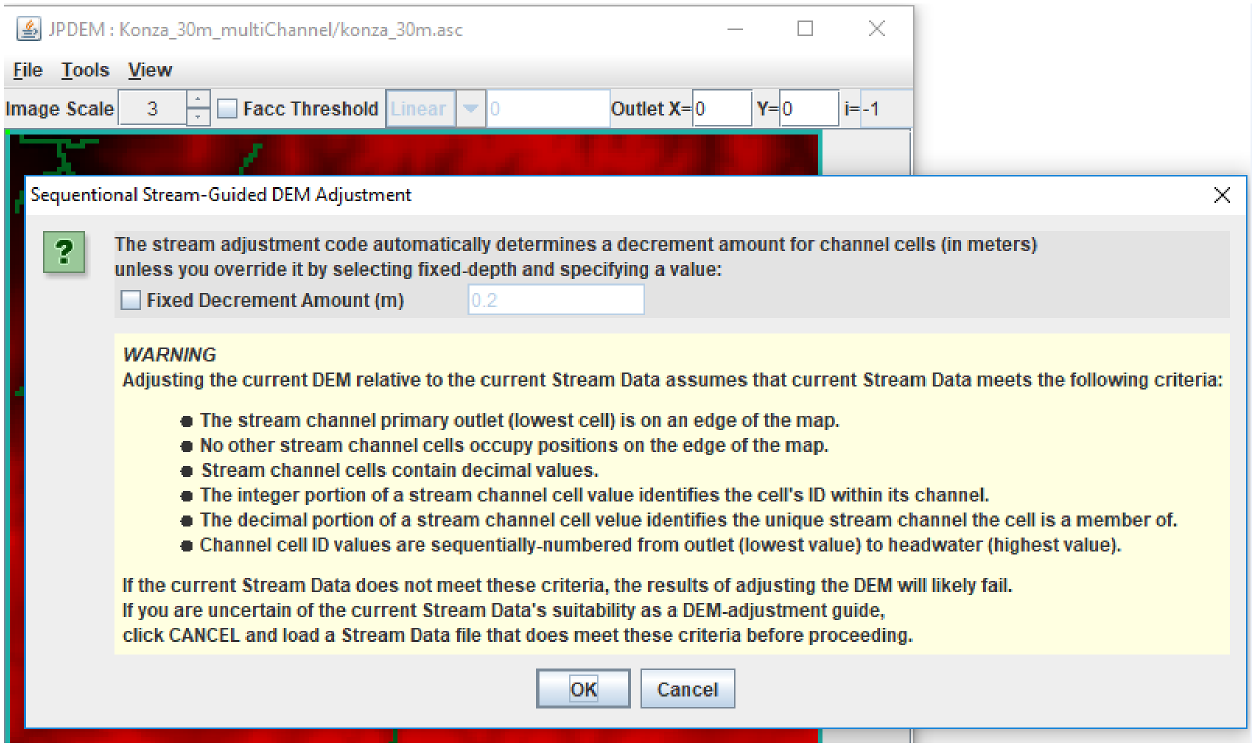
Step 3: Run the Channel Cutter

Perform the actual channel cutting by clicking the Tools -> "Adjust DEM via Sequential Stream Guide Data" menu item.


JPDEM : Konza_30m_multiChannel/konza_30m.asc
File Tools View
Adjust DEM via Binary Stream Guide Data
Adjust DEM via Seqeuential Stream Guide Data
Flow-Test the Current DEM
Outlet X= 0 Y= 0 i= -1

(The Tools -> “Adjust DEM via Binary Stream Guide Data” – is a different, mechanism that is still under development. If you click it by mistake, close JPDEM, then restart it and begin again with Step 1.)

When you click “Adjust DEM via Sequential Stream Guide Data”, the following dialog opens:



Sequential Stream-Guided DEM Adjustment

The stream adjustment code automatically determines a decrement amount for channel cells (in meters) unless you override it by selecting fixed-depth and specifying a value:

Fixed Decrement Amount (m) 0.2

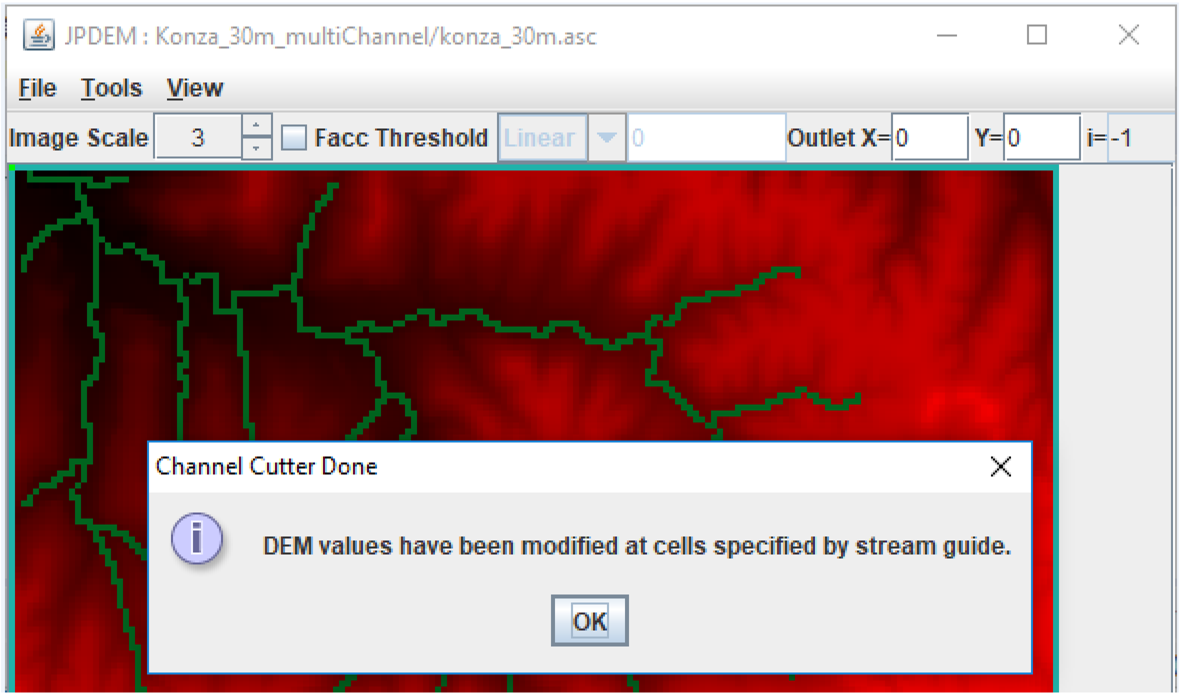
WARNING
Adjusting the current DEM relative to the current Stream Data assumes that current Stream Data meets the following criteria:

- The stream channel primary outlet (lowest cell) is on an edge of the map.
- No other stream channel cells occupy positions on the edge of the map.
- Stream channel cells contain decimal values.
- The integer portion of a stream channel cell value identifies the cell's ID within its channel.
- The decimal portion of a stream channel cell velue identifies the unique stream channel the cell is a member of.
- Channel cell ID values are sequentially-numbered from outlet (lowest value) to headwater (highest value).

If the current Stream Data does not meet these criteria, the results of adjusting the DEM will likely fail.
If you are uncertain of the current Stream Data's suitability as a DEM-adjustment guide,
click CANCEL and load a Stream Data file that does meet these criteria before proceeding.

The “WARNING” section provides a condensed overview of the guide map format specified in Appendix A of this document.
The “Decrement Amount” mentioned at the top of the dialog is the amount that channel cells are lowered relative to their adjacent bank cells. The value is in meters, and is automatically calculated unless you click the “Fixed Decrement Amount” and specify a value.
Unless you need a very large difference between channel and bank elevations, leave the Fixed Decrement option unchecked, and simply click the OK button at the bottom of the dialog.
Doing so will start the channel cutter mechanism.

The following message indicates that the channel cutter has completed its work:



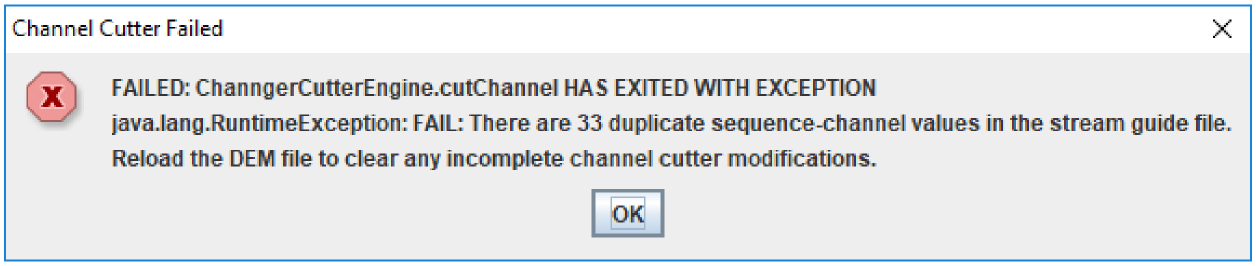
Channel Cutter Done

DEM values have been modified at cells specified by stream guide.

OK

Click the OK button to dismiss it.

When the channel cutter encounters an error, it terminates with the following type of message dialog:


Channel Cutter Failed

FAILED: ChanngerCutterEngine.cutChannel HAS EXITED WITH EXCEPTION
java.lang.RuntimeException: FAIL: There are 33 duplicate sequence-channel values in the stream guide file.
Reload the DEM file to clear any incomplete channel cutter modifications.

OK

Step 4: Check the Channel Cutter Runtime Messages

As it processes the channels, the cutter echoes a status report for each channel ID it encounters to stdout of the console JPDEM was started from. (Note, if you started JDPEM by double-clicking its application icon, the status report messages are suppressed.)

Here is an extract from channel cutter output for a single channel's status (the `[...]` indicate long text lines that were truncated for this document.)

```
Sequenced Channel #14 : 10024 9849 9674 9675 9500 9501 9326 9151 9152 8977 8802 [...]
minDecrement=2.000000000066393E-4 bankDecrement=1.026900000000012 maxDecrement=[...]
          Channel Cells: 10024 9849 9674 9675 9500 9501 9326 9151 9152 8977 [...]
Channel Cell Elevations: 369.2801 368.8617 368.44329999999997 368.0249 [...]
  Problem Channel Cells: -none-
             Bank Cells: 10025 10200 10199 10198 10023 9848 9850 9850 10025 10023 [...]
   Bank Cell Elevations: 372.6165 376.9927 373.6152 375.6302 375.5685 375.7058 [...]
     Problem Bank Cells: -none-
```

Note that the above channel processed cleanly: the "Problem Channel" and "Problem Bank" lines report "-none-". If either of the "Problem" report lines listed cell indices, that would indicate the channel cutter encountered some ambiguity at those cells. Careful review of the guide map data must then be performed: it may be that the sequencing of the channel is incorrect.

The cutter will fail – and lists to the stdout console – when it encounters one or more duplicate sequence.channel ID values in a single guide map. Here is short example of what the duplicate error message looks like:

```
At 2018-10-05 14:59:02: Starting Sequential Stream Guide Processor ...
Cutter depth: Automatic Detection
Duplicate: iSequence=15  iChannel=9
Duplicate: iSequence=2  iChannel=2
Duplicate: iSequence=141  iChannel=1
Duplicate: iSequence=197  iChannel=1
FAILED: ChanngerCutterEngine.cutChannel HAS EXITED WITH EXCEPTION
java.lang.RuntimeException: FAIL: There are 4 duplicate sequence-channel values in the
stream guide file.
Reload the DEM file to clear any incomplete channel cutter modifications.
```

Step 5: Flat-Process the Channel-Cut Map

You can save the flat-processed channel-cut DEM map immediately after running the cutter, and we recommend you do this, but we also recommend you flat-process the DEM immediately after successfully channel-cutting it. The results of the channel-cutter mechanism are most compatible with the 3 "Experimental …" flat-processing algorithms; choose one of those and run it on the just-cut DEM. After the flat-processor is done, save the resulting map to a file (File -> "Save DEM Data As" -> "ASCII Grid"). You may also use the "Facc Threshold" rendering feature to evaluate how well the guide map's cells correspond to high-flow cells of the flat-processed DEM (see Appendix B for details).

Appendix A:

Channel Cutter Sequential Guide Map Format

For some watersheds, a VELMA user may desire to increase the processing time required to flat process their area of interest or may need to control the representation of the VELMA simulation flow path. Both goals can be met using JPDEM’s advanced feature, the Dredger.

The current state of the dredging tool is beyond “Beta”. Though, currently this method is not as user friendly as JPDEM itself. As of VELMA 2.1, the user must use GIS software to prepare spatial data for an intermediate Java-based tool referred to as the Channel Cutter method (a.k.a Dredger), or by Java file referred to as “JpdemStreamDredgePrepper.jar”.

A channel cutter sequential guide map is an ESRI Grid ASCII (.asc) map file with the same dimensions as the DEM file it is intended to overlay. The contents of the file specify a collection of one or more channels, and the sequence of cells within each channel.

The following cell value details are generated for the user by the JpdemStreamDredgePrepper.jar tool. Understanding the end goal assists with trouble shooting the final stream dredge map provided to JPDEM. Each cell data value within the file must be either:

1. The DEM file’s nodata_value (usually -9999).
    a. Cells with the nodata_value are ignored by the channel cutter mechanism.
2. A decimal number signifying that cell’s stream flow sequence and channel ID numbers.
    a. The whole number portion (left side of decimal) of the number is the cell’s stream flow sequence number within the channel. First stream cell (stream mouth) is always 1 and each flowing cell is a greater number than touching downstream number. Numbers in the sequence can be skipped.
    b. The mantissa or cents portion (right side of decimal) of the number is the stream ID value for each channel that cell is a member. The ranking of each stream channel value dictates the hierarchy of dredging. A stream value of 1 has highest priority. Any stream cell of lower priority is dredged first with a higher priority being dredged later. This allows for proper dredging at stream confluences. The stream that terminates at the confluence should be of lower stream value priority, meaning a larger stream number.

The channel cutter mechanism is inelastic and cannot recover from map data errors. In addition to the above, the following requirements must be met:

- It is an error for the same `sequenceID.channelID` number value to occur more than once in a single map file.
- A channel’s sequence ID values must increase (lowest to highest) from its outlet to its “headwater” cell.
- A channel’s outlet cell must either terminate at the edge of the DEM grid, or adjacent to a cell of a different channel.
- A channel (and all connecting channels) may touch the edge of the DEM grid at only one cell location – that is, a channel or network of channels must terminate at a single DEM grid edge

cell. A sequential guide map *may* contain more than one channel that terminates at a DEM grid edge, but every channel or network of channels that terminates at a DEM grid edge cell must be independent (i.e. not connect) to any other channel or network of channels that terminate at a grid edge cell.

Preparing the channel cutter sequential guide map using the JPDEM Dredger Prepper Tool

Part 1: Preparation of stream data

1. Load into ArcMap your raw DEM in the format VELMA will receive, meaning the extent and final cell resolution are already resolved. Note: the extent is an important here, if you change the DEM extent, you must create a new Dredge map matching the new DEM.
2. Load the polyline shapefile representing your desired dredging pattern.
    a. These data can be existing NHD data or other prepared stream data representations, created from scratch when data does not exist, such as when proposed future flow paths are part of the research.
    b. Two key points on this polyline shapefile.
        i. Each "stream" polyline must be represented as one, meaning in the attribute table an entire stream must contain the same number.
        ii. A "stream" in this polyline shapefile is all the line segments together, and these line segments must all be connected to one network.
        iii. Each "stream" polyline must have its own unique number. Again, a "stream" can be made up of many line segments, but all the segments representing one "stream" must all have the same number, and that number must be unique to only that "stream".
        iv. The "stream" network must have one, and only one, unique "stream" touching the edge of the map.
        v. The numbers themselves DO NOT have to be in sequence. That is one of several tasks the "JpdemStreamDredgePrepper" accomplishes during initiation.
    c. Clip the "stream" polyline network to match the extent of the raw DEM file.
    d. Check the attribute table to ensure all line segments have the unique ID matching the "stream" that line segment belongs to.
        i. Edit the attribute table as needed to meet this tool requirement. Line segments with missing data will become the terminal end of that "stream" due to the discontinuity. Therefore, the JpdemStreamDredgePrepper tool will stop processing that stream at any point of discontinuity.

Convert the "stream" polyline to a raster with matching extent and matching cell resolution. If using ArcMaps Polyline to Raster conversion tool, be sure the attribute field used is the field representing your "streams" unique stream numbers.

Running the JpdemStreamDredgePrepper

JpdemStreamDredgePrepper is a tool that can be used to simplify the creation of the JPDEM Stream Guide input map. The tool is ran from command line as of 23rd May, 2017. On-going work is being done to streamline this process even further.

Part 2: Running JpdemStreamDredgePrepper Java Tool

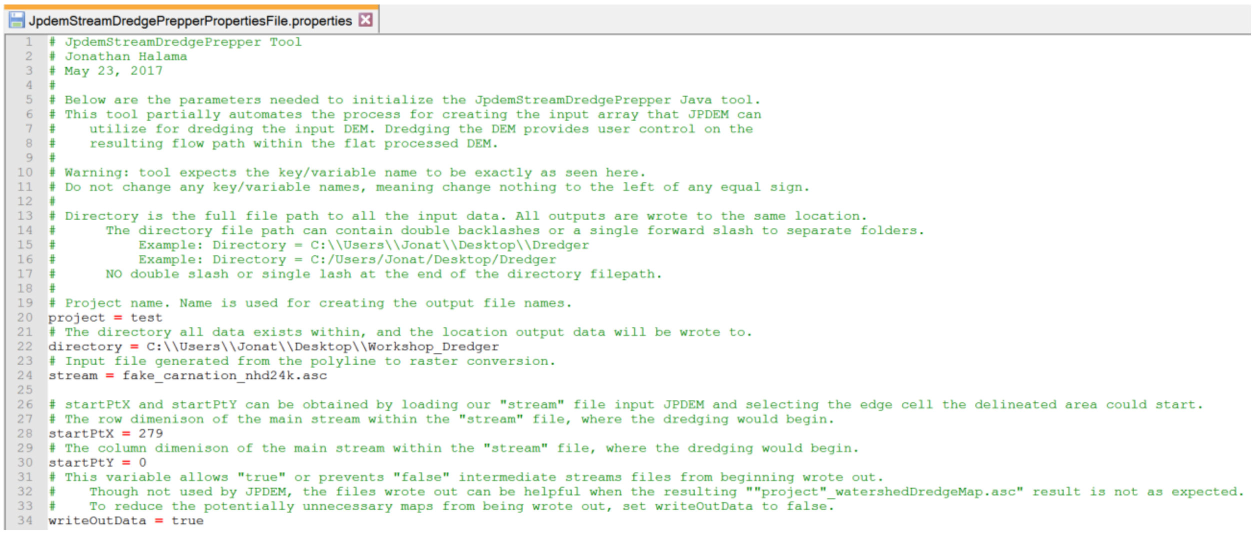
Steps for preparing to run the JpdemStreamDredgePrepper tool

1. Fill out the properties file using the JpdemStreamDredgePrepper_DefaultProperties.properties file provided.

   Directory and Burn Modification keys.

   a. Provide a project name. The ProjectName must be filled in when the runForModifiedBurns is set to true. The output modified burn file will be named using this key.
   b. Provide the full file path directory to the location the properties file and input data will be located at.

      *** The directory file path can contain double backlashes or a single forward slash to separate folders.
   c. stream key is the array generated from the Polyline to Raster conversion. Must be an ASCII. Use ArcMap Raster to ASCII tool to accomplish file conversion.
   d. startPtX key is the row dimension of the main stream within the "stream" file, where the dredging would begin against the edge of the area of interest.
   e. startPtY key is the column dimension of the main stream within the "stream" file, where the dredging would begin against the edge of the area of interest.
   f. writeOutData key is used to control the writing out of intermediate files that are not needed by JPDEM, but could be useful for trouble-shooting when the output ""project"_watershedDredgeMap.asc" file is not as expected.

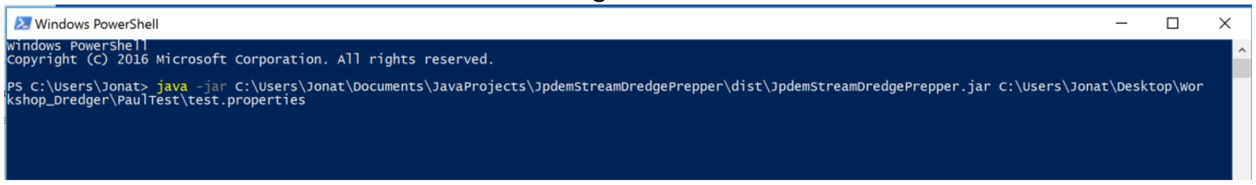


JpdemStreamDredgePrepperPropertiesFile.properties

```
# JpdemStreamDredgePrepper Tool
# Jonathan Halama
# May 23, 2017
#
# Below are the parameters needed to initialize the JpdemStreamDredgePrepper Java tool.
# This tool partially automates the process for creating the input array that JPDEM can
#    utilize for dredging the input DEM. Dredging the DEM provides user control on the
#    resulting flow path within the flat processed DEM.
#
# Warning: tool expects the key/variable name to be exactly as seen here.
# Do not change any key/variable names, meaning change nothing to the left of any equal sign.
#
# Directory is the full file path to all the input data. All outputs are wrote to the same location.
#      The directory file path can contain double backlashes or a single forward slash to separate folders.
#          Example: Directory = C:\\Users\\Jonat\\Desktop\\Dredger
#          Example: Directory = C:/Users/Jonat/Desktop/Dredger
#      NO double slash or single lash at the end of the directory filepath.
#
# Project name. Name is used for creating the output file names.
project = test
# The directory all data exists within, and the location output data will be wrote to.
directory = C:\\Users\\Jonat\\Desktop\\Workshop_Dredger
# Input file generated from the polyline to raster conversion.
stream = fake_carnation_nhd24k.asc

# startPtX and startPtY can be obtained by loading our "stream" file input JPDEM and selecting the edge cell the delineated area could start.
# The row dimenison of the main stream within the "stream" file, where the dredging would begin.
startPtX = 279
# The column dimenison of the main stream within the "stream" file, where the dredging would begin.
startPtY = 0
# This variable allows "true" or prevents "false" intermediate streams files from beginning wrote out.
#    Though not used by JPDEM, the files wrote out can be helpful when the resulting ""project"_watershedDredgeMap.asc" result is not as expected.
#    To reduce the potentially unnecessary maps from being wrote out, set writeOutData to false.
writeOutData = true
```

2. Save the Properties file within an accessible folder.
3. Launch PowerShell. Other command line methods will work, but PowerShell has the invaluable feature of drag-n-dropping full filepaths.
4. Type in PowerShell the following: java -jar
   a. There is a space between “java” and “-jar”, as well as a space after -jar

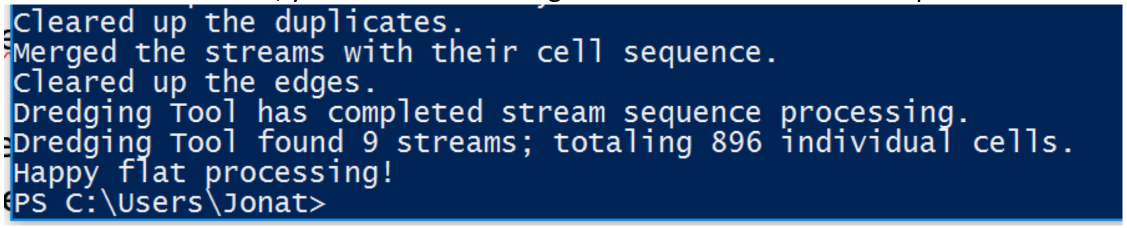
5. Drag-n-drop the "JpdemStreamDredgePrepper.jar" from its full file path location.
6. Add a space by clicking the spacebar.
7. Drag-n-drop the "JpdemStreamDredgePrepper_DefaultProperties.properties" from its full file path location.
   a. Properties file filename can vary, if you renamed it after modifying the variables to fit your study area.
   b. PowerShell should look something like this now:



Windows PowerShell

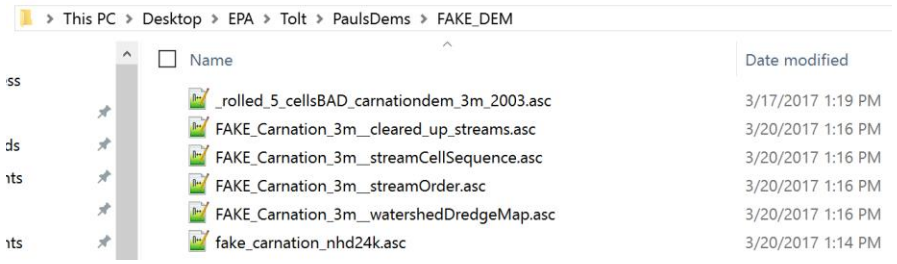
```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Jonat> java -jar C:\Users\Jonat\Documents\JavaProjects\JpdemStreamDredgePrepper\dist\JpdemStreamDredgePrepper.jar C:\Users\Jonat\Desktop\Wor
kshop_Dredger\PaulTest\test.properties
```

8. Hit enter! If successful, you will see something like this when the tool is complete:



```
Cleared up the duplicates.
Merged the streams with their cell sequence.
Cleared up the edges.
Dredging Tool has completed stream sequence processing.
Dredging Tool found 9 streams; totaling 896 individual cells.
Happy flat processing!
PS C:\Users\Jonat>
```

3. Outputs:
   a. The tool generates several output files. This is partly due to the tool not being 100% complete, so some files are used during development. The other reason is these files can be useful for trouble shooting problems when the final map does not match what you expected.



> This PC > Desktop > EPA > Tolt > PaulsDems > FAKE_DEM

Name	Date modified
_rolled_5_cellsBAD_carnationdem_3m_2003.asc	3/17/2017 1:19 PM
FAKE_Carnation_3m__cleared_up_streams.asc	3/20/2017 1:16 PM
FAKE_Carnation_3m__streamCellSequence.asc	3/20/2017 1:16 PM
FAKE_Carnation_3m__streamOrder.asc	3/20/2017 1:16 PM
FAKE_Carnation_3m__watershedDredgeMap.asc	3/20/2017 1:16 PM
fake_carnation_nhd24k.asc	3/20/2017 1:14 PM

   b. In the above file structure example:
      1. *projectName*_watershedDredgeMap.asc
         a. This is the file one provides to JPDEM. This file is a sorted stream network with the file structure of "sequentialCell.uniqueStream". Example: "1.1" "2.1" "3.1"
      2. *projectName*_cleared_up_streams.asc
         a. The results of an intermediate processing step. Normally only used for code debugging.
      3. *projectName*_streamOrder.asc
         a. The results from the tool sorting the "streams" per cell for the numbers to the right of the decimal.
      4. *projectName*_streamCellSequence.asc

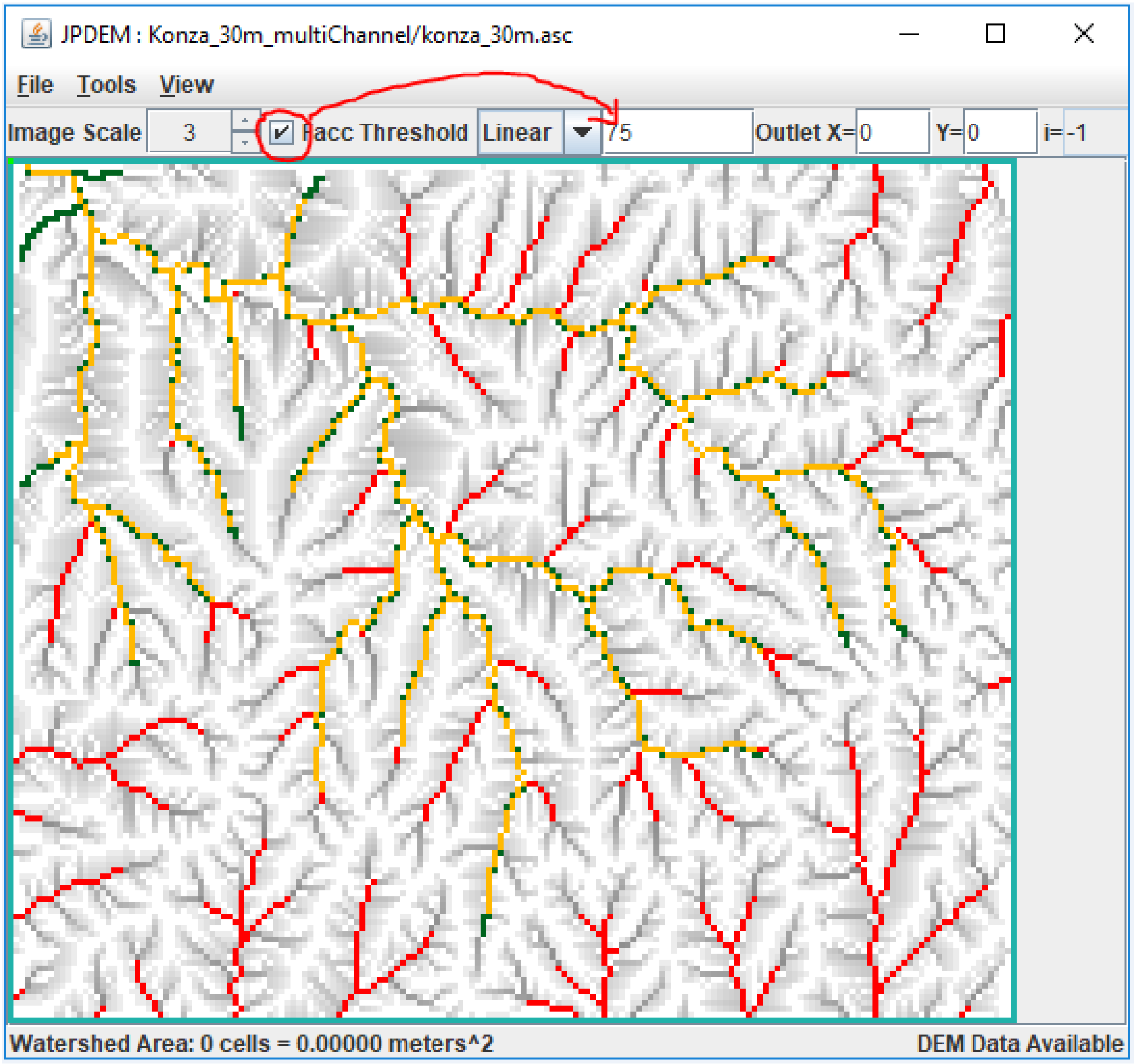
The results from the tool sorting the values per cell for the numbers to the left of the decimal.

Appendix B:

Viewing Channel Information After Flat-Processing

Assuming you have loaded a valid DEM and associated guide map, successfully run the channel cutter, and flat-process the channel-cut map, you can view how well the channel guide cells correspond to the DEM’s high-flow cells.
Click the “Facc Threshold” checkbox in JPDEM’s toolbar, and enter an appropriate threshold value.
The threshold image rendering is “guide-map aware”, and when guide map information is available, the Facc Threshold colors the DEM map’s cells as follows:

Cell Color	**>= Facc Threshold ?**	**Channel Guide Cell ?**
Grayscale	NO	NO
Red	YES	NO
Dark Green	NO	YES
Yellow/Orange	YES	YES



JPDEM : Konza_30m_multiChannel/konza_30m.asc

Image Scale 3 [✓] Facc Threshold Linear 75 Outlet X= 0 Y= 0 i= -1

Watershed Area: 0 cells = 0.00000 meters^2

DEM Data Available

C.1 | VELMA Reach Mapping

Overview (*Tutorial C.1 – VELMA Reach Mapping)*

Tutorial "*B.9_HowTo_JPDEM Sub-Reach Delineations*" described how to use JPDEM to establish a set of x, y coordinates for user-delineated sub-watershed (sub-reach) outlets, in preparation for running VELMA in parallel mode to simulate daily streamflow for those sub-reach outlets.

The present tutorial, "*Tutorial C.1 – VELMA Reach Mapping*", describes how to incorporate the x, y sub-reach coordinates developed with JPDEM into VELMA.

Tutorial "*Tutorial C.2 – How To Run VELMA Parallel Mode*" describes how to run VELMA in parallel mode for generating daily streamflow results for the specified sub-watershed (sub-reach) outlets.

Aside from potential gains in computational efficiency, sub-reach delineations for running VELMA in parallel mode can help watershed managers tease apart streamflow responses to local and more distant upstream changes in land cover and land use.

The VELMA simulator engine can be configured to subdivide the watershed specified by the primary outlet (via the `outx` and `outy` configuration parameters) into several sub-reaches. Ordinarily, this feature is unnecessary, but it is useful and/or required for certain simulation situations. For example, sub-reaches must be specified when running a simulation via the `VelmaParallelCmdLine. For additional details see "How to: Run VELMA in Parallel Mode`. [Note: Sub-reaches must also be specified in simulation configuration .xml files that will be used for multi-scale simulation runs via the `VelmaSimulatorAltCmdLine`. Multi-scale simulations are still under development].

The simulation configuration's sub-reach behavior and the locations of the sub-reaches are specified by two, separate, but interrelated parameters:

- The `enableReachMapping` parameter specifies the mode (i.e. behavior) of the VELMA simulator with regard to sub-reach maps.
  Its value must be the *exact* text representation of one of the four valid modes: AUTO_MODE, LEGACY_MODE, MULTI_MODE or SOLO_MODE.

- The `initialReachOutlets` parameter specifies the linear cell indices of all of the sub-reaches. When specified, its value must be a whitespace-separated sequence of
  (The `initialReachOutlets` parameter value is ignored for some `enableReachMapping` modes.)

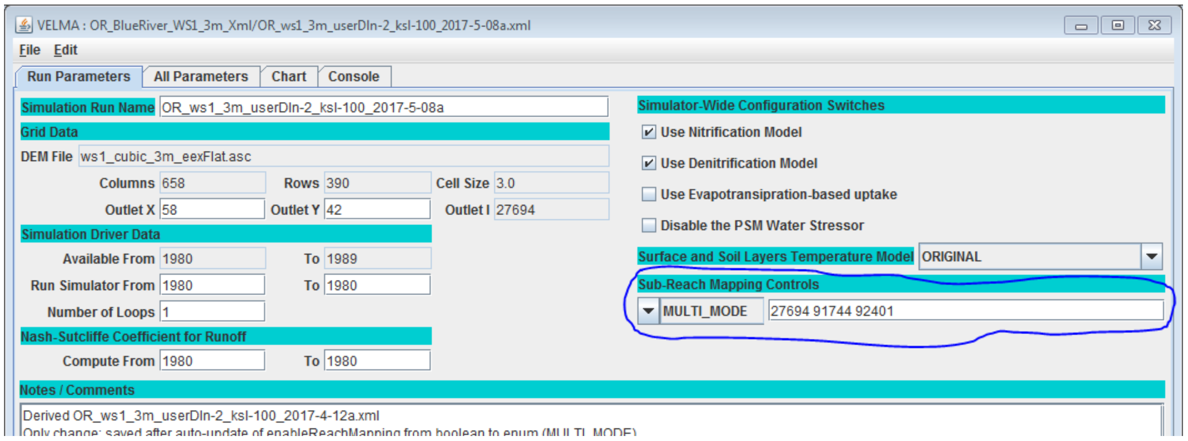
Specify the `enableReachMapping` mode first, then (depending upon whether the specified mode requires it or not) specify the `initialReachOutlets` parameter's list of sub-reach outlets.

enableReachMapping value is …	initialReachOutlets value is …	VELNA Simulator behavior is …	Notes
AUTO_MODE	Ignored	Auto-generated sub-reaches.	Seldom required; may generate many sub-reaches
LEGACY_MODE	Ignored	No sub-outlets, single watershed.	This mode is deprecated.
MULTI_MODE	Required	Sub-outlets specified by `initialReachOutlets`	Use for Velma Parallel or Alt cmdline runs.
SOLO_MODE	Ignored	No sub-outlets, single watershed.	Most common mode for JVelma runs.

Setting the Reach Mapping Parameters in JVelma GUI

You can set both of the reach mapping parameters in either JVelma's "Run Parameters" or from the "All Parameters" tab.

In the "Run Parameters" tab, underneath the label "Sub-Reach Mapping Controls" there is a drop-down selector for the `enableReachMapping` mode value, and a text field for the `initialReachOutlets`:


VELMA : OR_BlueRiver_WS1_3m_Xml/OR_ws1_3m_userDIn-2_ksl-100_2017-5-08a.xml
File
Edit
Run Parameters
All Parameters
Chart
Console

Simulation Run Name: OR_ws1_3m_userDIn-2_ksl-100_2017-5-08a

Grid Data
DEM File: ws1_cubic_3m_eexFlat.asc
Columns: 658 Rows: 390 Cell Size: 3.0
Outlet X: 58 Outlet Y: 42 Outlet I: 27694

Simulation Driver Data
Available From: 1980 To: 1989
Run Simulator From: 1980 To: 1980
Number of Loops: 1

Nash-Sutcliffe Coefficient for Runoff
Compute From: 1980 To: 1980

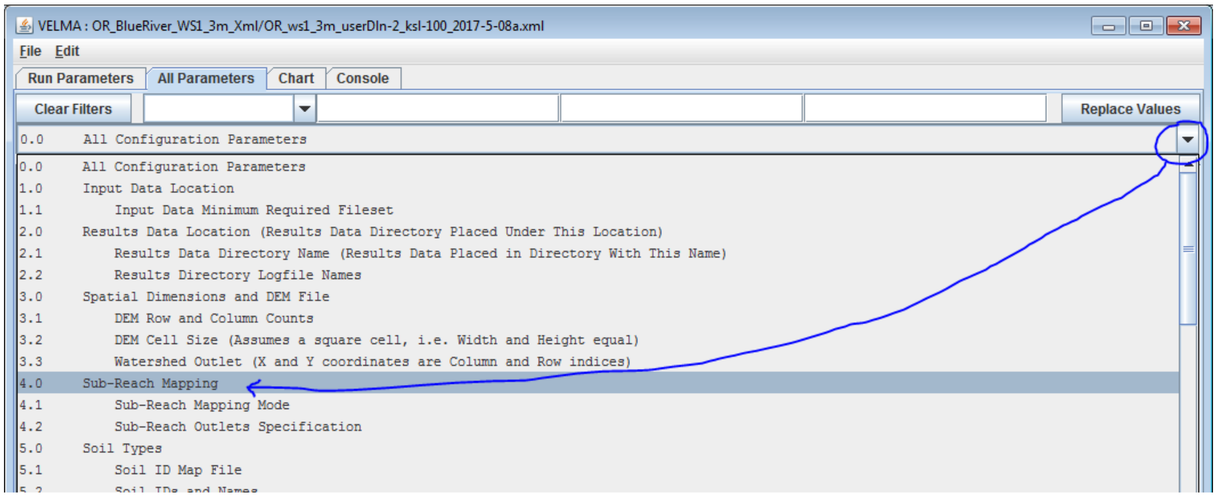
Simulator-Wide Configuration Switches
[✓] Use Nitrification Model
[✓] Use Denitrification Model
[ ] Use Evapotranspiration-based uptake
[ ] Disable the PSM Water Stressor
Surface and Soil Layers Temperature Model: ORIGINAL

Sub-Reach Mapping Controls
MULTI_MODE 27694 91744 92401

Notes / Comments
Derived OR_ws1_3m_userDIn-2_ksl-100_2017-4-12a.xml
Only change: saved after auto-update of enableReachMapping from boolean to enum (MULTI_MODE)

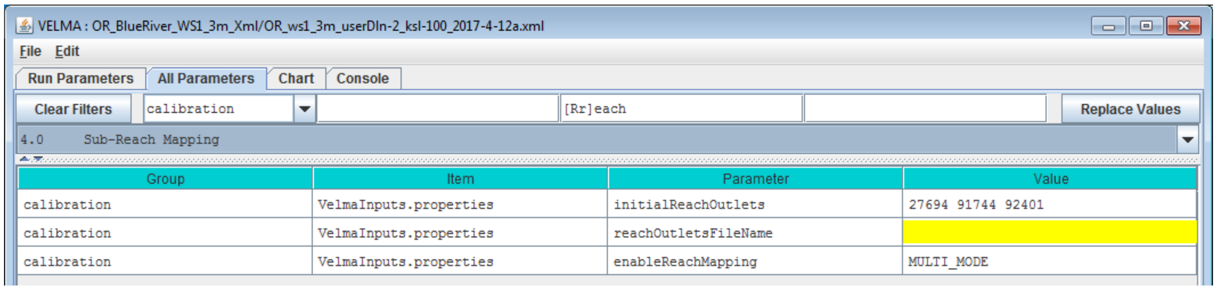
In the image above the `enableReachMapping` parameter is set to `MULTI_MODE`, and the `initialReachOutlets` value is a 3-index sequence of cell indices: `27694 91744 92401`.

In the “All Parameters” tab, you can quickly filter-limit the table to the reach mapping parameters by selecting the “Sub-Reach Mapping” topic from the drop-down outline menu as show below:



```
0.0    All Configuration Parameters
0.0    All Configuration Parameters
1.0    Input Data Location
1.1        Input Data Minimum Required Fileset
2.0    Results Data Location (Results Data Directory Placed Under This Location)
2.1        Results Data Directory Name (Results Data Placed in Directory With This Name)
2.2        Results Directory Logfile Names
3.0    Spatial Dimensions and DEM File
3.1        DEM Row and Column Counts
3.2        DEM Cell Size (Assumes a square cell, i.e. Width and Height equal)
3.3        Watershed Outlet (X and Y coordinates are Column and Row indices)
4.0    Sub-Reach Mapping
4.1        Sub-Reach Mapping Mode
4.2        Sub-Reach Outlets Specification
5.0    Soil Types
5.1        Soil ID Map File
```

Click-select topic “4.0 Sub-Reach Mapping”, as shown by the blue circle and arrow above, to set the filter for reach mapping, and limit the All Parameters table rows to only reach mapping parameters (as shown in the following image):



Group	Item	Parameter	Value
calibration	VelmaInputs.properties	initialReachOutlets	27694 91744 92401
calibration	VelmaInputs.properties	reachOutletsFileName	🟨
calibration	VelmaInputs.properties	enableReachMapping	MULTI_MODE

Notice the `reachOutletsFileName` parameter and blank (yellow) value in the above image. The `reachOutletsFileName` parameter is a legacy setting that is ignored. If you run JVelma’s Edit -> Remove Unused Parameters function, it will be deleted from the configuration.

Legacy Configurations Will (Mostly) Work Correctly

Earlier VELMA configurations specified Reach Mapping behavior differently.

- `enableReachMapping` was a boolean value (i.e. either `true` or `false` and nothing else).
- The `reachOutletsFileName` contents were combined with the `initialReachOutlets` value to form the final list of reach outlets.

When a legacy configuration .xml is loaded into JVelma or one of the VELMA command-line tools, the following updates occur:

- The `enableReachMapping` value is changed: true = `MULTI_MODE`, false = `LEGACY_MODE`.
- The `reachOutletsFileName` value is ignored.

- The `initialReachOutlets` value is untouched, and is used as-is if `enableReachMapping’s` new mode value is `MULTI_MODE`.

This set of update rules means that most legacy simulations’ Reach Mapping behavior will remain unchanged. The exceptions are configurations that contained `reachOutletsFileName` parameter values, and configurations that had `enableReachMapping` == `true` and `initialReachOutlets` == “”.

The legacy VELMA simulator engine used `enableReachMapping` == true and `initialReachOutlets` == “” as an implicit indication of what is called AUTO_MODE. One of the goals of the new Reach Mapping parameters is making running in AUTO_MODE a very deliberate intention.
Legacy configurations with `enableReachMapping` == true and `initialReachOutlets` == “” now map to MULTI_MODE, but the empty `initialReachOutlets` value is forced to = the primary (`outx`, `outy`) outlet.
This means that the run will effectively be the same as manually setting `SOLO_MODE`.

C.2 | How to Run VELMA in Parallel Mode

Overview *(Tutorial C.2 – How To Run VELMA Parallel Mode)*

This tutorial describes how to run VELMA in parallel mode, which splits a simulation run of a single, large watershed into multiple user-designated sub-watersheds, and runs each separately. When running in parallel mode, VELMA is programmed to collect simulated daily streamflow information for each designated sub-reach (sub-watershed) outlet. VELMA uses that information to compute, for each sub-reach outlet, the sum of flow contributions from

- all upstream grid cells draining to the sub-reach outlet, and
- only those cells falling between the sub-reach outlet and the next upstream sub-reach outlet (thereby excluding flow from all other upstream cells).

Thus, aside from potential gains in computational efficiency, sub-reach / sub-watershed delineations for running VELMA in parallel mode can help watershed managers tease apart streamflow responses to local and more distant upstream changes in land cover and land use.

Introduction

Within certain limitations, VELMA can be configured to split the simulation run of a single, large watershed into several sub-watersheds, and running each separately.

The sub-watersheds processes are scheduled such that downstream, dependent watersheds run after upstream contributing sub-watersheds.

For large (in terms of watershed total cell count) areas, splitting the "primary" watershed into sub-watersheds can reduce running time and memory footprint (per sub-process) by a significant amount.

Also, running a simulation in parallel mode is currently the only way to generate "only-my-own-sub-watershed runoff" data (as opposed to "my-sub-watershed-plus-upstream runoff" data) for a simulation.

Running in parallel mode imposes some special requirements and limitations:

- Parallel mode can only be run from a command-line front-end.
  You cannot launch a parallel mode run from JVelma, and no runtime display is available.

- Parallel mode requires a relatively capable computer.
  If your computer has a single processor with only 1 or 2 cores, or a limited (4 Gigabytes or less) amount of memory, you may not gain any benefit from running in VELMA parallel mode.

- There is some additional configuration effort involved.
  You must determine sub-watershed (a.k.a. "sub-reach") outlets for the primary watershed.

You must tweak the simulation configuration .xml (described later in this document).

- Some Disturbance functionality is unavailable.
  Specifically, Water Drain Disturbances are incompatible with parallel mode runs.

- Parallel Mode runs generate a larger number of results files, some of which may be redundant.

- For small watersheds, the overall running time of a parallel mode run may be greater than running it single-process style (i.e. under JVelma).

- Unlike JVelma or VelmaSimulatorCmdLine, if you want to run a simulation configuration in parallel mode multiple times, you must either manually rename the results directory from the previous run of that simulation configuration, or delete it. The Velma parallel mode tool will not create new “_0”, “_1”, etc. named-suffix results folders automatically.

(continued)

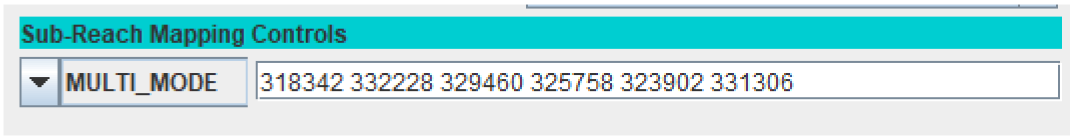
Configuration Steps for A VELMA Parallel Mode Run

Start with a “proven”, single-process configuration – i.e., a simulation configuration .xml file that has the enableReachMapping parameter set to SOLO_MODE, and that has been run successfully (even if only for one simulation year) under JVelma.

Load that simulation configuration .xml file into JVelma, and immediately change its “Simulation Run Name” (i.e. the run_index parameter’s value), then save it. This creates a copy of the original .xml that you can modify for parallel mode, while saving the original .xml for single-process use.

Next:

9. Change the enableReachMapping from SOLO_MODE to MULTI_MODE.
   Note: there is also an AUTO_MODE available, but we do not recommend using it at this time.

10. Enter the linear indices of the sub-reach outlet cells as the whitespace-separated value of the initializeReachOutlets parameter.
    The tutorial “Tutorial B.9 - JPDEM Sub-Reach Delineations” details how to determine sub-reach locations.
    Here is a snapshot of the “Run Parameters” tab’s “Sub-Reach Mapping Controls” section showing an example of configuring for MULTI_MODE.
    
    Sub-Reach Mapping Controls
    MULTI_MODE 318342 332228 329460 325758 323902 331306
    The numbers to the right of the MULTI_MODE setting are the sub-reach indices. Entering them into this numeric field assigns them to the initializeReachOutlets parameter.

11. Spatial Data must be output as "trimmed" maps.
    Set the trimOutputToWatershedBoundary parameter value to true for any and all spatialDataWriter parameter Group(s).
    Set the trimOutputToWatershedBoundary parameter value to true for any and all disturbance parameter Group(s).

12. All cover and soil results must be reported.
    Set the parameter reportAllCoverResults value to true.
    Set the parameter reportAllSoilResults value to true.
    (Note: this is so important that VELMA's parallel mechanism will perform it automatically during simulator initialization if it is not done before-hand.)

Save the changes – the simulation configuration .xml is now ready for use in VELMA Parallel Mode.

Running a Simulation via the VelmaParallelCmdLine

Although you can use JVelma to prepare a simulation configuration for a parallel mode run, you cannot launch the simulation run from JVelma, via its START button. Instead, you must open a Microsoft Powershell command window, and invoke the VelmaParallelCmdLine tool.
(See Appendix I for how to start a Microsoft Powershell.)

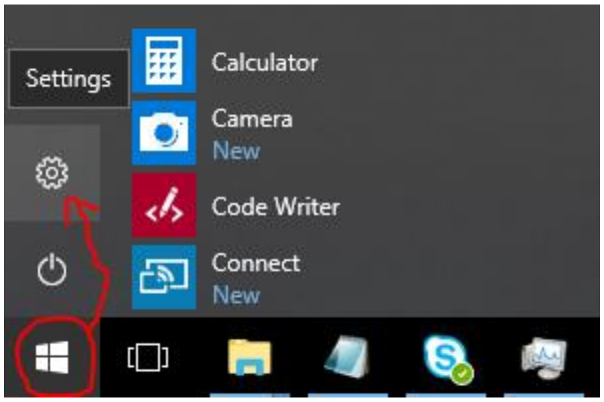
But, before you do that, you should gather the following information:

- Determine the number of processor cores available on your computer.
  If you know this already, you're done.
  If you don't, open a Microsoft Powershell (you will need one to run VelmaParallelCmdLine in any case) and run the command highlighted in this example:

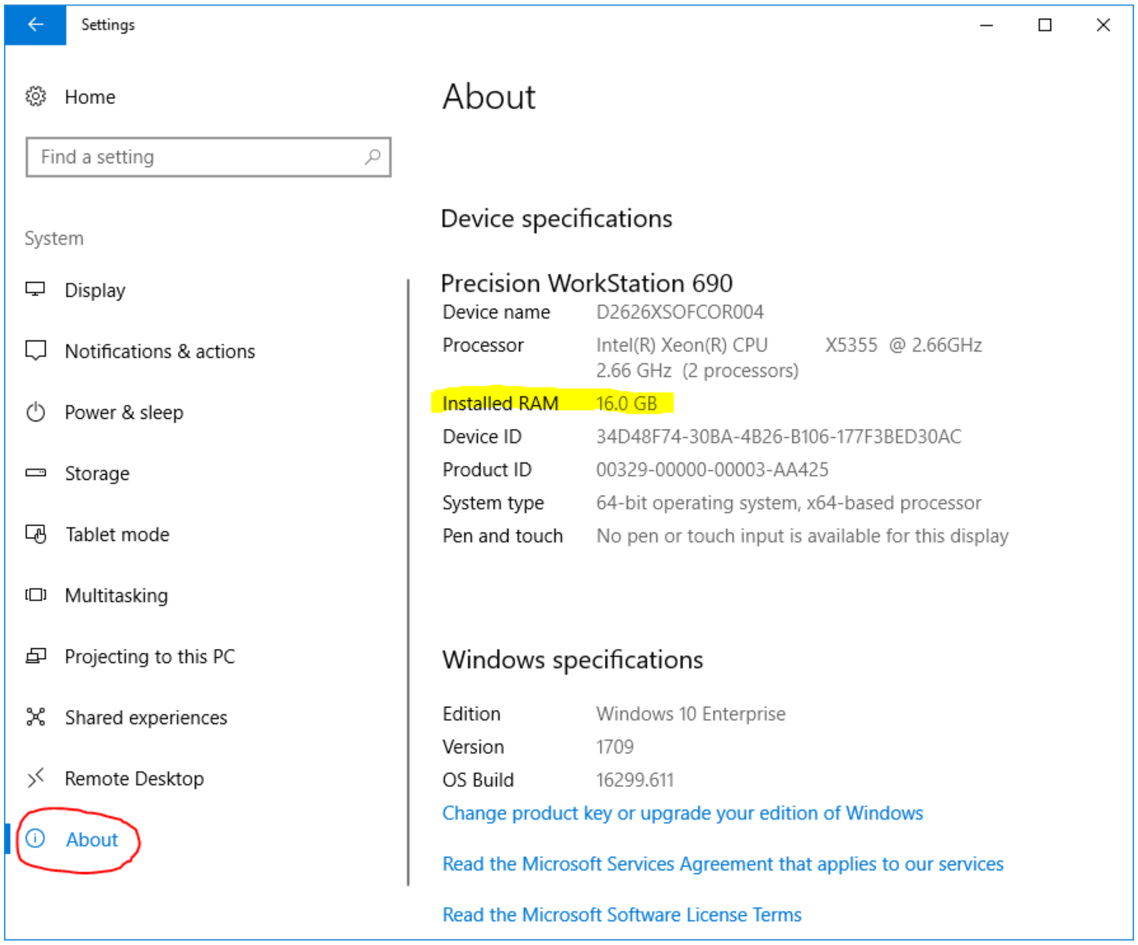
  ```
  PS C:\> Get-WmiObject Win32_Processor | % {$_.DeviceID + " cores=" + $_.numberOfCores}
  CPU0 cores=4
  CPU1 cores=4
  ```

  In the above example, the computer reports it has 2 CPUs, and each CPU has 4 processor cores, for a total of 8. As a rule of thumb, try to limit your VelmaParallelCmdLine runs to use less than the total number of available cores. Leaving 1 or 2 cores unused will keep the computer from becoming unresponsive to other processing requirements while your simulation is running.

- Determine the amount of RAM available on your computer.
  The simplest way to do this is to open the Windows Start -> Settings (the "gear" icon) Window, then click "System", and finally click "About".
  In the "About" window, look for the "Installed RAM" specification.
  Example:
  The Settings icon:



  Settings -> System -> About:



About

Device specifications

Precision WorkStation 690

Device name	D2626XSOFCOR004
Processor	Intel(R) Xeon(R) CPU X5355 @ 2.66GHz 2.66 GHz (2 processors)
Installed RAM	16.0 GB
Device ID	34D48F74-30BA-4B26-B106-177F3BED30AC
Product ID	00329-00000-00003-AA425
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

Edition	Windows 10 Enterprise
Version	1709
OS Build	16299.611

Change product key or upgrade your edition of Windows

Read the Microsoft Services Agreement that applies to our services

Read the Microsoft Software License Terms

Knowing the amount of RAM available is important, because you will probably want to specify an initial heap size (via Java’s -Xmx option), and VelmaParallelCmdLine will request the amount of memory you specify ***for each process starts***. This means that, if you request 6 processors and 4 gigabytes of memory (“-Xmx6g”) VelmaParallelCmdLine may request up to 6 x 6 = 36 gigabytes of memory at a time (if you have 6 or more sub-reaches that it determines can all be run in parallel).

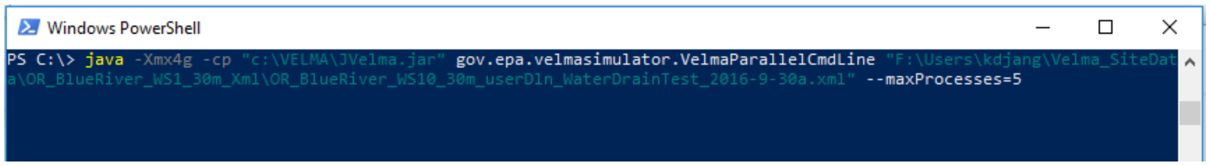
Once you know the number of available processes and memory, you can keep your VelmaParallelCmdLine startup within the limits of what your computer can process.

The VelmaParallelCmdLine front-end is part of the JVelma.jar. To run it, use the following command-line format, within a Powershell window:

```
java -Xmx<size> -cp “<path/JVelma.jar>”
gov.epa.velmasimulator.VelmaParallelCmdLine “<file.xml>” --maxProcesses=<n>
```

(*The above command should all be entered as one line – it wraps across more than one line in this document.*)

Here is an example of an actual command line:


Windows PowerShell

```
PS C:\> java -Xmx4g -cp "c:\VELMA\JVelma.jar" gov.epa.velmasimulator.VelmaParallelCmdLine "F:\Users\kdjang\Velma_SiteDat
a\OR_BlueRiver_WS1_30m_Xml\OR_BlueRiver_WS10_30m_userDln_WaterDrainTest_2016-9-30a.xml" --maxProcesses=5
```

In the above example …

- The <size> of "4g" for the -Xmx argument requests 4 gigabytes per process/sub-reach.
- The <path/JVelma.jar> argument is the full-path to the JVelma.jar on the example computer.
- The <file.xml> argument is the full-path of the simulation configuration .xml file to run.
- The <n> value of 5 asks VelmaParallelCmdLine to use up to 5 processors at a time.
- The <path/JVelma.jar> and <file.xml> are enclosed in double-quotes in case of whitespace characters in the path or file names.

VELMA Parallel Mode Results

Like JVelma, VelmaParallelCmdLine creates a new subdirectory using the .xml configuration file's simulation run name (i.e. the value of the run_index parameter) underneath the location specified by the initializeOutputDataLocationRoot parameter value. However, unlike JVelma, VelmaParallelCmdLine creates sub-subdirectories under the "root" output subdirectory for each sub-reach:

```
./<run_index-name>/
                |
                +--/Results_<a>/
                |            |
                |            GlobalStateLog.txt
                |            DailyResults.csv
                |            etc ...
                |
                +--/Results_<b>/
                |
                +--/Results_<c>/
                |
                etc ...
```

Each results directory contains the GlobalStateLog.txt, DailyResults.csv, and other files for that sub-reach's simulation run. The files in the Results_* directory of the primary watershed outlet are of primary importance when considering the overall watershed results, but the other Results_* folders' files may be of interest – especially the ReachFlowContributions.csv file which contain runoff values specific to that sub-reach alone.

Finally, again unlike JVelma, VelmaParallelCmdLine writes its own overall running log file during its operation, and leaves it underneath root results directory.
The file is always named VelmaParallelRunlog.txt, and it contains a meta-level log of the entire simulation run.

Here are the contents of an example:

```
PS C:\> cat .\VelmaParallelRunlog.txt
CONFIG 2018-07-20 10:16:43 VelmaParallelCmdLine: STARTING Parallel Processing
CONFIG 2018-07-20 10:16:43 VelmaParallelCmdLine: Processors: available=8 requested=4
utilized=4
CONFIG 2018-07-20 10:16:44 VelmaParallelCmdLine: Readying reach {WatershedInfo outlet=3590
dependencies=[] path=""}
CONFIG 2018-07-20 10:16:44 VelmaParallelCmdLine: Readying reach {WatershedInfo outlet=2434
dependencies=[] path=""}
CONFIG 2018-07-20 10:16:44 VelmaParallelCmdLine: Pending reach {WatershedInfo outlet=1735
dependencies=[2434, 2263] path=""}
CONFIG 2018-07-20 10:16:44 VelmaParallelCmdLine: Pending reach {WatershedInfo outlet=3236
dependencies=[3590] path=""}
CONFIG 2018-07-20 10:16:44 VelmaParallelCmdLine: Pending reach {WatershedInfo outlet=2263
dependencies=[3236] path=""}
CONFIG 2018-07-20 10:16:44 VelmaParallelCmdLine: Parallel VELMA initial status: reaches
pending=3 ready=2 completed=0
INFO 2018-07-20 10:16:44 VelmaParallelCmdLine: STARTED Reach iOutlet=3590
process=java.lang.ProcessImpl@5e2de80c memory="-Xmx1g"
INFO 2018-07-20 10:16:44 VelmaParallelCmdLine: STARTED Reach iOutlet=2434
process=java.lang.ProcessImpl@1d44bcfa memory="-Xmx1g"
INFO 2018-07-20 10:18:26 VelmaParallelCmdLine: TERMINATED Reach iOutlet=3590
process=java.lang.ProcessImpl@5e2de80c status=0
INFO 2018-07-20 10:18:26 VelmaParallelCmdLine: STARTED Reach iOutlet=3236
process=java.lang.ProcessImpl@266474c2 memory="-Xmx1g"
INFO 2018-07-20 10:19:48 VelmaParallelCmdLine: TERMINATED Reach iOutlet=2434
process=java.lang.ProcessImpl@1d44bcfa status=0
INFO 2018-07-20 10:20:39 VelmaParallelCmdLine: TERMINATED Reach iOutlet=3236
process=java.lang.ProcessImpl@266474c2 status=0
INFO 2018-07-20 10:20:39 VelmaParallelCmdLine: STARTED Reach iOutlet=2263
process=java.lang.ProcessImpl@6f94fa3e memory="-Xmx1g"
INFO 2018-07-20 10:24:52 VelmaParallelCmdLine: TERMINATED Reach iOutlet=2263
process=java.lang.ProcessImpl@6f94fa3e status=0
INFO 2018-07-20 10:24:52 VelmaParallelCmdLine: STARTED Reach iOutlet=1735
process=java.lang.ProcessImpl@5e481248 memory="-Xmx1g"
INFO 2018-07-20 10:27:07 VelmaParallelCmdLine: TERMINATED Reach iOutlet=1735
process=java.lang.ProcessImpl@5e481248 status=0
CONFIG 2018-07-20 10:27:10 VelmaParallelCmdLine: TERMINATED Parallel Processing
finalExitStatus=0
PS C:\>
```

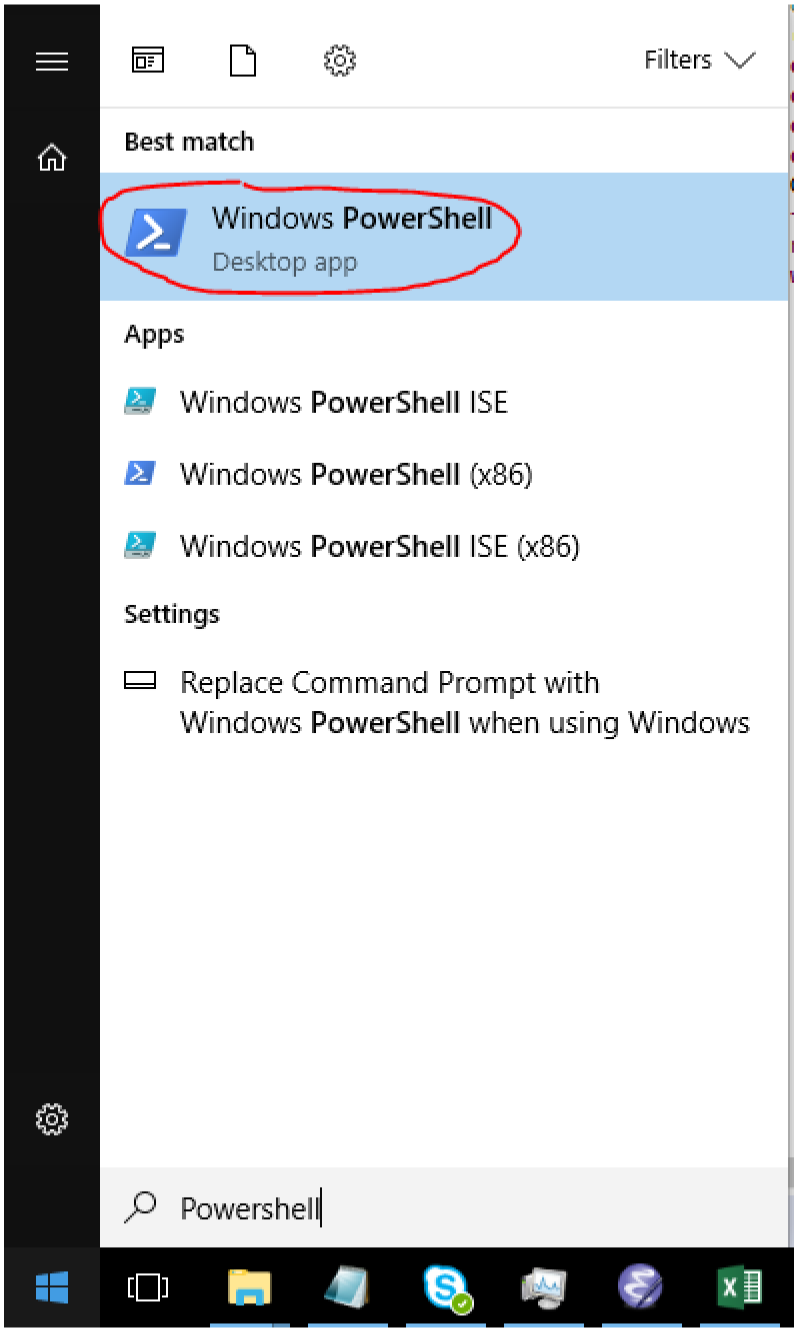
An exit status of “0” indicates success – any other value signals an error.

The VelmaParallelRunlog.txt file’s contents are useful for figuring out what went wrong when a VelmaParallelCmdLine run does not complete all the sub-reaches.

Appendix 1: How to Open a Microsoft PowerShell Window

Click the Windows Start button and start typing "Powershell".

The Start panel should change to show the following:



Click the "Windows PowerShell" (Best Match item) as shown above.

C.3 | Rationale for Parallel Velma and Reach Segmentation

Overview *(Tutorial C.3 – Rationale for Parallel Velma)*

This tutorial provides the rationale for setting up and running VELMA in parallel mode, whereby a large watershed is broken up into smaller subwatersheds that can be run in parallel more efficiently. This requires segmentation of the overall stream network.

VELMA links a hydrological model with a biogeochemical model. The modeling unit is a square grid cell with up to eight adjacent grid cells. Processing of each grid cell includes computing transfers of water and chemicals from adjacent uphill cells, and to adjacent downhill cells. Transfers into or out of a cell can happen from any adjacent uphill or downhill, so a cell may have multiple flow directions. Having multiple flow directions is more realistic in general but is problematic in certain cases. VELMA starts with a set of cells that are delineated from an outlet point and include all cells whose primary flow direction sends water to the given outlet. Cells that are on the edge of this set could have flow into cells that are not included in the set, and cells outside the set may have flow into the set. This is a problem for a couple reasons. Cells on the edge of the delineated watershed behave differently from cells within the watershed in that a pair of cells with the downhill cell in the watershed and the uphill cell not in the watershed does not contribute water from the uphill cell to the downhill cell, whereas a pair of cells with the same configuration inside the watershed would have water contributed from the uphill to the downhill cell.

A desirable property of VELMA watershed processing is to get identical results for a cell whether the cell is being processed as an edge cell or as an interior cell. This property would then extend to say that a sub-watershed processed independently should get identical results when processed as part of a larger watershed.

In VELMA we get this property by selecting a set of sub-outlets that divide the watershed into a set of catchments and then require that water transfer occur only when adjacent cells are in the same catchment or if the upper cell is one of the designated outlets. This produces several desirable behaviors. First, given a set of outlets, processing a cell or catchment independently will produce the same results if the cell or catchment is processed as part of a larger watershed. In addition, communication between catchments is only through the set of outlets, further streamlining processing. This allows catchments to be processed by VELMA independently, either serially or in parallel, allowing for faster processing and allowing for the processing of much larger watersheds. It produces a less desirable behavior though. The overall results depend on the assignment of outlets. Adding an outlet changes the water flow from cells that are now considered to be outside the catchment delineated by the new outlet. Differences are small but significant.

Another way to get identical behavior would be to include in processing all cells uphill from a given outlet. This ensures that for each cell, whether interior or edge, processing will be identical. However, in many cases, the additional number of cells can be quite large. For processing in parallel, the communication between catchments now many include all border cells, not just the outlets. The redundancy of processing plus the cost of communication between catchments makes parallel processing effectively worthless. Even in the purely serial case the amount of extra processing is significant.

Using square grid cells will always be a compromise. For channel cells we don’t account for how much of the cell is the stream and how much is riparian boundary. For ridge cells we don’t account for where the high point is within the cell. Adding an additional convention to ridge cells, that all water and chemicals flow into only one of the catchments divided by the cell, seems like a small cost to pay for the benefit of being about to subdivide a watershed and process catchments individually.

D.1 | Acquisition & Generation of VELMA Input Data

Overview *(Tutorial D.1 – Basic Acquisition and Generation of VELMA Input Data)*

This tutorial describes the minimum data input files required to run the VELMA simulator, where to acquire them, and steps to prepare them for a VELMA simulation.

The following data files must be complete and available for the VELMA Simulator to run:

FILE	TYPE	CONTAINS
Flat-Processed DEM	Spatial / Grid ASCII (.asc)	Elevation in meters
Cover Species ID Map	Spatial / Grid ASCII (.asc)	Cover Species ID integers
Cover Species Age Map	Spatial / Grid ASCII (.asc)	Cover Species age in years
Soil Parameters ID Map	Spatial / Grid ASCII (.asc)	Soil Parameterization ID integers
Precipitation Driver Data	Temporal / (.csv or .txt)	Precipitation per in mm
Air Temperature Driver Data	Temporal / (.csv or .txt)	Air Temperature in degrees C

Digital Elevation Model and Soils

Elevation and other common GIS data can be retrieved from many local, state, and federal GIS repository libraries. The USDA’s “Geospatial Data Gateway” provides easy access to multi resolution USGS National elevation data along with many other types of natural resource data. Higher resolution data, like LiDAR can many times be found from alternative public and private stakeholders such as the “Puget Sound Lidar Consortium”. Other popular data, such as STATSGO and newer 10m gridded SSURGO soils GIS layers can also be acquired from the “Geospatial Data Gateway”.

References

Geospatial Data Gateway - https://datagateway.nrcs.usda.gov/

Puget Sound Lidar Consortium - http://pugetsoundlidar.ess.washington.edu/

Weather Data

Daily meteorological summaries driver data can most easily be acquired from both observational and modeled sources such as:

- NOAA - https://www.ncdc.noaa.gov/cdo-web/
- SNOTEL - https://www.wcc.nrcs.usda.gov/snow/
- PRISM – http://www.prism.oregonstate.edu/
- DAYMET - https://daymet.ornl.gov/

Care needs to be taken with all weather data drivers to ensure completeness of temporal coverage and accuracy of values. Consistently, observational data will have “not available” (NA’s) or gaps in coverage, false readings from equipment malfunction, and user input errors. Modeled data tends to bring with it other types of errors or consistency gaps, which will need to be accounted for. For instance, DAYMET estimates are updated yearly, currently only available for 1980 – 2017, and uniformly truncate leap years to 365 days, where the VELMA model accounts for leap days. Benhnke et al. 2016 highlight the general comparison of observational versus modeled errors, as well as the regional variations of the various popular downscale gridded meteorological data. Appended tools and alternative weather models were created to deal with these and other inconsistencies in daily diver data.

References

Behnke, R., Vavrus, S., Allstadt, A., Albright, T., Thogmartin, W. E., & Radeloff, V. C. (2016). Evaluation of downscaled, gridded climate data for the conterminous United States. Ecological Applications.

PRISM Climate Group, Oregon State University, http://prism.oregonstate.edu

Thornton, P.E., M.M. Thornton, B.W. Mayer, Y. Wei, R. Devarakonda, R.S. Vose, and R.B. Cook. 2016. Daymet: Daily Surface Weather Data on a 1-km Grid for North America, Version 3. ORNL DAAC, Oak Ridge, Tennessee, USA. http://dx.doi.org/10.3334/ORNLDAAC/1328

Cover Species and Land Cover

Species and land cover data can be either custom generated or retrieved from data libraries. There are several national and regional scale land cover and species type data sets to be chosen from. The popular remotely sensed National Land Cover Data for the years 2006 and 2011 can be acquired from either the “Multi-Resolution Land Characteristics consortium” or “Geospatial Data Gateway”. The Gap Analysis Program (GAP) has create a national scale vegetation inventory, which was also combined with LANDFIRE, another national GIS data set, to help with both ecosystem management and fire resource management. For Pacific coast states, gradient analysis and nearest neighbor imputation (GNN) techniques were used to create more accurate forested species cover maps. Since cover species is site specific, general reductions can be made from national data sets to meet the user’s needs.

References

NLCD - https://www.mrlc.gov/data/statistics/national-land-cover-database-2011-nlcd2011-statistics

GAP - https://gapanalysis.usgs.gov/gaplandcover/

LANDFIRE - https://www.landfire.gov/

GNN – https://lemma.forestry.oregonstate.edu/data/structure-maps

Gergely, K.J., and McKerrow, A., 2016, Terrestrial ecosystems—National inventory of vegetation and land use (ver. 1.1, August 2016): U.S. Geological Survey Fact Sheet 2013–3085, 1 p., https://pubs.usgs.gov/fs/2013/3085/.

Homer, C.G., Dewitz, J.A., Yang, L., Jin, S., Danielson, P., Xian, G., Coulston, J., Herold, N.D., Wickham, J.D., and Megown, K., 2015, Completion of the 2011 National Land Cover Database for the conterminous United States-Representing a decade of land cover change information. Photogrammetric Engineering and Remote Sensing, v. 81, no. 5, p. 345-354

LANDFIRE, LANDFIRE: Existing Vegetation Type, U.S. Department of Agriculture and U.S. Department of the Interior.

Ohmann, Janet L.; Gregory, Matthew J. 2002. Predictive mapping of forest composition and structure with direct gradient analysis and nearest neighbor imputation in coastal Oregon, U.S.A. Canadian Journal of Forest Research. 32: 725-741

Cover Species Age

In some instances, stand ages maps might exist, but generally cover species age maps are custom made for each VELMA area of interest (AOI) model run. Age maps may be developed from remotely sensed data, surveys, land ownership maps, expert knowledge, or some combination of these methods. Below is an example of a custom species age creation process.

The Winant AOI, is a small tidally influence watershed that drains into the Yaquina bay along the Oregon coast. Modeling efforts are being made to quantify and qualify how nutrients are transport and cycled through a salt marsh whose headwaters are drainages of typical coastal conifer species and high nitrogen fixing hardwoods. It was there for necessary to model both types of unique biogeochemistry of the contributing Alder and Conifer forests, and their ages.

(continued)

Cover species maps were developed for the Winant AOI from a combination of ortho-imagery, aerial imagery, and on-site ground surveys (Figure 1).

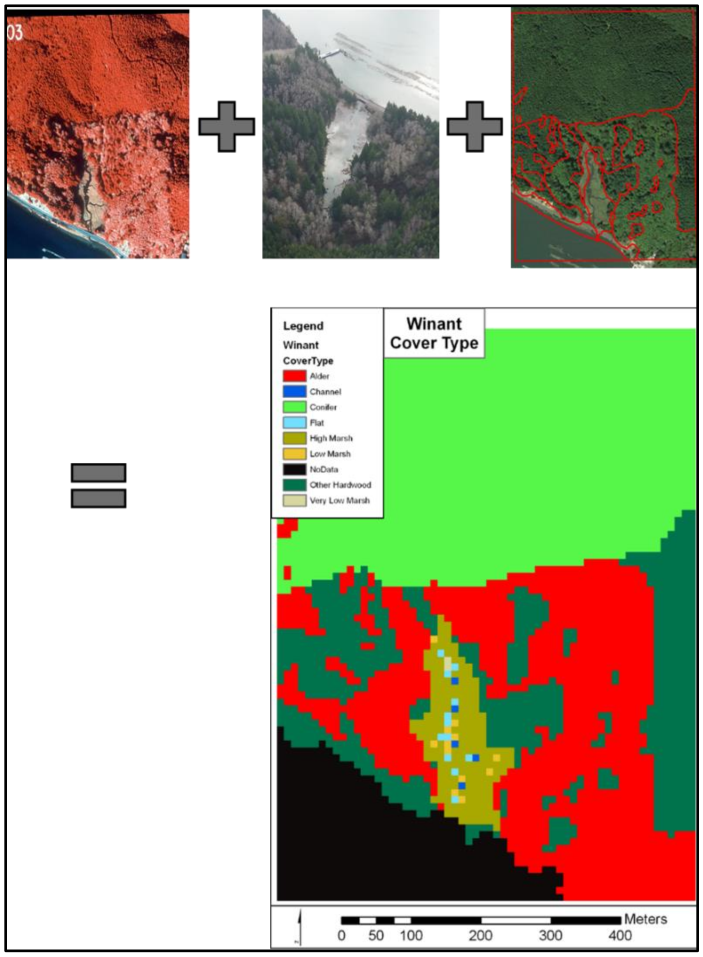
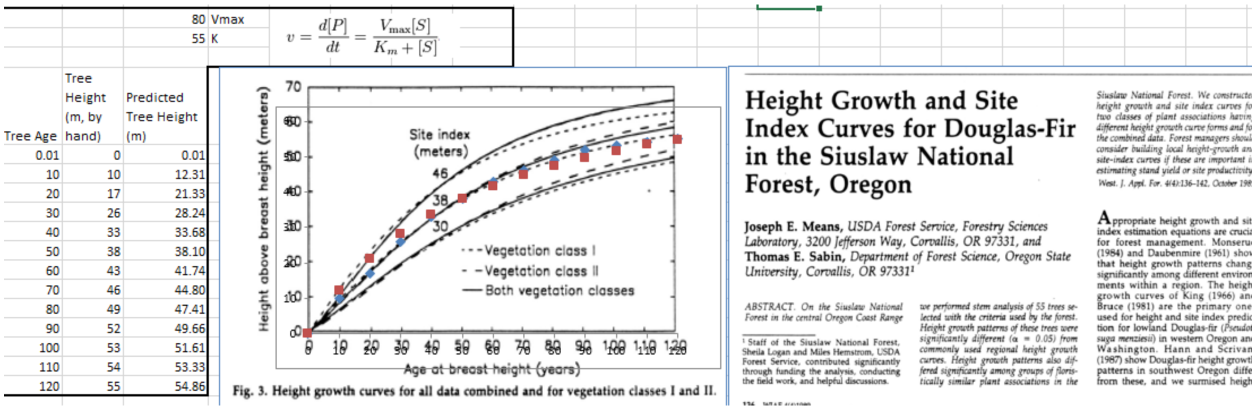


Figure 1. From Top left to right to bottom, ortho-imagery, aerial imagery, and site survey were combined to create cover type raster maps.

To generate species specific age maps, Michaelis-Menten curves we developed using literature data to produce functions that related species specific age to height. LiDAR height values were used to run the fitted curves of a particular species and derive its appropriate age cell value (Figure 2 and 3). Similar techniques can be used to develop relationships between above ground biomass or above ground carbon and vegetation age. Woods Hole has produced a national data set which include estimates of above ground carbon and vegetation height (Kellndorfer et al, 2012).



		80	Vmax
		55	K

$$v = \frac{d[P]}{dt} = \frac{V_{\max}[S]}{K_m + [S]}$$

Tree Age	Tree Height (m, by hand)	Predicted Tree Height (m)
0.01	0	0.01
10	10	12.31
20	17	21.33
30	26	28.24
40	33	33.68
50	38	38.10
60	43	41.74
70	46	44.80
80	49	47.41
90	52	49.66
100	53	51.61
110	54	53.33
120	55	54.86

Fig. 3. Height growth curves for all data combined and for vegetation classes I and II.

Height Growth and Site Index Curves for Douglas-Fir in the Siuslaw National Forest, Oregon

Joseph E. Means, *USDA Forest Service, Forestry Sciences Laboratory, 3200 Jefferson Way, Corvallis, OR 97331, and* **Thomas E. Sabin,** *Department of Forest Science, Oregon State University, Corvallis, OR 97331*[1]

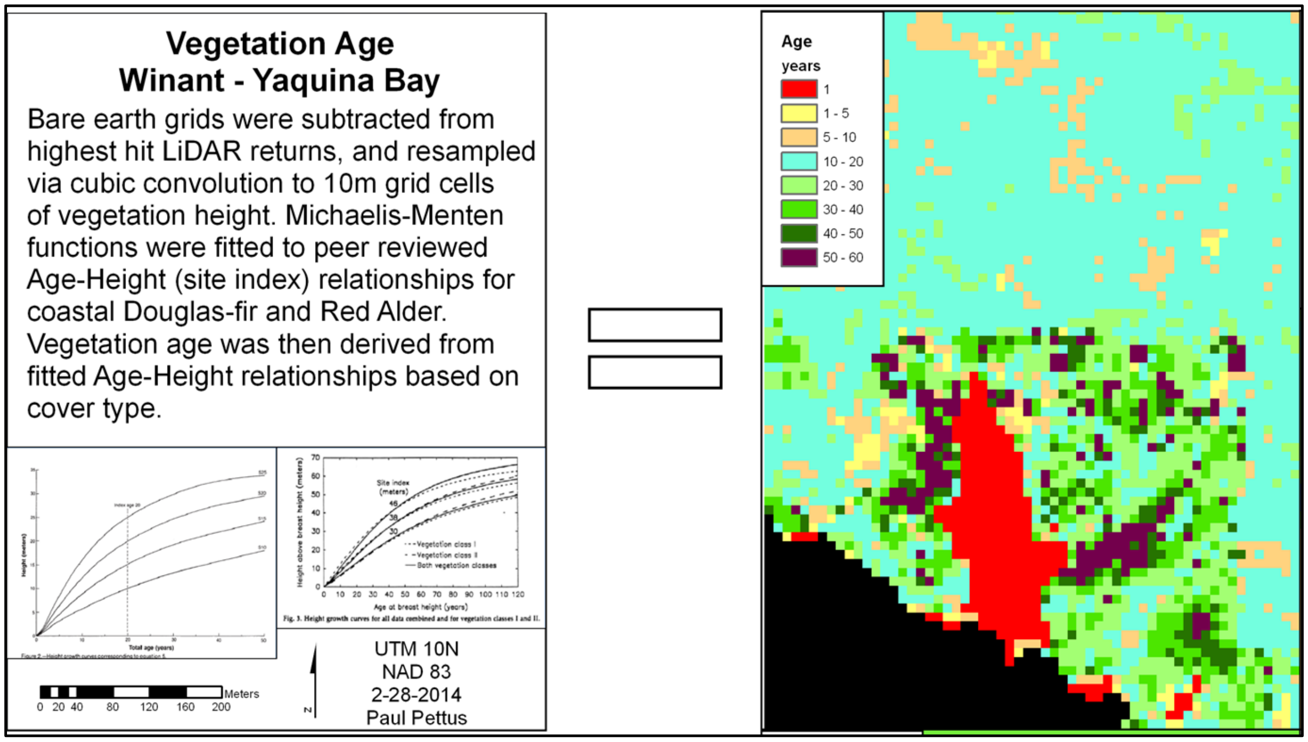
ABSTRACT. On the Siuslaw National Forest in the central Oregon Coast Range we performed stem analysis of 55 trees selected with the criteria used by the forest. Height growth patterns of these trees were significantly different (α = 0.05) from commonly used regional height growth curves. Height growth patterns also differed significantly among groups of floristically similar plant associations in the Siuslaw National Forest. We constructe height growth and site index curves fo two classes of plant associations havin different height growth curve forms and fo the combined data. Forest managers shoul consider building local height-growth an site-index curves if these are important i estimating stand yield or site productivity

West. J. Appl. For. 4(4):136–142, October 198

[1] Staff of the Siuslaw National Forest, Sheila Logan and Miles Hemstrom, USDA Forest Service, contributed significantly through funding the analysis, conducting the field work, and helpful discussions.

Appropriate height growth and sit index estimation equations are crucia for forest management. Monseru (1984) and Daubenmire (1961) show that height growth patterns chang significantly among different environ ments within a region. The heigh growth curves of King (1966) an Bruce (1981) are the primary one used for height and site index predic tion for lowland Douglas-fir (*Pseudot suga menziesii*) in western Oregon an Washington. Hann and Scrivan (1987) show Douglas-fir height growt patterns in southwest Oregon diffe from these, and we surmised heigh

Figure 2. Height to age index curve calculation for Douglas-Fir.



Vegetation Age
Winant - Yaquina Bay

Bare earth grids were subtracted from highest hit LiDAR returns, and resampled via cubic convolution to 10m grid cells of vegetation height. Michaelis-Menten functions were fitted to peer reviewed Age-Height (site index) relationships for coastal Douglas-fir and Red Alder. Vegetation age was then derived from fitted Age-Height relationships based on cover type.

UTM 10N
NAD 83
2-28-2014
Paul Pettus

Figure 3. Site index and curve fitted relationships species age and tree height.

References

Kellndorfer, J., Walker, W., LaPoint, E., Bishop, J., Cormier, T., Fiske, G., Hoppus, M., Kirsch, K., and Westfall, J. 2012. NACP Aboveground Biomass and Carbon Baseline Data (NBCD 2000), U.S.A., 2000. Data set. Available on-line [http://daac.ornl.gov] from ORNL DAAC, Oak Ridge, Tennessee, U.S.A. http://dx.doi.org/10.3334/ORNLDAAC/1081

D.2 | Soil Data Mapping and Parameter Initialization

Overview *(Tutorial D.2 – Soil Data Mapping and Parameter Initialization)*

This tutorial describes the minimum data input files required to run the VELMA simulator, where to acquire them, and steps to prepare them for a VELMA simulation.

Soil Data Sources

- gSSURGO - https://gdg.sc.egov.usda.gov/
    - gSSURGO User Guide (PDF; 10.7 MB) Soil Survey Staff. Gridded Soil Survey Geographic (gSSURGO) Database for Washington. United States Department of Agriculture, Natural Resources Conservation Service.
- STATSGO2 - https://gdg.sc.egov.usda.gov/
    - Soil Survey Staff. Gridded Soil Survey Geographic (STATSGO2) Database for Washington. United States Department of Agriculture, Natural Resources Conservation Service.

Soil Data Pre-Processing

To create a thematic soil texture map for either the gSSURGO, SSURGO, or STATSGO2 data sets the one too many to many hierarchies of relationships must be resolved with spatial map units and tabular data. Both soil tabular and spatial data tables are included when any of the data sets are downloaded from the USDA. For a complete overview of relationships, refer to the SSURGO/STATSGO2 documentation “SSURGO 2.2 Data Model – Diagram 1 of 2”. To link the spatial map units to their textures a series of joins must be made between the tabular text file and the Shapefiles (Figure 1).

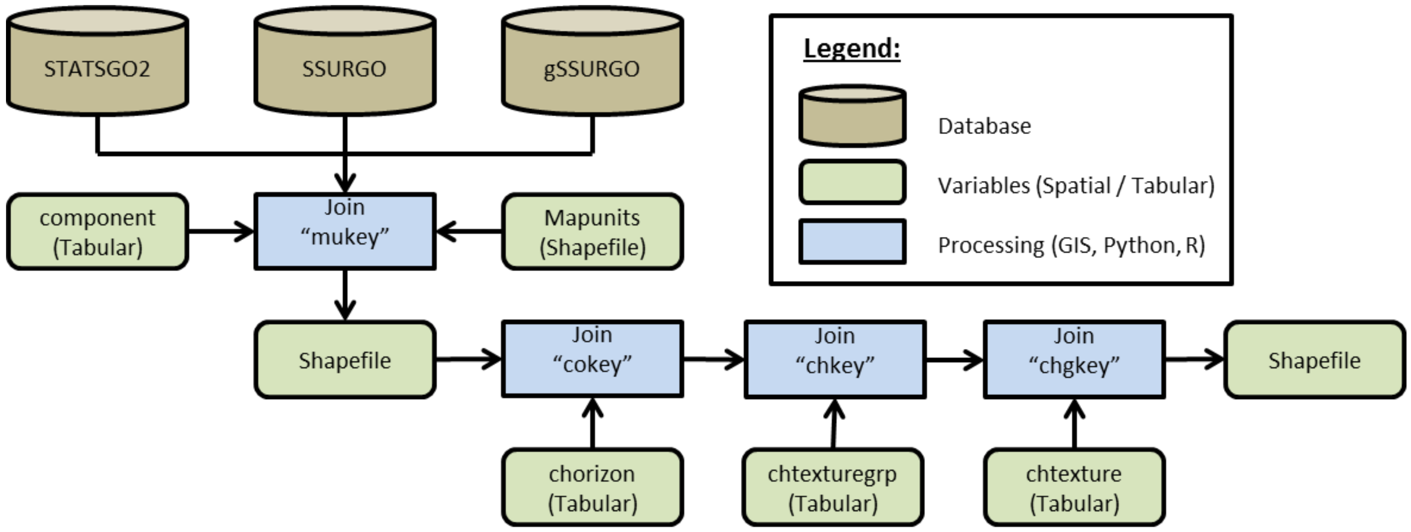


Figure 3. Spatial join process

Example: Tolt River Watershed Soil Data Pre-Processing

1. Within ArcGIS, load the STATSGO2 shapefile "gsmsoilmu_a_wa.shp"
2. Using the "Join Field" (Data Management) tool set:
    - Input Table = "gsmsoilmu_a_wa.shp"
    - Input Join Field = "MUKEY"
    - Join Table = PATH + "component"
    - Output Join Field = "mukey"
    - Join Fields (optional) = CHECK "cokey"
3. Repeat "Join Field" for the C horizon, chtexturegrp, and chtexture tables to the final texture column.

- Dataset 1.
    - In cases where the gSSURGO soil textures were initially classified under subclass, the soil textures need to be re-classified to their parent class of the USDA Soil Survey Manual, Chapter 3 (USDA, 2017) of the soil triangle textures that are used by VELMA (Figure 2).
        - Sands <- Coarse Sand, Sand, Fine Sand, Very Fine Sand
        - Loamy Sand <- Loamy Coarse Sand, Loamy Sand, Loamy Fine Sand, Loamy Very Find Sand
        - Sandy Loams <- Coarse Sandy Loam, Sandy Loam, Fine Sandy Loam, Very Fine Sandy Loam
        - Silt Loam = Silt

(continued)

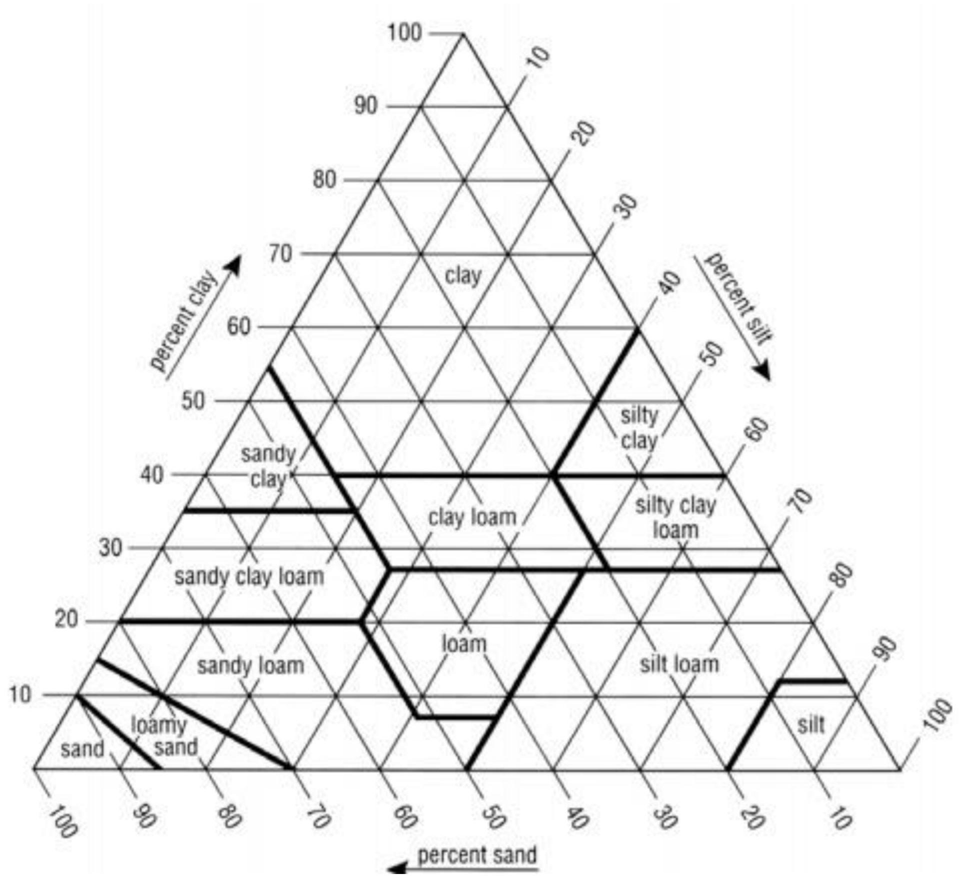


Figure 4. USDA Soil texture triangle

- Replaced all missing gSSURGO soils texture cells with STATSGO soil textures, creating a merged high- and low-resolution soils data layer (Table 1, Figure 4, Figure 5, and Figure 6).

Table 1. Soil texture key for VELMA

File Name

tolt_ssurgo_statsgo_soil.tif

Key	Value
1	sand
2	loamy sand
3	sandy loam
4	loam
5	silt loam
6	sandy clay loam
7	clay loam
8	silty clay loam
9	sandy clay
10	silty clay
11	clay

Software Requirements and Processing

To run and produce the merged soil texture map (ASCII), you will need the following:

A. Python version 2.x: Current release is 2.7.11
   1. Python comes pre-packaged within ArcGIS, so it is likely you already have it installed on your computer. For example, a default install is currently here: C:\Python27\ArcGIS10.2\python.exe. Check to see if you have Python installed before installing a new version. If Python is not installed, you can obtain a copy here: https://www.python.org/download/releases/2.7/ *Note that Python 2.7 is currently considered safe for use on U.S. EPA network and non-network computers.*

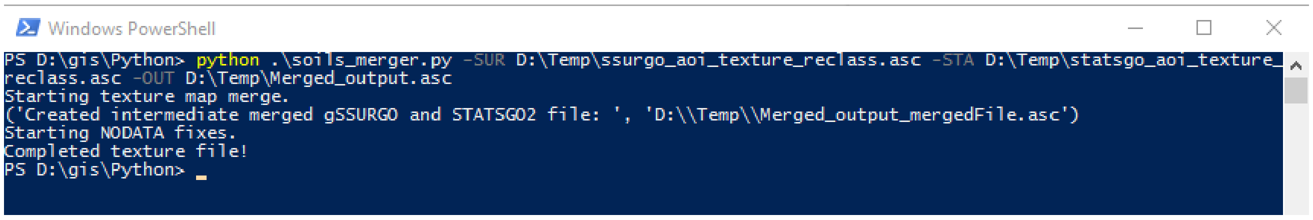
B. ASCII Map requirements:
   1. gSSURGO ASCII format (.asc) for the AOI and a STATSGO2 ASCII format (.asc) for the AOI.
      - Cell values set to match the soil texture key for VELAM (Table 1).
      - No-data cells values set to -9999, which is the default ERSI ASCII no data value.
      - These map(s) should match the exact extent and resolution of the AOI / DEM ASCII regardless if they are higher resolution.
   2. Output filename to be determined by the user, ASCII format (.asc).

C. “**`Soil_merger.py`**” and can be run on the Command Prompt line with Python. This script requires at a minimum a gSSURGO map, one STATSGO2 map, and one output file name. The script will output an intermediate merged file map with No-data values. The merged soil texture values will assign values to the output map from the higher resolution gSSURGO map first and then from the STATSGO2 map. The final output map will fill No-data values with soil texture values. The technique uses a nearest neighbor search algorithm that will radiate out in one cell radius increments searching for the nearest cell with a valid texture value which will then be assigned to the No-data cell.

Command prompt input example:

“`python .\soils_merger.py -SUR D:\Temp\ssurgo_aoi_texture_reclass.asc -STA D:\Temp\statsgo_aoi_texture_reclass.asc -OUT D:\Temp\Merged_output.asc`”

“`python C:\Path\To\soils_merger.py -help`” for exact arguments (Figure 4).


Windows PowerShell

```
PS D:\gis\Python> python .\soils_merger.py -SUR D:\Temp\ssurgo_aoi_texture_reclass.asc -STA D:\Temp\statsgo_aoi_texture_
reclass.asc -OUT D:\Temp\Merged_output.asc
Starting texture map merge.
('Created intermediate merged gSSURGO and STATSGO2 file: ', 'D:\\Temp\\Merged_output_mergedFile.asc')
Starting NODATA fixes.
Completed texture file!
PS D:\gis\Python> _
```

Figure 5. Windows PowerShell python example

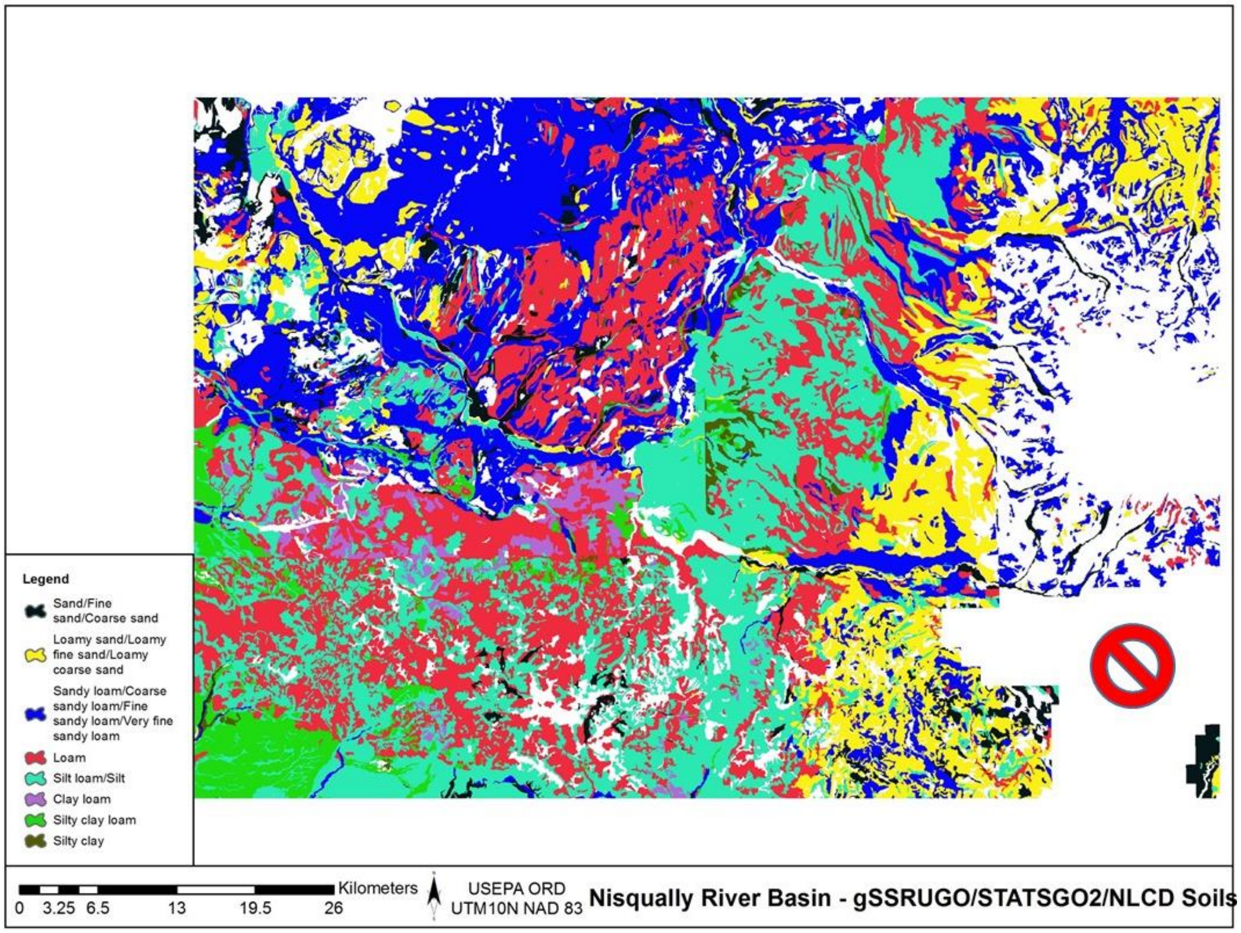

Legend
Sand/Fine sand/Coarse sand
Loamy sand/Loamy fine sand/Loamy coarse sand
Sandy loam/Coarse sandy loam/Fine sandy loam/Very fine sandy loam
Loam
Silt loam/Silt
Clay loam
Silty clay loam
Silty clay
0 3.25 6.5 13 19.5 26 Kilometers
USEPA ORD
UTM10N NAD 83
Nisqually River Basin - gSSRUGO/STATSGO2/NLCD Soils

Figure 4. gSSURGO Soils texture and NODATA

(continued)

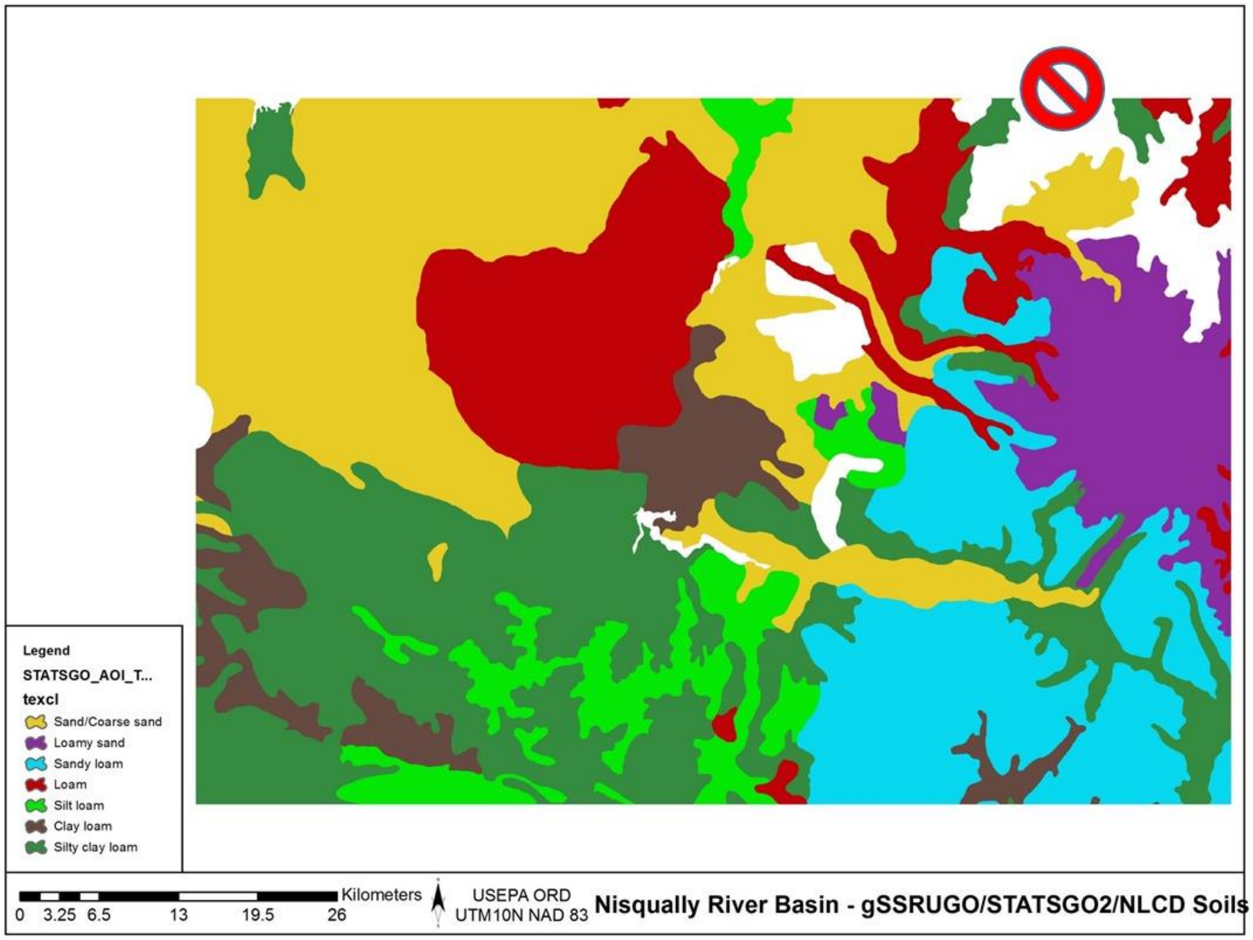


Figure 5. STATSGO Soils texture and NODATA

(continued)

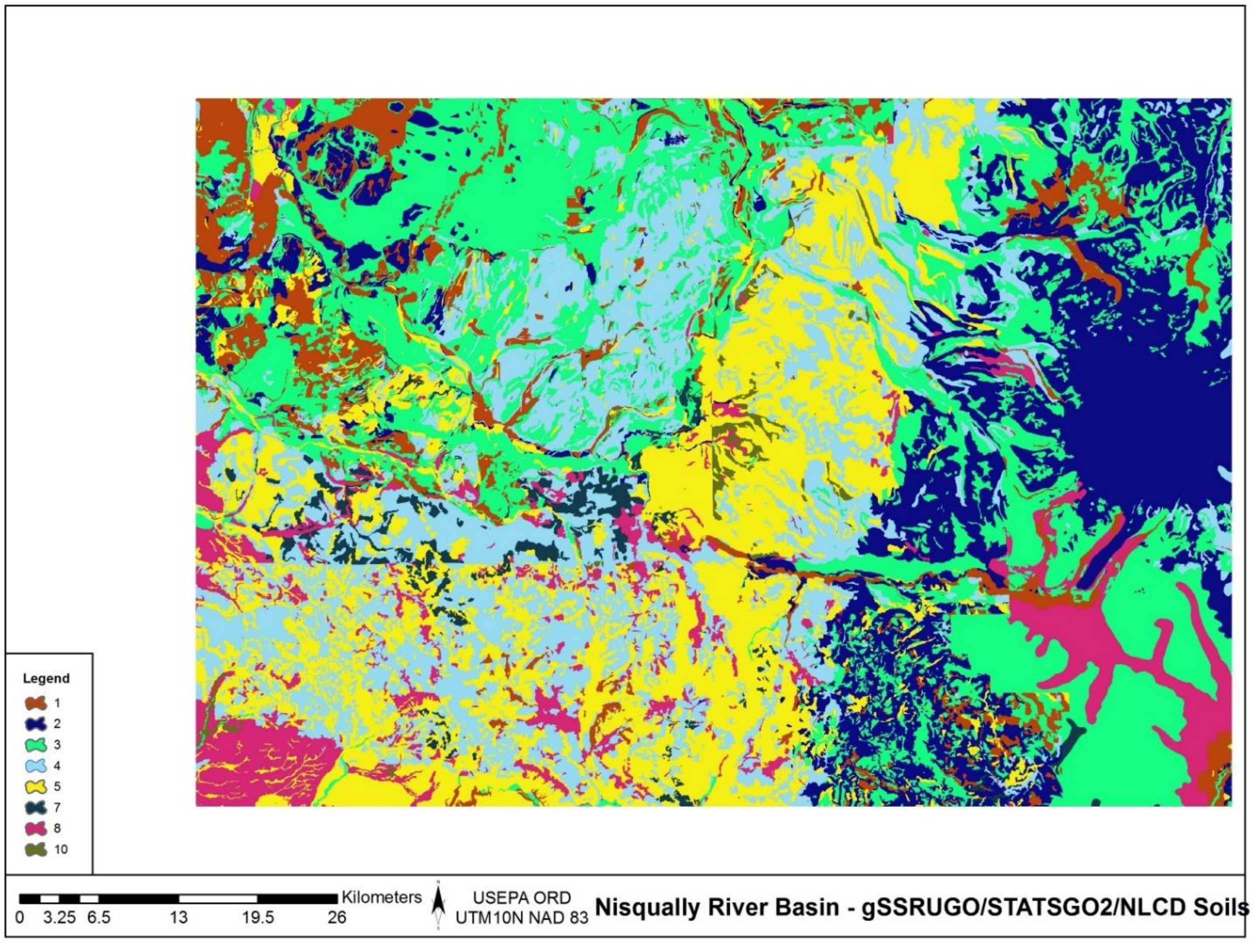


Figure 6. Merged soils, gSSURGO and STATSGO.

References

Soil Science Division Staff. 2017. Soil survey manual. C. Ditzler, K. Scheffe, and H.C. Monger (eds.). USDA Handbook 18. Government Printing Office, Washington, D.C.

Scripts

```
# soils_merger.py
# Author: Paul Pettus
# Date: 10-20-2017
# Description: Merge STATSGO2 and gSSURGO soil texture ASCII Maps
#
# Output is a single ASCII map that will have an assigned soil texture value
# for every cell
#
# By default, cell values will be first assigned the value of the higher
# resolution gSSURGO and if no data exists in the gSSURGO map values
# will be assigned to the output from the lower resolution STATSGO2.
#
# If no values are found in either soil input maps, nodata cells will
# be assigned a nearest neighbor value from a circling radial stepping
# algorithm. The stepping algorithm steps one cell radius per iteration until
# it finds a soil texture cell value.  The algorithm ignores border cells.
```

```
#
# Last updated: 11-16-2017

import os, sys, numpy, re, argparse, itertools

# ----------------------------------------------------------------------------------------------------


# Error message class
class Usage(Exception):
    def __init__(self, msg):
        self.msg = msg

def main(argv=None):

    if argv is None:
        argv = sys.argv

    try:
        parser = argparse.ArgumentParser(description='Output is a single ASCII map that will'+
                                    ' have an assigned soil texture value for every cell')

        parser.add_argument('-SUR', action='store', dest='surFILE',default='D:/GIS/Nisqually/Soil/ssurgo_aoi_texture_reclass.tif.asc',
                    help='Fully-qualified path + name of ".asc" gSSURGO soil texture file.')

        parser.add_argument('-STA', action='store', dest='staFILE',default='D:/GIS/Nisqually/Soil/statsgo_aoi_texture_reclass.asc',
                    help='Fully-qualified path + name of ".asc" STATSGO2 soil texture file.')

        parser.add_argument('-OUT', action='store', dest='outFILE',default='D:/GIS/Nisqually/Soil/build_old_timey_mudButt_6.asc',
                    help='Fully-qualified path + name of ".asc" output file.')

        args = parser.parse_args()

        # args parsing
        ssurgoAsc = os.path.abspath(args.surFILE)
        statsgoAsc = os.path.abspath(args.staFILE)
        buildFile = os.path.abspath(args.outFILE)

        # Check that files exists
        if not os.path.exists(ssurgoAsc):
            raise Usage('Cannot find AOI file "' + ssurgoAsc + '"')

        if not os.path.exists(statsgoAsc):
            raise Usage('Cannot find AOI file "' + statsgoAsc + '"')

        # do the work
        mergeSoils(ssurgoAsc,statsgoAsc,buildFile)

    except Usage as e:
        print(e.msg)
        return 2

    except Exception as e:
        # STUB exception handler
        # Warning: poor programming style.
        # Catches almost any exception (but not KeyboardInterrupt -- which is a Good Thing)
        raise e

# ----------------------------------------------------------------------------------------------------
# Return an ascii file header
def readHeader(asciiFile):

    if not os.path.exists(asciiFile):
        raise Usage('Cannot find ASCII "' + asciiFile + '"')

    # Open file and read in header info
    readFile = open(asciiFile)

    header = readFile.readline()  #ncols
    header += readFile.readline() #nrows
    header += readFile.readline() #xllcorner
    header += readFile.readline() #yllcorner
    header += readFile.readline() #cellsize
    header += readFile.readline() #NODATA_value
    readFile.close()

    return header

# -------------------------------------------------------------------------------------------------
# Loop by one + one cell radius of surrounding cells to find nearest neighbor cell value
def lookAround(passRow,passCol,inArray):

    inRow, inCol = inArray.shape

    # Initial search radius at one cell
    radius = 1

    found = False  # found a value

    # Loop by one + one cell radius of surrounding cells to find nearest neighbor cell value
    while found != True:

        rowList = [0]  #
        colList = [0]

        # Create search box of one cell distance
        if radius == 1:

            for i in xrange(radius):
                for j in xrange(radius):
                    rowList.append((i + 1) * -1)
                    rowList.append(i + 1)
                    colList.append((i + 1) * -1)
                    colList.append(i + 1)

            # creates a list of the relative coordinates which to search around the missing value cell
            setList = list(itertools.product(rowList, colList))

        # Else cell radius is larger than one cell
        else:

            rowList = [0]
            colList = [0]

            # Create an inner one radius cell shorter search box
            # Keeps track of already searched cells in radius
            for i in xrange((radius-1)):
                for j in xrange((radius-1)):
                    rowList.append((i + 1) * -1)
                    rowList.append(i + 1)
                    colList.append((i + 1) * -1)
                    colList.append(i + 1)

            minusList = list(itertools.product(rowList, colList))

            rowList = [0]
            colList = [0]

            # Create an full radius cell search box around the missing value cell
            for i in xrange(radius):
                for j in xrange(radius):
                    rowList.append((i + 1) * -1)
                rowList.append(i + 1)
                colList.append((i + 1) * -1)
                colList.append(i + 1)

            fullList = list(itertools.product(rowList, colList))

            # Select only the out radius cells from full box list
            setList = list(set(fullList) - set(minusList))

        for item in setList:

            # Check that searched cells are not out of array bounds
            if ((passRow + item[0]) >= 0) and ((passRow + item[0]) < inRow) and ((passCol + item[1]) >= 0) and ((passCol + item[1]) < inCol):

                # nearest neighbor cell value and return it
                value = inArray[(passRow + item[0]),(passCol + item[1])]

                if value != -9999:
                    found = True
                    return value
        # Increase radius by a cell if NA / no values are found
        radius = radius + 1

# ---------------------------------------------------------------------------------------------------
# Merge SSUGO STATSGO Soils, then replace nodata values
def mergeSoils(ssurgoAsc,statsgoAsc,buildFile):

    # Load ssrgo array file
    ssgoArray = numpy.loadtxt(ssurgoAsc, skiprows=6)
    # Load statsgo array file
    statsArray = numpy.loadtxt(statsgoAsc, skiprows=6)

    row, col = ssgoArray.shape
    # Create new merge array
    mergeArray = numpy.zeros((row,col))

    print("Starting texture map merge.")

    for i in xrange(row):
        for j in xrange(col):
            # Get soil values
            ssgoValue = ssgoArray[i,j]
            statsValue = statsArray[i,j]

            # Assign higher resolution ssurgo values first
            if ssgoValue != -9999:
                mergeArray[i,j] = ssgoValue
            # Assign lower resolution statsgo next
            elif statsValue != -9999:
                mergeArray[i,j] = statsValue
            # Assign no data value if neither has a set has a value
            else:
                mergeArray[i,j] = -9999

    # Merged ssurgo statsgo, export complete ascii

    fileName, fileExtension = os.path.splitext(buildFile)

    mergeFile = fileName + "_mergedFile" + fileExtension

    header = readHeader(ssurgoAsc)

    f = open(mergeFile, "w")
    f.write(header)
    numpy.savetxt(f, mergeArray, fmt="%i")
    f.close()
    print("Created intermediate merged gSSURGO and STATSGO2 file: ", mergeFile)

    reloadArray = numpy.loadtxt(mergeFile, skiprows=6)

    noDataArray = numpy.zeros((row,col))

    print("Starting NODATA fixes.")

    # Replace nodata cells with search radius algorythm
    for i in xrange(row):
        for j in xrange(col):
            mergeValue = reloadArray[i,j]

            # if nodata, send to search algorythm
            if mergeValue == -9999:
                newValue = lookAround(i,j,reloadArray)

                noDataArray[i,j] = newValue
            # else keep merged data value
            else:
                noDataArray[i,j] = mergeValue

    # Merged ssurgo statsgo and nodata filled, export complete ascii
    header = readHeader(ssurgoAsc)

    outputFile = buildFile

    f = open(outputFile, "w")
    f.write(header)
    numpy.savetxt(f, noDataArray, fmt="%i")
    f.close()

    print("Completed texture file!")


if __name__ == "__main__":
    sys.exit(main())

# ---------------------------------------------------------------------------------------------------
# Old code for internal testing only
#####################################################################################################
### Modify files and locations
#####################################################################################################
### SSURGO ASCII file
##ssurgoAsc = "D:/GIS/Nisqually/Soil/ssurgo_aoi_texture_reclass.tif.asc"
### STATSGO ASCII file
##statsgoAsc = "D:/GIS/Nisqually/Soil/statsgo_aoi_texture_reclass.asc"
### SSURGO and STATSGO merged ASCII file, intermediate
##mergeFile = "D:/GIS/Nisqually/Soil/merged_soils_5.asc"
### SSURGO and STATSGO merged ASCII file with nodata fixed, final output
##buildFile = "D:/GIS/Nisqually/Soil/build_old_timey_mudButt_5.asc"
#####################################################################################################
##
### Call main function to merge and replace nodata values
##mergeSoils(ssurgoAsc,statsgoAsc,buildFile)
```

D.3 | Initialize a Spatial Data Pool Using an Ascii Grid

Overview *(Tutorial D.3 – Initialize a Spatial Data Pool Using an Ascii Grid)*

VELMA is a spatially-distributed model, with each grid cell within a watershed representing a soil column and vegetation with characteristics that can vary greatly in space, reflecting variations in geology, land cover, land use and many other factors.

Before VELMA can reasonably simulate cell-to-cell transfers of water and chemicals within a real-world watershed, all supporting soil and vegetation spatial data must be initialized for day 1 of the simulation.

This tutorial describes the basic procedure for initializing soil and vegetation spatial data pools that use an Ascii grid of the same size and georeferencing as the DEM.

General Procedure

VELMA Simulator spatial data pools may be initialized by specifying “Set Spatial Data By Map” disturbances as part of a simulator run’s simulation configuration. Each spatial data pool (and each layer of that pool for multi-layer pools) that is initialized must have a separate disturbance specified for it, and its own Grid Ascii (.asc) map file available for the disturbance to use as a data source[1].

Assuming an otherwise already configured VELMA application, load the configuration (XML file) into JVelma. To initialize a spatial data pool (using the leaf biomass pool as our example), add a “Set Spatial Data By Map” disturbance event to the configuration:

Note
It’s assumed that before configuring the simulation, you’ve already created a Grid Ascii map file (.asc) for the disturbance to use. How you create the map is up to you, but it must in standard Grid Ascii format, and have the same dimensions as the Grid Ascii map of DEM data that the simulation will use. For this example, each cell value in the Grid Ascii map file is assumed to be an amount of leaf biomass (in grams per square meter).

Add the New Disturbance Parameters to the Simulation Configuration

1. In JVelma, with the simulation configuration loaded, Click/open the “Edit” menu and Click/Select “Disturbances “ -> “Add a Disturbance”.
2. In the Disturbance Model Selector dialog, use the drop-down selector to find the “SetSpatialDataByMapDisturbanceModel” and Click/Set it as the Disturbance Type.

[1] Two or more Set Spatial Data By Map disturbances may be set to use the same Grid Ascii map file as their input data source, but it’s more likely that a given spatial data pool to have its own unique input data source.

3. In the text box next to the “DisturbanceName” prompt, type in a name for this particular disturbance. The name should be unique (the dialog will warn you if you enter a name already in use) and it is better to avoid whitespace and non-alphanumeric characters.
   For this is example, the name will be “Init_Leaf_Biomass”.
4. Click the “OK” button. The dialog closes, the parameters for a new Set Spatial Data By Map disturbance are added to the simulation configuration, and the view should shift to these new parameters in JVelma’s “All Parameters” tab window.

Configure the New Disturbance Parameters

The parameters for the Set Spatial Data By Map disturbance tell the disturbance when it should occur, What its data source is, and Which Spatial Data Pool to apply that data source to.

The `initializeActiveLoops`, `initializeActiveYears`” and `initializeActiveJdays` parameters, plus the `occursAtStepStart` parameter specify when this disturbance should occur. To initialize the leaf biomass pool before any simulation steps, set them as follows:

Parameter Name	**Set Value As …**	**Comment**
`initializeActiveLoops`	1	Occurs during the first loop
`initializeActiveYears`	2000	Assumes 2000 is the simulation’s start year
`initializeActiveJdays`	1	Occurs during the first Julian day
`occursAtStepStart`	true	Disturbance will occur before any simulation work is performed on the day when it occurs

Set the `spatialDataFileFullName` parameter to the fully-qualified path and name of the Grid Ascii file that you plan to use as the input data source for biomass leaf values.
For this example, suppose that file is named “LeafBiomassData.asc” and resides in directory “C:\My Data\”. The `spatialDataFileName` value should then be set to: “C:/My Data/LeafBiomassData.asc” (without the double-quotes).
Note
Instead of Windows backslash “\” path separators, use forward slash “/” characters as path separators. Also, be very careful with whitespace; if there is any, it must *exactly* match whatever whitespace is present in the actual path and filename.

Set the `spatialDataName` parameter to the Spatial Data Pool that will receive data from the file.
For our example of leaf biomass, the pool name is “BIOMASS_LEAF_N” (without the double-quotes).
Note
A table containing the names of the Velma Simulator’s spatial data pools is available at the end of this document.

Set the `spatialDataLayer` parameter to the (one-based) layer index number of the spatial data pool layer that will receive data from the file.

For single-layer spatial data pools this value MUST be set to "1" see the table of spatial data at the end of this document for a summary of how many layers each pool has.

In summary, here is the full parameterization for our example Set (Leaf Biomass) Spatial Data Pool By Map disturbance:

ID Key	Value
`/disturbance/Init_Leaf_Biomass/initializeActiveJdays`	`1`
`/disturbance/Init_Leaf_Biomass/initializeActiveLoops`	`1`
`/disturbance/Init_Leaf_Biomass/initializeActiveYears`	`2000`
`/disturbance/Init_Leaf_Biomass/modelClass`	`SetSpatialDataByMapDisturbanceModel`
`/disturbance/Init_Leaf_Biomass/occursAtStepStart`	`true`
`/disturbance/Init_Leaf_Biomass/spatialDataFileFullName`	`C:/My Data/LeafBiomassData.asc`
`/disturbance/Init_Leaf_Biomass/spatialDataLayer`	`1`
`/disturbance/Init_Leaf_Biomass/spatialDataName`	`BIOMASS_LEAF_N`

Configured as specified by the parameters above, this `SetSpatialDataByMapDisturbanceModel` disturbance occurs once during the simulation; in the first loop, on the first day of 2000 (assumed to be the first year of simulation for our example) and before any simulation work is done on that day. When the disturbance occurs, it will open the Grid Ascii spatial data file "C:/My Data/LeafBiomassdata.asc", and set each cell of the BIOMASS_LEAF_N spatial data pool in layer 1 to the value for that cell read from the .asc spatial data file.

That's all there is to it!
Although `SetSpatialDataByMapDisturbanceModel` disturbances may be used for other purposes, by varying when they occur, this example shows the commonest use-case; initializing specific non-uniform values into a spatial data pool at the start of a simulation.

Caveat

There is no cross-checking of user-specified `SetSpatialDataByMapDisturbanceModel` disturbances. This means that it is possible to configure a simulation with multiple `SetSpatialDataByMapDisturbanceModel` disturbances targeting a single spatial data pool. In such circumstances, all multiple maps' values are applied to a single pool in succession, and the last map's values are the pool's "initialization". However, it is not possible to specify the sequence for the application of the multiple maps; hence what the pool is actually initialized to is ill-defined.

Cross-checking is automatically performed when the simulator's setStartStateSpatialDataFullDir parameter is specified, but under no other conditions.

The user, is therefore responsible for confirming that a collection of `SetSpatialDataByMapDisturbanceModel` distrubances do not "step on each other" with regard to the spatial data pools they modify.

A Partial List of Velma Simulator Spatial Data Pools (*see Tutorial D.8 for full list*)

Spatial Data Pool Name	Layers	Unit Type
BIOMASS_AG_STEM_N	1	N g/m^2
BIOMASS_BG_STEM_N	1	N g/m^2
BIOMASS_DELTA_AG_STEM_N	1	N g/m^2
BIOMASS_DELTA_BG_STEM_N	1	N g/m^2
BIOMASS_DELTA_LEAF_N	1	N g/m^2
BIOMASS_DELTA_ROOT_N	1	N g/m^2
BIOMASS_HARVESTED_TO_OFFSITE_C	1	C g/m^2
BIOMASS_LEAF_N	1	N g/m^2
BIOMASS_ROOT_N	4	N g/m^2
CO2	4	Carbon
DENITRIFICATION	4	N g/m^2
DETRITUS_AG_STEM_N	1	N g/m^2
DETRITUS_BG_STEM_N	4	N g/m^2
DETRITUS_BURNED_C	1	C g/m^2
DETRITUS_LEAF_N	1	N g/m^2
DETRITUS_ROOT_N	4	N g/m^2
DOC	4	C g/m^2
DON	4	N g/m^2
GROUND_SURFACE_TEMPERATURE	1	Degrees C
GROUND_TEMPERATURE_LAYERS	4	Degrees C
HUMUS	4	N g/m^2
NH4	4	N g/m^2
NITRIFICATION	4	N g/m^2
NO3	4	N g/m^2
NPP_C	1	C g/m^2
NPP_N	1	N g/m^2
UPTAKE_N	4	N g/m^2

D.4 | Create Initial Soil and Plant Chemistry Spatial (ASCII Grid) Data Pools

Overview *(Tutorial D.4 – Create Initial Spatial Soil and Plant Chemistry Pools)*

VELMA simulates responses of 58 soil and plant chemistry pools to changes in climate, land use and land cover. This document explains how to initialize the sizes (nitrogen or carbon grams/m^2) of these pools to reflect conditions at the beginning of a simulation. It is up to the user to obtain the spatial data describing those initial conditions. For example, data describing the amount of forest aboveground biomass across a landscape 30-meter grid can be obtained from remote sensing sources such as http://geotrendr.ceoas.oregonstate.edu/landtrendr/, or https://www.sciencedaily.com/releases/2011/04/110420125458.htm.

Initial chemistry or nutrient pools are created through a series of transformations of above ground biomass to each of the respective spatial and chemistry data pools. These transformations are defined by relationship ratios (fractions) each pool has to “real” or estimated measurements of above ground carbon, gC/m^2 for the study area. Optionally, a cover age file can be used to create these same chemistry pool ASCII rasters from the ratio lookup table.

58 VELMA Simulator spatial data pools will be created from this one raster grid biomass layer (Table 1 & 2). See Table 1 for the complete list of initial layers that will be created, and Table 2 for the C:N ratio pools created. Each spatial data pool layer will have its own corresponding Grid Ascii (.asc) map file upon completion of the Python script. These Grid Ascii (.asc) map files will be used as inputs to VELMA for initializing the map disturbance events.

Software Requirements

To run and produce the processed permeability fraction map (ASCII) from the “roads” or impervious surface layer (ASCII), you will need the following:

D. Python version 2.7: Current release is 2.7.11
    1. Python comes pre-packaged within ArcGIS, so it is likely you already have it installed on your computer. For example, a default install is currently here: C:\Python27\ArcGIS10.2\python.exe. Check to see if you have Python installed before installing a new version. If Python is not installed, you can obtain a copy here: https://www.python.org/download/releases/2.7/ *Note that Python 2.7 is currently considered safe for use on U.S. EPA network and non-network computers.*
    2. NumPy package installed, for the Python version

File Requirements for Data Processing

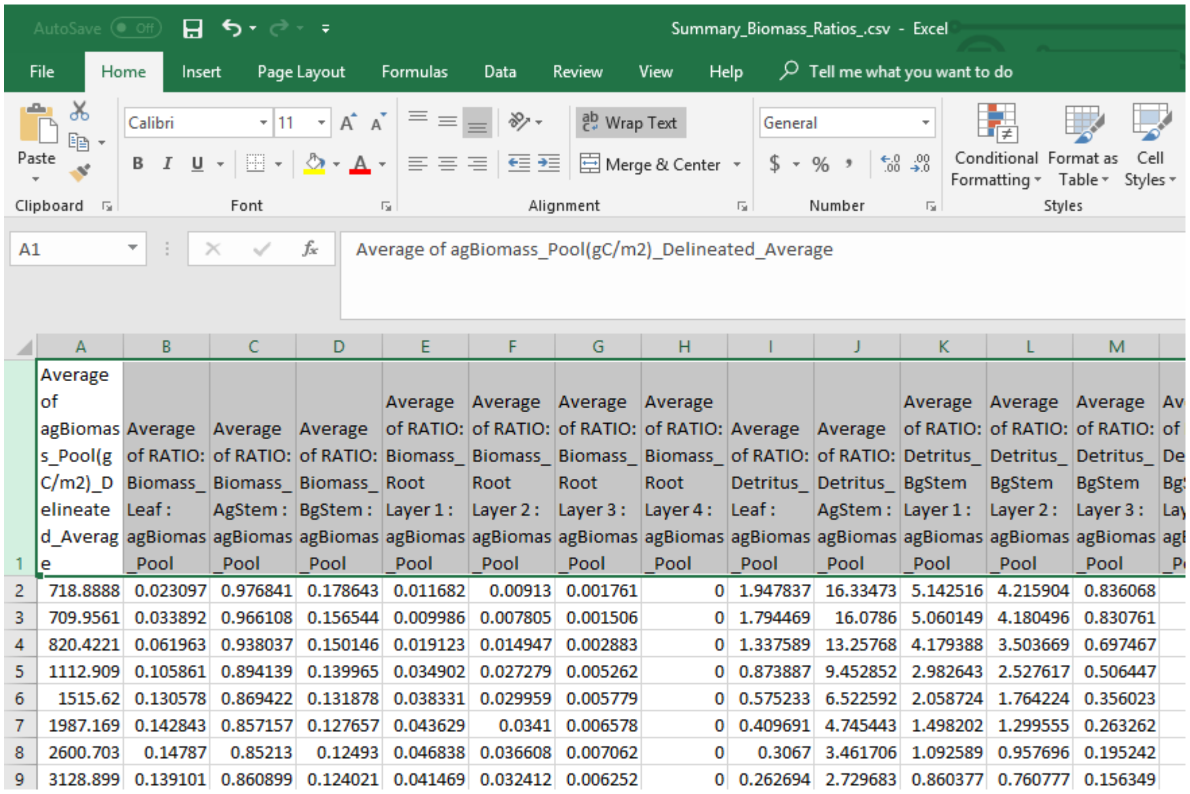
1. ASCII Map requirements:
    - Above ground carbon g / m² ASCII (.asc) raster that matches the VELMA modeling DEM to the exact extent, row and column count, and cell size. (Optional, if using an age map)
    - Age ASCII raster that matches the VELMA modeling DEM to the exact extent, row and column count, and cell size. (Optional, if using a biomass map)
2. Biomass ratio look up file which contains all the pool attributes seen in table 1.
    - Pool ratios values in subsequent columns (quantity 37) (Table 1).
    - Biomass values (stand age / mass) is in the first column (Figure 1).
    - Header and Column arrangement must match order in Table 1.
    - Saved in “.csv” comma delaminate format
3. Conversion driver file that relates ratio look up file column inputs to output names and additional C:N nitrogen conversion pools.
    - Pool ratios values in subsequent columns (quantity 37) (Table 1) and the nitrogen conversion pools (quantity 21) (Table 2)
    - Column descriptions (Figure 2 and 3):
        - **i.** Outfile_Pool = The output file name for the ratio pool
        - **ii.** Biomass_Ratio_Type = Spatial data pool type
        - **iii.** Column_Number = The column number associate with the spatial data pool type.
        - **iv.** Conversion_output = “FALSE” if the spatial data pool type is initial pool, “TRUE” if it a C:N nitrogen conversion.
        - **v.** C_to_N_Associate_File = The filename of the spatial pool type to be converted to the nitrogen spatial pool by the C:N ratio.
        - **vi.** C_to_N_Ratio = The C:N ratio that is divided into the carbon spatial pool to create the nitrogen pool
    - Header and Column arrangement must match the order in Figure 2 and 3.
    - Saved in “.csv” comma delaminate format
4. Maximum age (years) or above ground (gC / m²) (Optional, default to 400 years or 62000 gC / m²)
5. Minimum age (years) or above ground (gC / m²) (Optional, default to 1 years or 400 gC / m²)

Table 1. Column ordering of the biomass ratio pool attributes file. Unless noted by it's chemistry, all units are grams carbon / m². Highlighted pools are created from relationships to the first column, total above ground carbon.

Column Letter	Pool Attribute
A	Average of agBiomass_Pool(gC/m2)_Delineated_Average
B	Average of RATIO: Biomass_Leaf : agBiomas_Pool
C	Average of RATIO: Biomass_AgStem : agBiomas_Pool
D	Average of RATIO: Biomass_BgStem : agBiomas_Pool
E	Average of RATIO: Biomass_Root Layer 1 : agBiomas_Pool
F	Average of RATIO: Biomass_Root Layer 2 : agBiomas_Pool
G	Average of RATIO: Biomass_Root Layer 3 : agBiomas_Pool
H	Average of RATIO: Biomass_Root Layer 4 : agBiomas_Pool
I	Average of RATIO: Detritus_Leaf : agBiomas_Pool
J	Average of RATIO: Detritus_AgStem : agBiomas_Pool
K	Average of RATIO: Detritus_BgStem Layer 1 : agBiomas_Pool
L	Average of RATIO: Detritus_BgStem Layer 2 : agBiomas_Pool
M	Average of RATIO: Detritus_BgStem Layer 3 : agBiomas_Pool
N	Average of RATIO: Detritus_BgStem Layer 4 : agBiomas_Pool
O	Average of RATIO: Detritus_Root Layer 1 : agBiomas_Pool
P	Average of RATIO: Detritus_Root Layer 2 : agBiomas_Pool
Q	Average of RATIO: Detritus_Root Layer 3 : agBiomas_Pool
R	Average of RATIO: Detritus_Root Layer 4 : agBiomas_Pool
S	Average of RATIO: NH4_Pool_Layer 1 : agBiomas_Pool
T	Average of RATIO: NH4_Pool_Layer 2 : agBiomas_Pool
U	Average of RATIO: NH4_Pool_Layer 3 : agBiomas_Pool
V	Average of RATIO: NH4_Pool_Layer 4 : agBiomas_Pool
W	Average of RATIO: NO3_Pool_Layer 1 : agBiomas_Pool
X	Average of RATIO: NO3_Pool_Layer 2 : agBiomas_Pool
Y	Average of RATIO: NO3_Pool_Layer 3 : agBiomas_Pool
Z	Average of RATIO: NO3_Pool_Layer 4 : agBiomas_Pool
AA	Average of RATIO: DON_Pool_Layer 1 : agBiomas_Pool
AB	Average of RATIO: DON_Pool_Layer 2 : agBiomas_Pool
AC	Average of RATIO: DON_Pool_Layer 3 : agBiomas_Pool
AD	Average of RATIO: DON_Pool_Layer 4 : agBiomas_Pool
AE	Average of RATIO: DOC_Pool_Layer 1 : agBiomas_Pool
AF	Average of RATIO: DOC_Pool_Layer 2 : agBiomas_Pool
AG	Average of RATIO: DOC_Pool_Layer 3 : agBiomas_Pool
AH	Average of RATIO: DOC_Pool_Layer 4 : agBiomas_Pool
AI	Average of RATIO: Humus_Layer1 : agBiomas_Pool
AJ	Average of RATIO: Humus_Layer2 : agBiomas_Pool
AK	Average of RATIO: Humus_Layer3 : agBiomas_Pool
AL	Average of RATIO: Humus_Layer4 : agBiomas_Pool

Table 2. Additional nitrogen spatial pools derived from C:N ratios of above ground carbon pools.

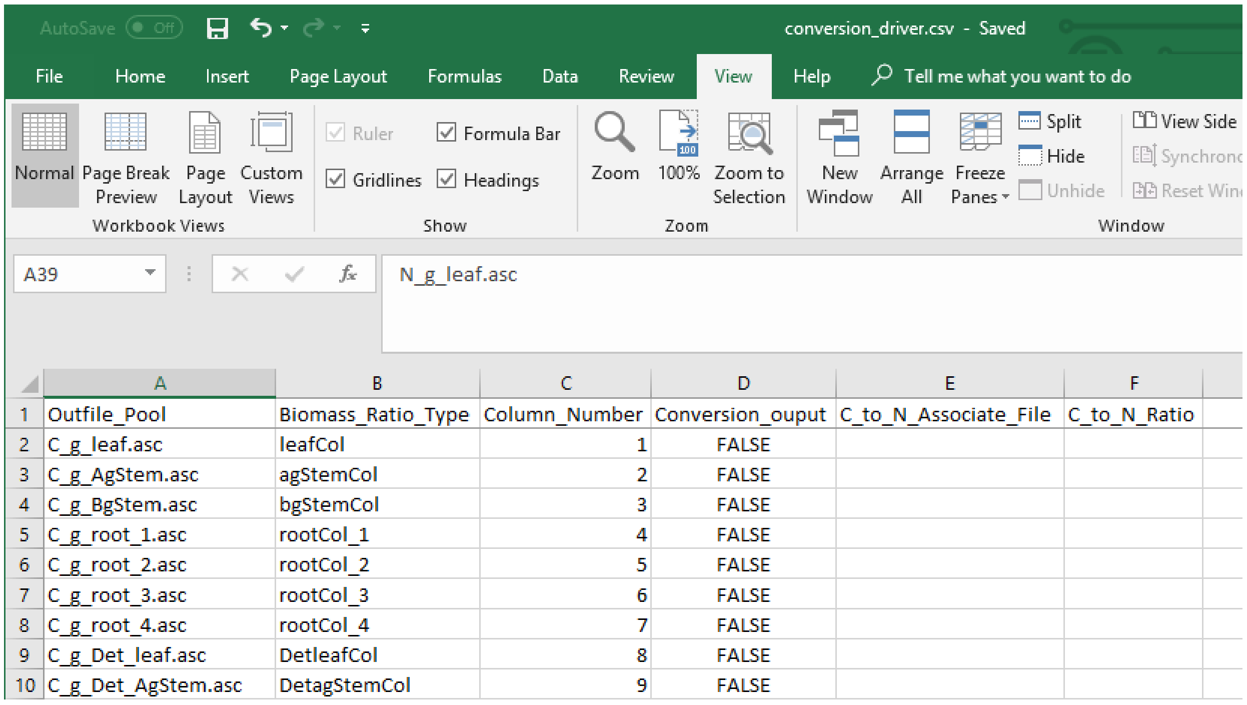
Spatial Data Pool Name	Unit Type
N_g_leaf.asc	N g/m^2
N_g_AgStem.asc	N g/m^2
N_g_BgStem.asc	N g/m^2
N_g_root_1.asc	N g/m^2
N_g_root_2.asc	N g/m^2
N_g_root_3.asc	N g/m^2
N_g_root_4.asc	N g/m^2
N_g_Det_leaf.asc	N g/m^2
N_g_Det_AgStem.asc	N g/m^2
N_g_Det_BgStem_1.asc	N g/m^2
N_g_Det_BgStem_2.asc	N g/m^2
N_g_Det_BgStem_3.asc	N g/m^2
N_g_Det_BgStem_4.asc	N g/m^2
N_g_Det_root_1.asc	N g/m^2
N_g_Det_root_2.asc	N g/m^2
N_g_Det_root_3.asc	N g/m^2
N_g_Det_root_4.asc	N g/m^2
N_g_Humus_1.asc	N g/m^2
N_g_Humus_2.asc	N g/m^2
N_g_Humus_3.asc	N g/m^2
N_g_Humus_4.asc	N g/m^2


Summary_Biomass_Ratios_.csv - Excel
File
Home
Insert
Page Layout
Formulas
Data
Review
View
Help
Tell me what you want to do

A1: Average of agBiomass_Pool(gC/m2)_Delineated_Average

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Average of agBiomass_Pool(gC/m2)_Delineated_Average	Average of RATIO: Biomass_Leaf : agBiomass_Pool	Average of RATIO: Biomass_AgStem : agBiomass_Pool	Average of RATIO: Biomass_BgStem : agBiomass_Pool	Average of RATIO: Biomass_Root Layer 1 : agBiomass_Pool	Average of RATIO: Biomass_Root Layer 2 : agBiomass_Pool	Average of RATIO: Biomass_Root Layer 3 : agBiomass_Pool	Average of RATIO: Biomass_Root Layer 4 : agBiomass_Pool	Average of RATIO: Detritus_Leaf : agBiomass_Pool	Average of RATIO: Detritus_AgStem : agBiomass_Pool	Average of RATIO: Detritus_BgStem Layer 1 : agBiomass_Pool	Average of RATIO: Detritus_BgStem Layer 2 : agBiomass_Pool	Average of RATIO: Detritus_BgStem Layer 3 : agBiomass_Pool
2	718.8888	0.023097	0.976841	0.178643	0.011682	0.00913	0.001761	0	1.947837	16.33473	5.142516	4.215904	0.836068
3	709.9561	0.033892	0.966108	0.156544	0.009986	0.007805	0.001506	0	1.794469	16.0786	5.060149	4.180496	0.830761
4	820.4221	0.061963	0.938037	0.150146	0.019123	0.014947	0.002883	0	1.337589	13.25768	4.179388	3.503669	0.697467
5	1112.909	0.105861	0.894139	0.139965	0.034902	0.027279	0.005262	0	0.873887	9.452852	2.982643	2.527617	0.506447
6	1515.62	0.130578	0.869422	0.131878	0.038331	0.029959	0.005779	0	0.575233	6.522592	2.058724	1.764224	0.356023
7	1987.169	0.142843	0.857157	0.127657	0.043629	0.0341	0.006578	0	0.409691	4.745443	1.498202	1.299555	0.263262
8	2600.703	0.14787	0.85213	0.12493	0.046838	0.036608	0.007062	0	0.3067	3.461706	1.092589	0.957696	0.195242
9	3128.899	0.139101	0.860899	0.124021	0.041469	0.032412	0.006252	0	0.262694	2.729683	0.860377	0.760777	0.156349

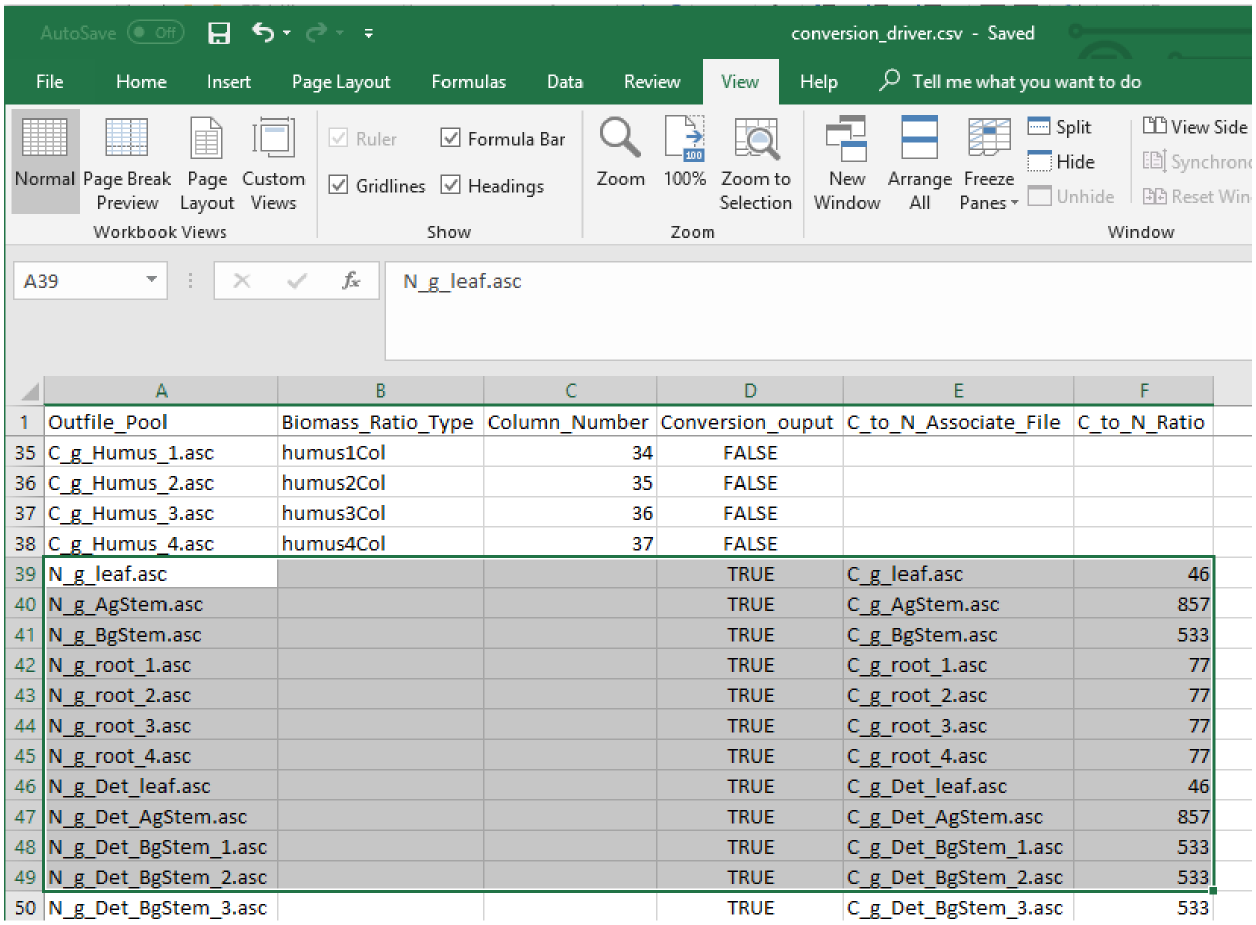
Figure 6. Summary Biomass Ratio Look up file, .csv



A39: N_g_leaf.asc

	A	B	C	D	E	F
1	Outfile_Pool	Biomass_Ratio_Type	Column_Number	Conversion_ouput	C_to_N_Associate_File	C_to_N_Ratio
2	C_g_leaf.asc	leafCol	1	FALSE		
3	C_g_AgStem.asc	agStemCol	2	FALSE		
4	C_g_BgStem.asc	bgStemCol	3	FALSE		
5	C_g_root_1.asc	rootCol_1	4	FALSE		
6	C_g_root_2.asc	rootCol_2	5	FALSE		
7	C_g_root_3.asc	rootCol_3	6	FALSE		
8	C_g_root_4.asc	rootCol_4	7	FALSE		
9	C_g_Det_leaf.asc	DetleafCol	8	FALSE		
10	C_g_Det_AgStem.asc	DetagStemCol	9	FALSE		

Figure 7. Conversion diver file



N_g_leaf.asc

	A	B	C	D	E	F
1	Outfile_Pool	Biomass_Ratio_Type	Column_Number	Conversion_ouput	C_to_N_Associate_File	C_to_N_Ratio
35	C_g_Humus_1.asc	humus1Col	34	FALSE		
36	C_g_Humus_2.asc	humus2Col	35	FALSE		
37	C_g_Humus_3.asc	humus3Col	36	FALSE		
38	C_g_Humus_4.asc	humus4Col	37	FALSE		
39	N_g_leaf.asc			TRUE	C_g_leaf.asc	46
40	N_g_AgStem.asc			TRUE	C_g_AgStem.asc	857
41	N_g_BgStem.asc			TRUE	C_g_BgStem.asc	533
42	N_g_root_1.asc			TRUE	C_g_root_1.asc	77
43	N_g_root_2.asc			TRUE	C_g_root_2.asc	77
44	N_g_root_3.asc			TRUE	C_g_root_3.asc	77
45	N_g_root_4.asc			TRUE	C_g_root_4.asc	77
46	N_g_Det_leaf.asc			TRUE	C_g_Det_leaf.asc	46
47	N_g_Det_AgStem.asc			TRUE	C_g_Det_AgStem.asc	857
48	N_g_Det_BgStem_1.asc			TRUE	C_g_Det_BgStem_1.asc	533
49	N_g_Det_BgStem_2.asc			TRUE	C_g_Det_BgStem_2.asc	533
50	N_g_Det_BgStem_3.asc			TRUE	C_g_Det_BgStem_3.asc	533

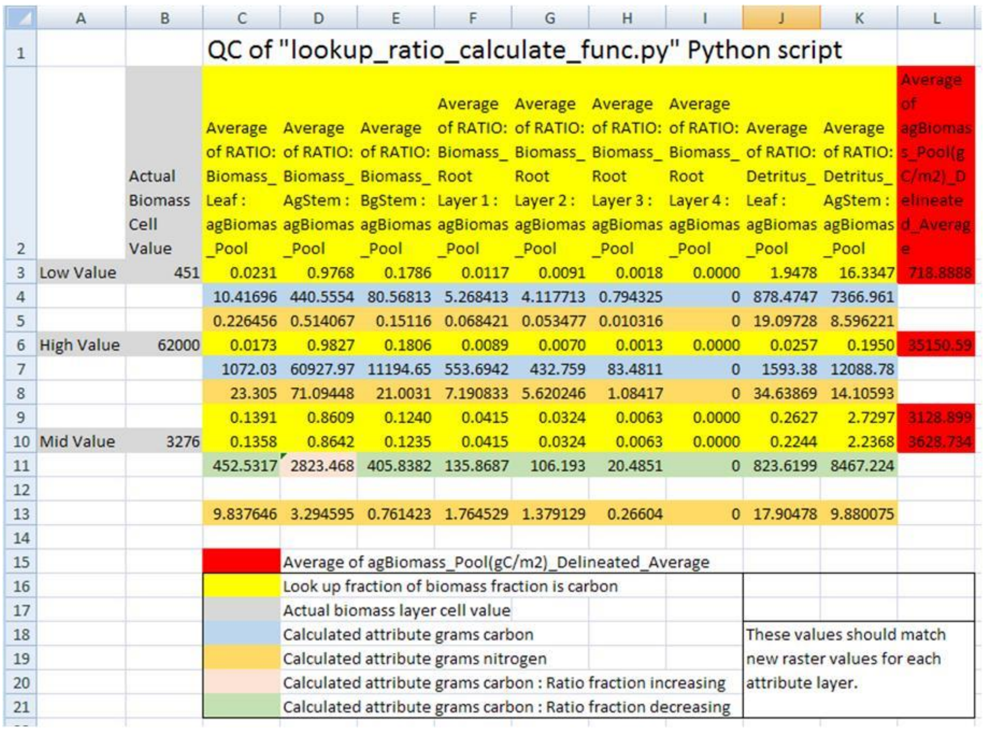
Figure 8. Conversion driver file

Processing

Conversion of biomass (g / m²) to the final spatial pools and nitrogen pools (g / m²) are made through an intermediate step of converting mass to carbon (g / m²) pools, and then using the conversion driver file, different C to N ratios for each pool are used to calculate the final nitrogen pools. Fractions (ratios) of spatial pools change both in the positive and negative direction as a forest ages or increases in total biomass. Ratio pools can increase or decrease between stand age increments, and the calculations of these intermediate biomass observational values must reflect this dynamic nature of changing ratios between time steps. Intermediate values of biomass, those that land in between time step years, were therefore linearly interpolated between the time point bounds. Spatial map ascii grid values for each of the 37 pools should then backward match ratio conversions and final C to N calculations to the original "observed" above ground biomass. (Only a partial list of pools is displayed here).

When using age maps instead of carbon biomass maps, spatial pools are created from the year and row number each cell corresponds too. If a cell is 1 year old, row 1 values would be used, while a 400-year-old cell relates to row 400.

Example: A mid biomass cell value at B10 = 3276 (g / m²) and is part way between stand age values at L9 and L10. It's above ground stem carbon mass fraction increases from 0.8609 (D9) to 0.8642 (D10). So, its fraction stem mass (column D) should increase as its percentage difference moves from L9 towards LD10. Conversely, for all the other pool attributes, its percentage difference would increase as its fraction moves the opposite direction from rows 10 to rows 9.



	A	B	C	D	E	F	G	H	I	J	K	L
1			QC of "lookup_ratio_calculate_func.py" Python script									
2		Actual Biomass Cell Value	Average of RATIO: Biomass_ Leaf : agBiomas _Pool	Average of RATIO: Biomass_ AgStem : agBiomas _Pool	Average of RATIO: Biomass_ BgStem : agBiomas _Pool	Average of RATIO: Biomass_ Root Layer 1 : agBiomas _Pool	Average of RATIO: Biomass_ Root Layer 2 : agBiomas _Pool	Average of RATIO: Biomass_ Root Layer 3 : agBiomas _Pool	Average of RATIO: Biomass_ Root Layer 4 : agBiomas _Pool	Average of RATIO: Detritus_ Leaf : agBiomas _Pool	Average of RATIO: Detritus_ AgStem : agBiomas _Pool	Average of agBiomass_Pool(gC/m2)_Delineated_Average
3	Low Value	451	0.0231	0.9768	0.1786	0.0117	0.0091	0.0018	0.0000	1.9478	16.3347	718.8888
4			10.41696	440.5554	80.56813	5.268413	4.117713	0.794325	0	878.4747	7366.961	
5			0.226456	0.514067	0.15116	0.068421	0.053477	0.010316	0	19.09728	8.596221	
6	High Value	62000	0.0173	0.9827	0.1806	0.0089	0.0070	0.0013	0.0000	0.0257	0.1950	35150.59
7			1072.03	60927.97	11194.65	553.6942	432.759	83.4811	0	1593.38	12088.78	
8			23.305	71.09448	21.0031	7.190833	5.620246	1.08417	0	34.63869	14.10593	
9			0.1391	0.8609	0.1240	0.0415	0.0324	0.0063	0.0000	0.2627	2.7297	3128.899
10	Mid Value	3276	0.1358	0.8642	0.1235	0.0415	0.0324	0.0063	0.0000	0.2244	2.2368	3628.734
11			452.5317	2823.468	405.8382	135.8687	106.193	20.4851	0	823.6199	8467.224	
12												
13			9.837646	3.294595	0.761423	1.764529	1.379129	0.26604	0	17.90478	9.880075	
14												
15			🟥	Average of agBiomass_Pool(gC/m2)_Delineated_Average								
16			🟨	Look up fraction of biomass fraction is carbon								
17			⬜	Actual biomass layer cell value								
18			🟦	Calculated attribute grams carbon						These values should match		
19			🟧	Calculated attribute grams nitrogen						new raster values for each		
20			▨	Calculated attribute grams carbon : Ratio fraction increasing						attribute layer.		
21			🟩	Calculated attribute grams carbon : Ratio fraction decreasing								

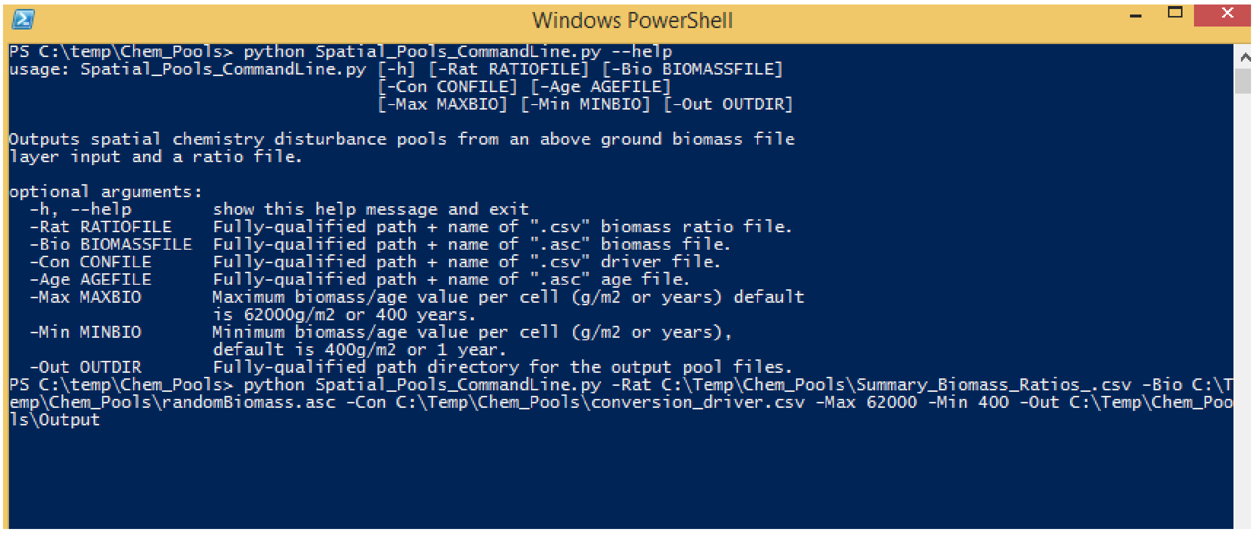
Figure 9. Quality control check of Python script's dynamic conversions of "observed" above ground biomass (g / m²) to the final nitrogen (g / m²) spatial data pools.

The "**`Spatial_Pools_CommandLine.py`**" and can be run on the Command Prompt line with Python. This script requires at a minimum, one biomass or one age surface map, a ratio look up file, a conversion driver file, and one output directory name. Optionally, a minimum and maximum age or grams carbon value can be provided.

Windows PowerShell Command prompt input example:

```
"python C:\Path\To\Spatial_Pools_CommandLine.py -Rat
C:\Temp\Chem_Pools\Summary_Biomass_Ratios_.csv -Bio
C:\Temp\Chem_Pools\randomBiomass.asc -Con C:\Temp\Chem_Pools\conversion_driver.csv -
Max 62000 -Min 400 -Out C:\Temp\Chem_Pools\Output"
```

"`python C:\Path\To\Spatial_Pools_CommandLine.py –help`" for exact arguments (Figure 5).



```
PS C:\temp\Chem_Pools> python Spatial_Pools_CommandLine.py --help
usage: Spatial_Pools_CommandLine.py [-h] [-Rat RATIOFILE] [-Bio BIOMASSFILE]
                                    [-Con CONFILE] [-Age AGEFILE]
                                    [-Max MAXBIO] [-Min MINBIO] [-Out OUTDIR]

Outputs spatial chemistry disturbance pools from an above ground biomass file
layer input and a ratio file.

optional arguments:
  -h, --help        show this help message and exit
  -Rat RATIOFILE    Fully-qualified path + name of ".csv" biomass ratio file.
  -Bio BIOMASSFILE  Fully-qualified path + name of ".asc" biomass file.
  -Con CONFILE      Fully-qualified path + name of ".csv" driver file.
  -Age AGEFILE      Fully-qualified path + name of ".asc" age file.
  -Max MAXBIO       Maximum biomass/age value per cell (g/m2 or years) default
                    is 62000g/m2 or 400 years.
  -Min MINBIO       Minimum biomass/age value per cell (g/m2 or years),
                    default is 400g/m2 or 1 year.
  -Out OUTDIR       Fully-qualified path directory for the output pool files.
PS C:\temp\Chem_Pools> python Spatial_Pools_CommandLine.py -Rat C:\Temp\Chem_Pools\Summary_Biomass_Ratios_.csv -Bio C:\T
emp\Chem_Pools\randomBiomass.asc -Con C:\Temp\Chem_Pools\conversion_driver.csv -Max 62000 -Min 400 -Out C:\Temp\Chem_Poo
ls\Output
```

Figure 10 Window PowerShell

Note

Processing time is very dependent on processing power and the size of your study area. For example, the Blue River, in Oregon, had a study grid of ~350,000 cells and took ~10 minutes for each of the 37 pools, for a total of 6-8 hours. The Python shell will print updates of pool processing steps, and say “All Done!” when it’s done.

* Creating a notepad text file may make coping and pasting the prompt input easier.

Scripts

```
#  Command line for gathering info for processing and generating initial spatial pools
#  9-11-2018
#  Author: Paul Pettus
#  Spatial_Pools_CommandLine.py
#

import time
import csv
import os, sys, argparse
import numpy
import linecache

# Error message class
class Usage(Exception):
    def __init__(self, msg):
        self.msg = msg

# Main class
def main(argv=None):

    if argv is None:
        argv = sys.argv

    try:
        # Describe the tool and required aruments
        parser = argparse.ArgumentParser(description='Outputs spatial chemistry disturbance pools
from an'+
                                          ' above ground biomass file layer input and a ratio
file.')

        # Ratio file input
        parser.add_argument('-Rat', action='store', dest='ratioFILE',
                            help='Fully-qualified path + name of ".csv" biomass ratio file.')

        # Biomass ascii file input
        parser.add_argument('-Bio', action='store', dest='biomassFILE',
                            help='Fully-qualified path + name of ".asc" biomass file.')

        # Biomass ascii file input
        parser.add_argument('-Con', action='store', dest='conFILE',
                            help='Fully-qualified path + name of ".csv" driver file.')

        # Age ascii file input
        parser.add_argument('-Age', action='store', dest='ageFILE',
                            help='Fully-qualified path + name of ".asc" age file.')

        # Maximum biomass value per cell
        parser.add_argument('-Max', action='store', dest='maxBio',
                            help='Maximum biomass/age value per cell (g/m2 or years) default is
62000g/m2 or 400 years.')

        # Minimum biomass value per cell
        parser.add_argument('-Min', action='store', dest='minBio',
                            help='Minimum biomass/age value per cell (g/m2 or years), default is
400g/m2 or 1 year.')

### Minimum biomass value per cell
##        parser.add_argument('-Min', action='store', metavar='minBio',
##                            type=int,
##                            help='Minimum biomass/age value per cell (g/m2 or years), default
is 400g/m2 or 1 year.')

        # Output dir for spatial chemistry pools
        parser.add_argument('-Out', action='store', dest='outDIR',
                            help='Fully-qualified path directory for the output pool files.')

        args = parser.parse_args()
        # args parsing
        if args.ratioFILE == None:
            ratioFile = args.ratioFILE
        else:
            ratioFile = str(os.path.abspath(args.ratioFILE))

        if args.biomassFILE == None:
            biomassFile = args.biomassFILE
        else:
            biomassFile = str(os.path.abspath(args.biomassFILE))

        if args.conFILE == None:
            conFile = args.conFILE
        else:
            conFile = str(os.path.abspath(args.conFILE))

        if args.ageFILE == None:
            ageFile = args.ageFILE
        else:
            ageFile = str(os.path.abspath(args.ageFILE))

        outDir = str(os.path.abspath(args.outDIR)) + "\\"

        if args.minBio == None:
            minValArg = args.minBio
        else:
            minValArg = int(args.minBio)

        if args.maxBio == None:
            maxValArg = args.maxBio
        else:
            maxValArg = int(args.maxBio)

                # Print all input arguments
        print('\n' + '\n' + "Echo inputs" + '\n')
##        print(args)
##        print('\n')

        print(outDir, ratioFile, biomassFile, conFile, ageFile, minValArg, maxValArg)

        # do the work function
        buildRatios(outDir, ratioFile, biomassFile, conFile, ageFile, minValArg, maxValArg)

    except Usage as e:
        print(e.msg)
        return 2

    except Exception as e:
        # STUB exception handler
        # Warning: poor programming style.
        # Catches almost any exception (but not KeyboardInterrupt -- which is a Good Thing)
        raise e

## Return an numpy array from an ascii raster
def ascii2array(asciifn):
#### GDAL version
##    dsAscii = gdal.Open(asciifn,gdalconst.GA_ReadOnly)
##    cols = dsAscii.RasterXSize
##    rows = dsAscii.RasterYSize
##    bands = dsAscii.RasterCount
##    band = dsAscii.GetRasterBand(1)
    array = numpy.loadtxt(asciifn, skiprows=6, dtype= numpy.float32)
    return array

## Return and ascii raster file from an numpy array, output file name
## (optional inputs are an alternative ascii area of interest and or
## an optional header text: these are not used in the script
def array2ascii(array, asciiOutfn, asciiaoi = None, headerList = None):

    # set up header text for the output ascii raster
    if asciiaoi == None:
##        for i in headerList:
        header = headerList
    else:
        header = linecache.getline(asciiaoi,1)
        header += linecache.getline(asciiaoi,2)
        header += linecache.getline(asciiaoi,3)
        header += linecache.getline(asciiaoi,4)
        header += linecache.getline(asciiaoi,5)
        header += linecache.getline(asciiaoi,6)

    # open the output file and write the header text and numpy array
    # and save the output file
    f = open(asciiOutfn, "w")
    f.write(header)
    numpy.savetxt(f, array, fmt="%f")
    f.close()

## return the header information of an ascii raster
def readHeader(asciiFile):

    # return error if the ascii is not found
    if not os.path.exists(asciiFile):
        raise Usage('Cannot find ASCII "' + asciiFile + '"')

    # Open file and read in header info
    readFile = open(asciiFile)

    header = readFile.readline()  #ncols
    header += readFile.readline() #nrows
    header += readFile.readline() #xllcorner
    header += readFile.readline() #yllcorner
    header += readFile.readline() #cellsize
    header += readFile.readline() #NODATA_value
    readFile.close()

    return header

# Do the work class, build the ratio files
def buildRatios(finalPoolDir, ratioFile, biomassFile, conFile, ageFile, minValArg, maxValArg):

    ##################################################################################
    # Start preprocess code block
    ##################################################################################

    # return error if the ratio file is not found
    if not os.path.exists(ratioFile):
        raise Usage('Cannot find ratio file "' + ratioFile + '"')

    # return error if both the age or biomass asciis are arguments
    if ageFile != None and biomassFile != None:
        print(ageFile)
        print(biomassFile)
        raise Usage('Please choose either a biomass or an age ASCII raster, not both.')

    # return error if the age or biomass ascii is not found
    if ageFile == None:
        # return error if the biomass ascii is not found
        print("Calculating chemistry pool rasters from biomass raster."+ '\n')
##        print("hello")
##        print(biomassFile)
        if not os.path.exists(biomassFile):
            raise Usage('Cannot find biomass ASCII "' + biomassFile + '\n')
    else:
        print("Calculating chemistry pool rasters from age raster."+ '\n')
        if not os.path.exists(ageFile):
            raise Usage('Cannot find age ASCII "' + ageFile + '"')

##    # return error if the biomass ascii is not found
##    if not os.path.exists(biomassFile):
##        raise Usage('Cannot find conversion driver file "' + conFile + '"')
```

```
# create the output dir if it doesn't exist
if not os.path.exists(os.path.dirname(finalPoolDir)):
    print("Final outpur dir not found, creating it."+ '\n')
    # Safety bachslash to slash replacement
    finalPoolDir.replace("\\",'/')
    os.makedirs(os.path.dirname(finalPoolDir))

# Set maximum and minimum biomass value
if biomassFile != None:
    print(biomassFile)
    # Set default maximum and minimum biomass value
    if maxValArg == None:
        maxValue = 62000
    else:
        maxValue = maxValArg

    if minValArg == None:
        minValue = 400
    else:
        minValue = minValArg

# Set maximum and minimum age value
if ageFile != None:

    # Set default maximum and minimum age value
    if maxValArg == None:
        maxValue = 400
    else:
        maxValue = maxValArg

    if minValArg == None:
        minValue = 0
    else:
        minValue = minValArg

    # Check to that maxValue isn't larger than length of ratio lookup
    ratioLookupArray = numpy.genfromtxt(ratioFile,skip_header=1, dtype=float, delimiter=',')
    rCount, cCount = ratioLookupArray.shape
    if int(maxValue) > int(rCount):
        raise Usage('Max age is larger than ratio file rows ' + str(rCount))

print(maxValue)
print(minValue)

# numpy method to read .csv to array
ratioLookupArray = numpy.genfromtxt(ratioFile,skip_header=1, dtype=float, delimiter=',')

# Read biomass or age raster to array
if ageFile != None:
    # Read age raster to array
    ageArray = ascii2array(ageFile)
    # get the ascii raster header information
    buildRatios.header = readHeader(ageFile)
else:
    # Read biomass to array
    biomassArray = ascii2array(biomassFile)
    # get the ascii raster header information
    buildRatios.header = readHeader(biomassFile)

# read through ratio conversion driver file
datafile = open(conFile, 'r')
next(datafile,None)
myreader = csv.reader(datafile)
# create ratio conversion to a list
driverList = list(myreader)
# close file
datafile.close()

##################################################################################
```

```
# End preprocess code block
##################################################################################

##################################################################################
# Start functions block
##################################################################################

# Function: Returns grams of carbon for the upper and lower most
# biomass lookup cells.  These two cells do not need to dynamically
# calculated.
# Inputs:
# upval = the lookup ratio
# cellmass = biomass cell value
# Output:
# gramsC = grams of carbon
def calLowHighper(upval,cellmass):
    gramsC = upval*cellmass
    return gramsC

# Function: Dynamically returns grams of carbon from the larger and smaller
# biomass lookup cells.  Creates fraction difference between a larger
# and smaller biomass value in the biomass lookup table, then applies that
# fraction value difference to the lookup attribute.
# Inputs:
# smallMass = smaller biomass from the velma attribute array
# largeMass = larger biomass from the velma attribute array
# upval = the lookup ratio that corresponds to the largeMass value
# lowval= the lookup ratio that corresponds to the smallMass value
# cellmass = biomass cell value
# Output:
# gramsC = grams of carbon
# Variables:
# difmass = difference bewteen the large and small biomass
# permass = fraction of cellmass that is over the smallMass
# difval = difference between upval and lowval
# deltaRatioValue = corresponding fraction of the lookup value
# newValue = the correct ratio of the the lookup value
# gramsC = grams of carbon
def calMidper(upval,lowval,largeMass,smallMass,cellmass):
    difmass = (largeMass-smallMass)
    permass = (cellmass-smallMass)/difmass
    difval = (upval-lowval)
    # If the difval is negative then lowval is larger than upval.
    if difval < 0:
        difval = (lowval-upval)
        deltaRatioValue = permass*difval
        newValue = (lowval-deltaRatioValue)
        gramsC = cellmass*newValue
        return gramsC
    elif difval == 0:
        gramsC = cellmass*upval
        return gramsC
    # Else: upval is larger than lowval
    else:
        deltaRatioValue = permass*difval
        newValue = deltaRatioValue+lowval
        gramsC = cellmass*newValue
        return gramsC

# Function: For each cell in biomass raster, lookup its biomass in the biomass ratio lookup
# table and calculate a new grams of carbon value.  Returns an array that can be turned back
# into a new raster for any inputed velma attribute.
# Inputs:
# biomassArray = input biomass array, from biomass raster
# ratioLookupArray = input biomass lookup array, attribute table
# column = column number of the attribute of interest
# Output:
# outArray = array with corresponding values of attribute calculated from biomass
# Variables:
# i = current row of biomass array
# j = current column of biomass array, from biomass raster
```

```
    # row = current row of biomass lookup array, attribute table
    # rowPlus = one row ahead of item or current row
    # cellValue = value of the current cell in the biomass array, from biomass raster
    # returnCell = grams of carbon returned from function call

    def biomassConverter(biomassArray,ratioLookupArray,column,maxValue,minValue):
        #creates numpy array with zeros  numpy.zeros()
        #array is the same size as biomass raster array
        rowArraySize, colArraySize = biomassArray.shape

        # create empty output array
        outArray = numpy.zeros( (rowArraySize,colArraySize) )

        rowRatioArray, colRatioArray = ratioLookupArray.shape

        # Loop through all the biomass raster values
        for i in xrange(rowArraySize):
            for j in xrange(colArraySize):
                # Loop through all of the biomass values in the lookup chem ratios
                for row in xrange(rowRatioArray):
                    # index for the next row
                    rowPlus = row + 1
                    cellValue = float(biomassArray[i,j])

                    # Makes sure no cell value is over the maximum biomass allowed
                    if cellValue > maxValue:
                        cellValue = float(maxValue)
                    # Make sure no cell value is under the minimum biomass allowed
                    if cellValue < minValue:
                        cellValue = float(minValue)

                    # Breaks loop if first element is in lookup chem ratios
                    if cellValue <= ratioLookupArray[row,0]:
                        returnCell = calLowHighper(ratioLookupArray[row,column],cellValue)
                        outArray[i,j] = returnCell

                        break
                    # Breaks loop if last element in lookup chem ratios
                    if row == (rowRatioArray-1):
                        returnCell = calLowHighper(ratioLookupArray[row,column],cellValue)
                        outArray[i,j] = returnCell
                        break

                    # Looks to see if the biomass cellvalue is inbetween the current lookup chem
ratio
                    # item and the next row lookup item
                    if cellValue > ratioLookupArray[row,0] and cellValue <=
ratioLookupArray[rowPlus,0]:
                        # Return value that is between the two lookup chem ratio rows
                        returnCell =
calMidper(ratioLookupArray[rowPlus,column],ratioLookupArray[row,column],

ratioLookupArray[rowPlus,0],ratioLookupArray[row,0],cellValue)
                        outArray[i,j] = returnCell
        # Return the chem pool array
        return outArray

    #creates numpy array with zeros  numpy.zeros()
    #array is the same size as biomass raster array
    def ageConverter(ageArray,ratioLookupArray,column,maxValue,minValue):

        clockStartBiomass = time.clock()

        rowArraySize, colArraySize = ageArray.shape

        outArray=numpy.zeros( (rowArraySize,colArraySize) )

        for i in xrange(rowArraySize):
            for j in xrange(colArraySize):
                yearRow = int(ageArray[i,j])
                # Change to the year to a max of 400 yo
                if yearRow > maxValue:
                    yearRow = maxValue

                if yearRow < minValue:
                    yearRow = minValue

                # the row number in the lookup ratio file corisponds to age (yr)
                biomassValue = ratioLookupArray[yearRow,0]
                ratioValue = ratioLookupArray[yearRow,column]
                cellValue = biomassValue * ratioValue
                outArray[i,j] = cellValue

        return outArray

    ################################################################################
    # End functions block
    ################################################################################

    print("Starting the 37 Biomass Ratio convertions!"+'\n')

    ################################################################################
    # Processing block code
    ################################################################################
    # Loop once through the driver list to create the carbon ratio ascii rasters first
    for i in driverList:

        # Get output file name
        outName = i[0]
        # Get column name, Unused
        colName = i[1]
        # Get ratio column number, check that it is not empty first
        # biomass ratio file column number associated with the attribute.
        # Empty sets are nitrogen conversion completed next step
        if i[2] != '':
            ratioCol = int(i[2])
        # Get conversion boolean, FALSE equals file is not a ratio conversion
        conversionOut = i[3]

        # Function call to calculate each of the attributes and create
        # a raster from the returned array
        if conversionOut == 'FALSE':

                # start time clock
            clockStartBiomass = time.clock()

            if ageFile != None:
                # For each ratio attribute column determin the ratio of biomass
                # associated with that particular chemestry pool and return an
                # array used to create chem pool raster
                ratioArray = ageConverter(ageArray,ratioLookupArray,ratioCol,maxValue,minValue)

            else:

                # For each ratio attribute column determin the ratio of biomass
                # associated with that particular chemestry pool and return an
                # array used to create chem pool raster
                ratioArray =
biomassConverter(biomassArray,ratioLookupArray,ratioCol,maxValue,minValue)

            # Concatinate output dir with output file name
            outName = finalPoolDir + outName
            # Create the chem pool raster
            array2ascii(ratioArray, outName, asciiaoi = None, headerList = buildRatios.header)

            # end time clock
            clockFinishBiomass = time.clock()
            biomassTime = round((clockFinishBiomass-clockStartBiomass)/60,2)

            print("Completed: " + outName + " in " + str(biomassTime)+" minutes!"+'\n')
    print("Starting the C to N convertions!" + '\n')

    # Second time through the driver list to complete nitrogen ratios last
    # Create Nitrogen grams per square meter from carbon layers
    # Values are carbon to nitrogen ratios from Bob ratio chem file
    for i in driverList:

        # Get output file name
        outName = i[0]
        # Get conversion boolean, TRUE equals file is a nitrogen ratio conversion
        conversionOut = i[3]
        # Carbon filename associated with the nitrogen ratio conversion
        cToNassociateFile = i[4]
        # Get ratio conversion, check that it is not empty first.
        # Empty set are files that don't have a conversion n to c ratio
        if i[5] != '':
            cToNratio = float(i[5])

        if conversionOut == 'TRUE':

            # start time clock
            clockStartBiomass = time.clock()

            # Concatinate output dir with input file name
            associateFile = finalPoolDir + cToNassociateFile
            # Get array from ascii raster values from associate carbon raster
            associateArray = ascii2array(associateFile)

            # Create the associated c to n array chemestry pool and return an
            # array used to create chem pool raster
            ratioArray = numpy.divide(associateArray,cToNratio)
            # Concatinate output dir with output file name
            outName = finalPoolDir + outName
            # Create the chem pool raster
            array2ascii(ratioArray, outName, asciiaoi = None, headerList = buildRatios.header)

            # end time clock
            clockFinishBiomass = time.clock()
            biomassTime = round((clockFinishBiomass-clockStartBiomass)/60,2)

            print("Completed: " + outName + " in " + str(biomassTime)+" minutes!"+'\n')

    print("Done!"+'\n')

    ##################################################################################
    # End Processing block
    ##################################################################################

if __name__ == "__main__":
    sys.exit(main())
```

D.5 | Create Land-zoning Maps for a VELMA Application

Overview *(Tutorial D.5 – Create Land-zoning VELMA Input Data)*

This document describes how to set up a land-zoning map, which includes the same integer code for all pixels in a particular zone. Once a land-zoning map is established, a VELMA user can apply a disturbance (harvest, fertilization, etc.) to a specified zone. Land-zoning maps can be used to define boundaries for land management units, land ownership, cover types, and so on.

Note: Detailed instructions for setting up and applying disturbances to land-zoning maps are provided in section 24.0 in the VELMA 2.0 User Manual (https://www.epa.gov/water-research/visualizing-ecosystem-land-management-assessments-velma-model-20).

The DEM file is the “master” layer for a simulation run: all other spatially-explicit data is assumed to have the same row, column, cell size and x, y offset values as the DEM file. Spatial data is nearly always input from files in ESRI Grid ASCII format (.asc), therefor all spatial data inputs including the Land-zoning maps must match the DEM file (.asc.).

Shapefile Conversions

Forest practices, land ownership, and other land parcel maps are typical packaged as shapefiles in various State projections and need to be converted into VELAM compatible spatial inputs.

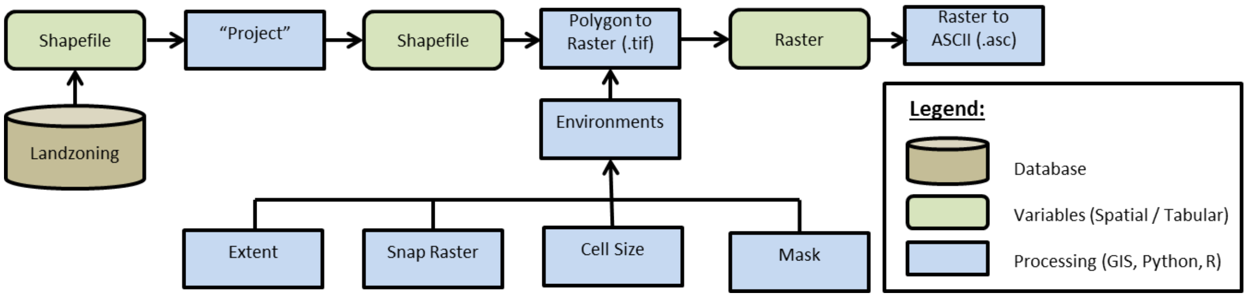


Figure 11 GIS Processing

Using a common GIS package such as ArcMap the process flow would be as follows (figure 1):

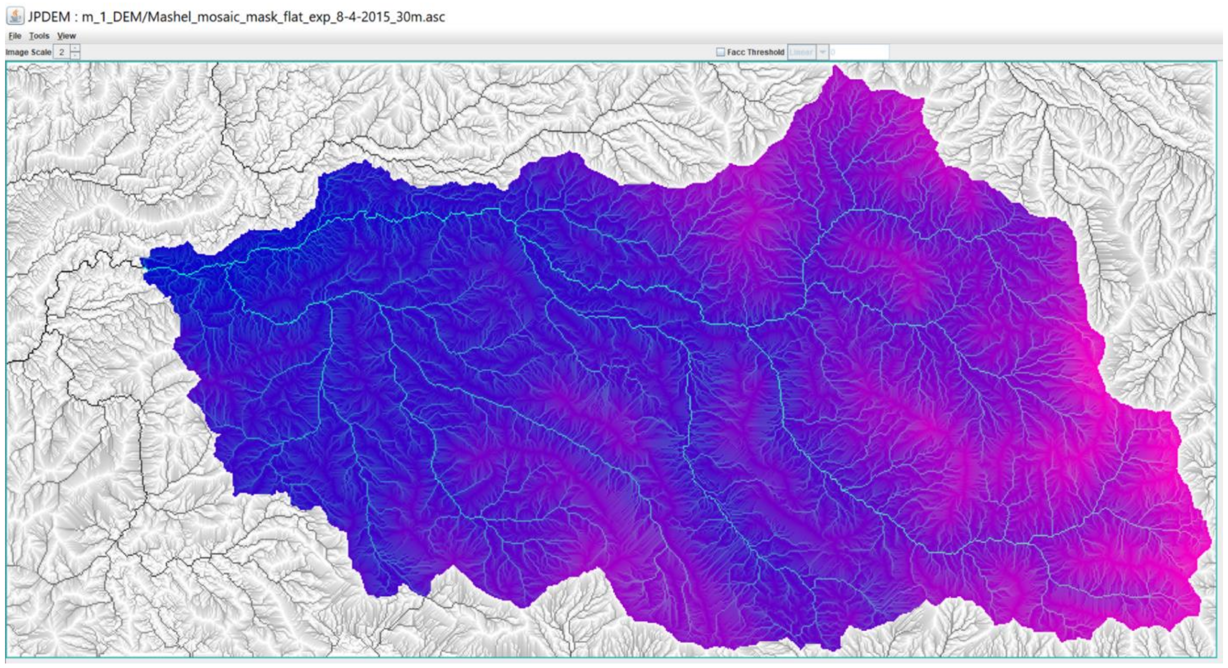
1. Initial data acquisition from a database -> Shapefile
2. Reproject the Shapefile to match the VELAM DEM projection.
    - “Project” Data Management
3. Convert the Shapeflie to Raster.
    - “Polygon to Raster” (.tif) Conversion
    - While using the “Polygon to Raster” tool the user must define the “Environments settings” to ensure correct extent, cell size, and snapping to the DEM.

- “Processing Extent”
  1. “Extent” -> to match DEM file
  2. “Snap Raster” -> to match DEM file
- “Raster Analysis”
  1. “Cell Size” -> to match DEM file
  2. “Mask” -> to match DEM file

4. Convert Raster to ASCII
   - “ASCII to Raster” (.asc)

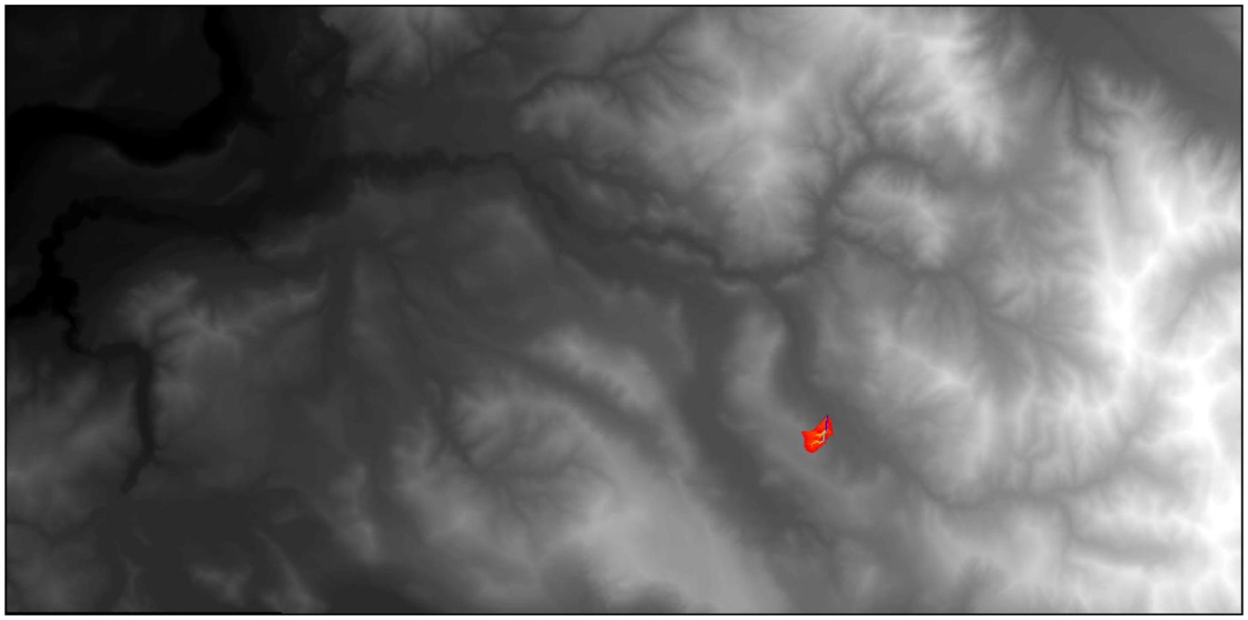
GIS Examples

JPDEM Delineation of the 209 km2 Mashel River Watershed, WA.

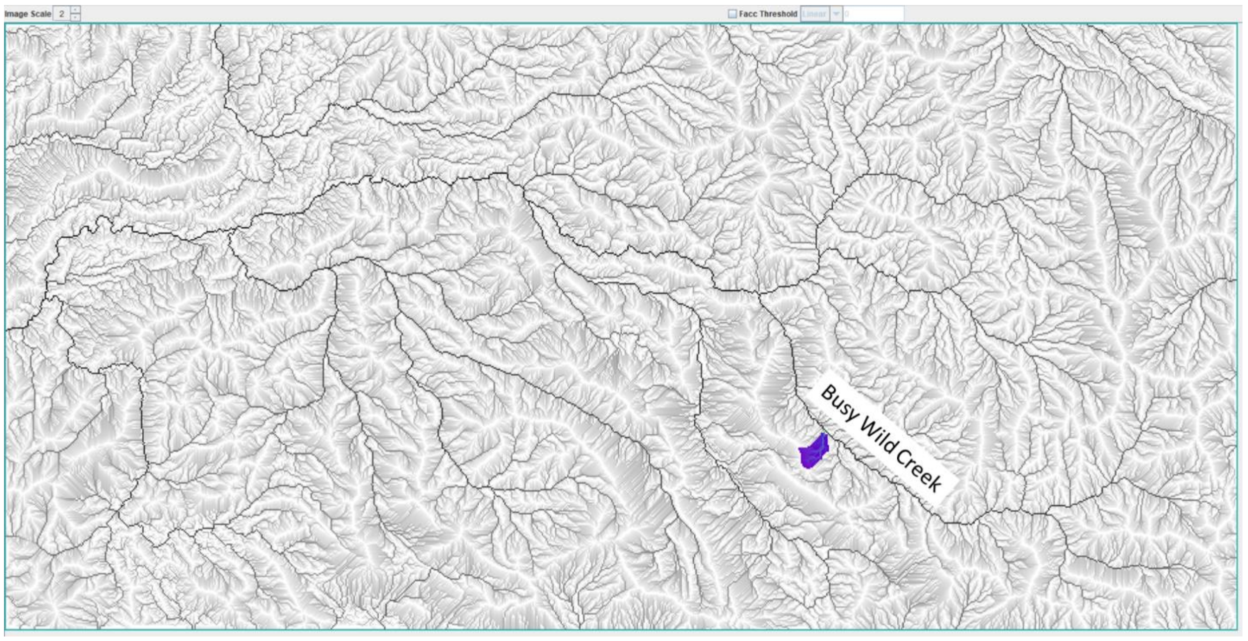

JPDEM : m_1_DEM/Mashel_mosaic_mask_flat_exp_8-4-2015_30m.asc
File
Tools
View
Image Scale
2
Facc Threshold

(continued)

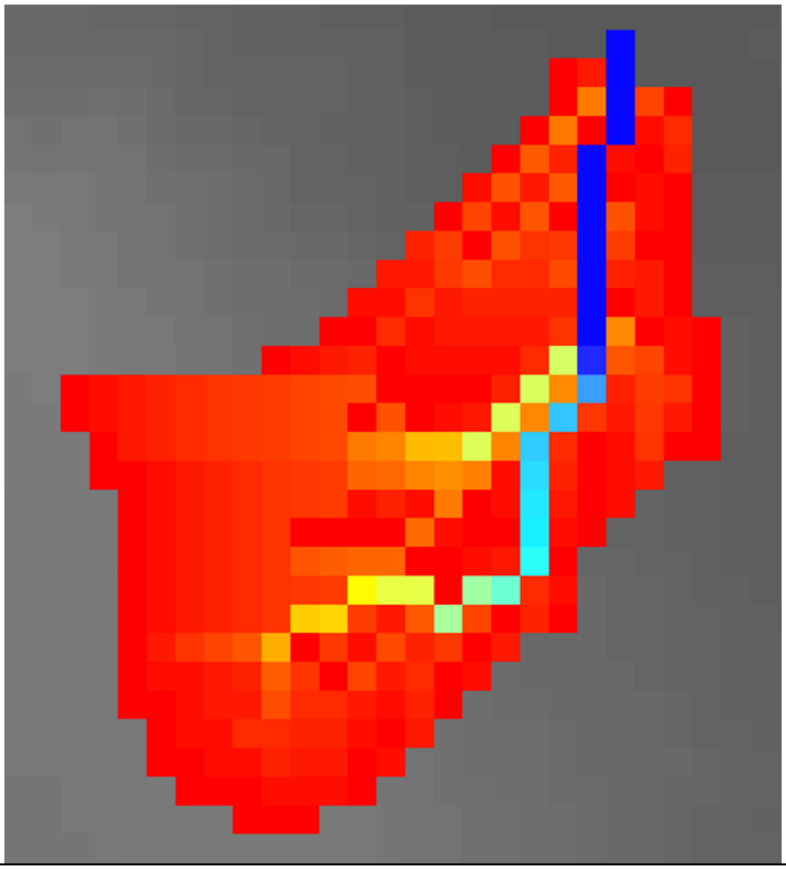
Hill-shade view of 4 km^2 Busy Wild Creek subwatershed (red) within Mashel River Watershed (209 km^2)



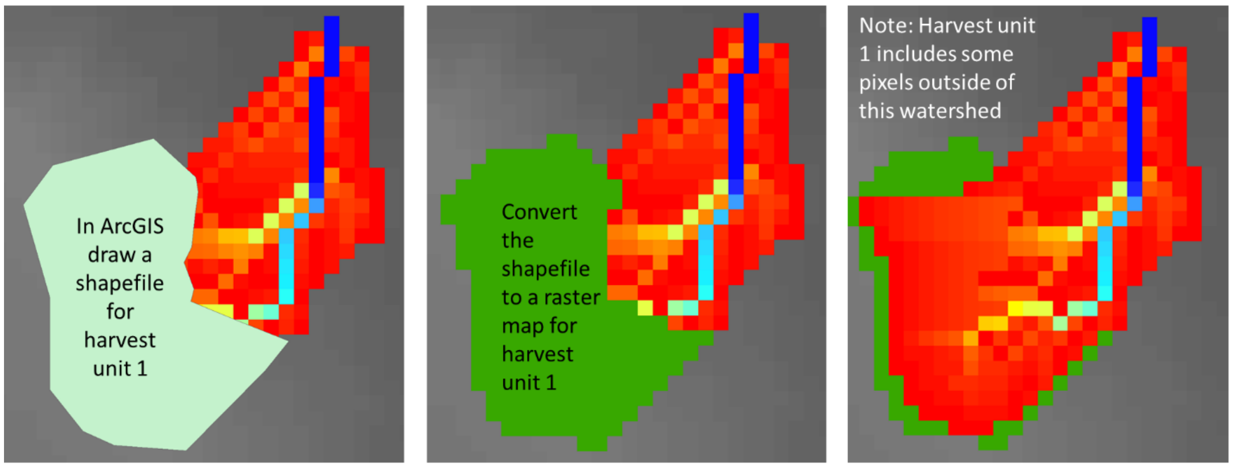
Flow-path view of 4 km^2 Busy Wild Creek subwatershed (blue) within Mashel River Watershed (209 km^2)



Zoomed-in view of the 4 km^2 Busy Wild Creek subwatershed shown on page 5.
This view shows individual 30-meter pixels with coloration indicating flow path intensity (blue = highest flow; red = lowest flow).

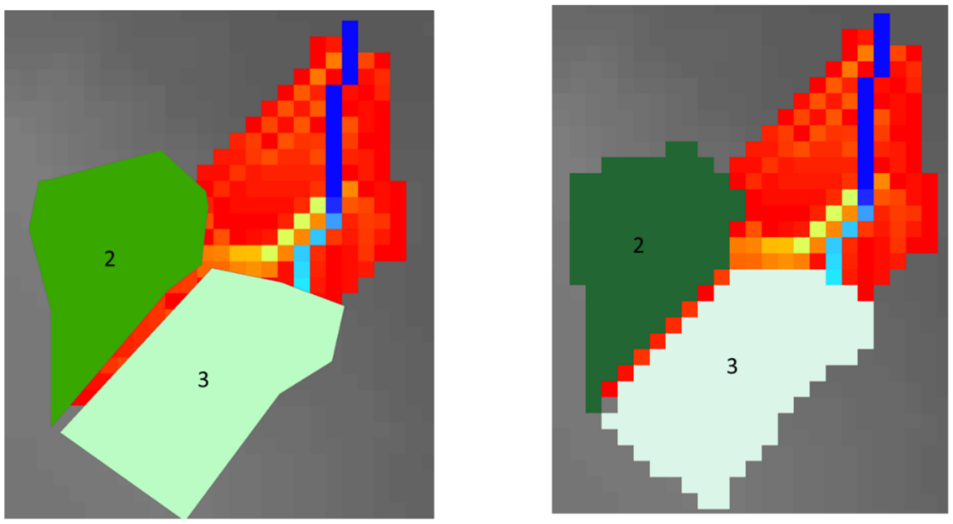


How to establish a harvest unit


In ArcGIS draw a shapefile for harvest unit 1
Convert the shapefile to a raster map for harvest unit 1
Note: Harvest unit 1 includes some pixels outside of this watershed

fpa_all_x613_y306_HarvestTreatment1.asc
Location: .....\DataInputs\o_1_LandZoning

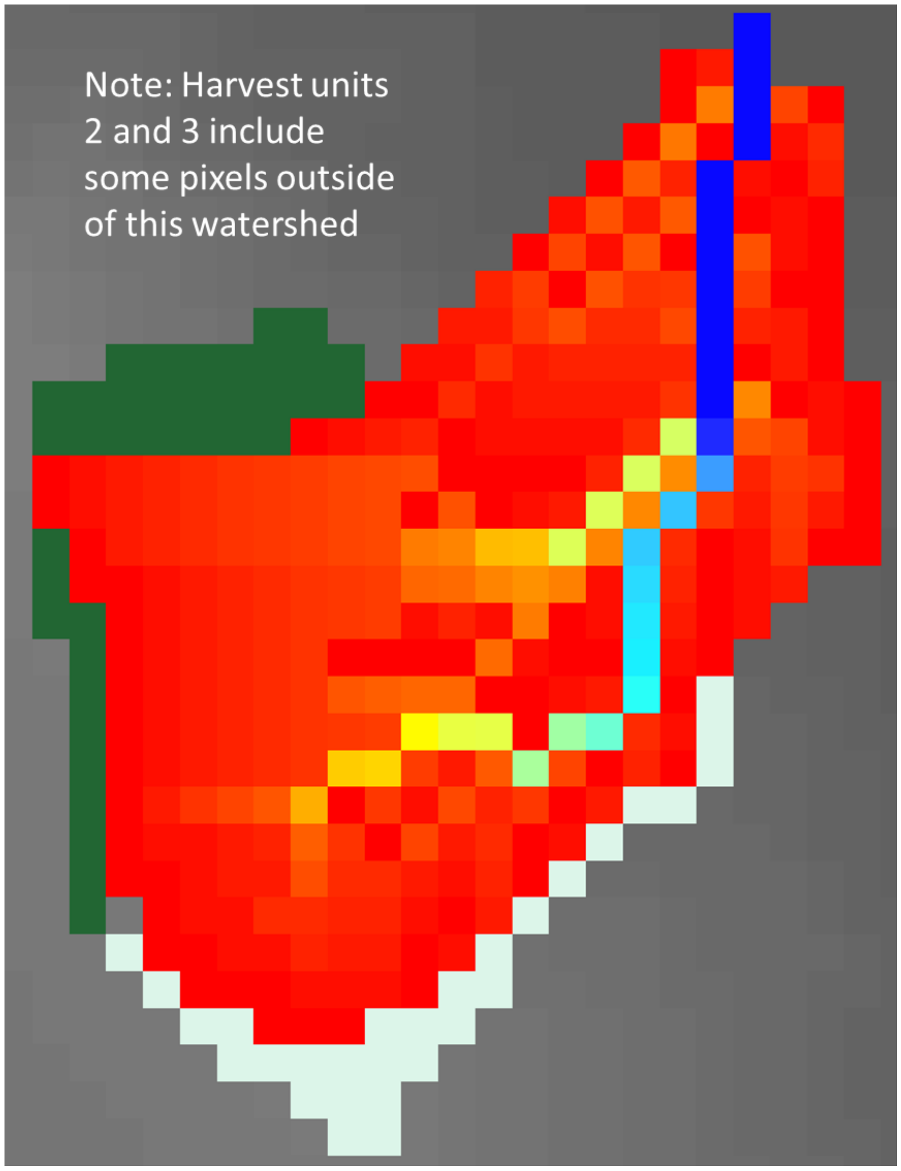
How to establish two harvest units that may differ in management



fpa_all_x613_y306_HarvestTreatment2and3.asc
Location: .....\DataInputs\o_1_LandZoning

To apply a forest harvest or other disturbance to these units, the VELMA user will need to assign a single, unique value to each unit. For this example, all pixels falling within map unit 2 would need to be labeled 2, and all pixels in map unit 3 would need to be labeled 3. Separate disturbances could then be designed for each unit, for example, unit 2 = clearcut; unit 3 = 50% thinning.

For details on how to set up and apply disturbances, see Section 24, page 102, of the VELMA Version 2.0 User Manual *(McKane, R.; Brookes, A.; Djang, K.; Stieglitz, M.; Abdelnour, A.; Pan, F.; Halama, J.; Pettus, P.; Phillips, D. 2014. Velma User Manual and Technical Documentation. U.S. Environmental Protection Agency Office of Research and Development National Health and Environmental Effects Research Laboratory: Corvallis, OR, USA. Link: https://www.epa.gov/water-research/visualizing-ecosystem-land-management-assessments-velma-model-20*



D.6 | Add Georeferenced Projection Information to VELMA Output ASCII Maps

Overview *(Tutorial D.6 – Add Georeferenced Projection Info to VELMA Output ASCII Maps)*

This document describes how to add georeferenced projection information to VELMA's spatial output files so that the outputs can be correctly displayed in space with reference to other geospatial data.

Very important when using ArcGIS, VISTAS or other visualization software to compare VELMA outputs with other geospatial data.

Note: Instructions for creating VELMA spatial output files can be found in section "23.0 – Spatial Data Items" in the VELMA 2.0 User Manual (https://www.epa.gov/water-research/visualizing-ecosystem-land-management-assessments-velma-model-20).

VELMA output maps are by default produced without adjoining projection files. This script will create the corresponding projection information for each output map so that the user can properly geolocate the ASCII maps in a GIS software, such as ArcGIS or QGIS. The user will have to create or provide a single projection file from which all other maps will be referenced to.

Software Requirements and Processing

To run and produce the adjoining projection files for a set of VELMA ASCII map outputs you will need the following:

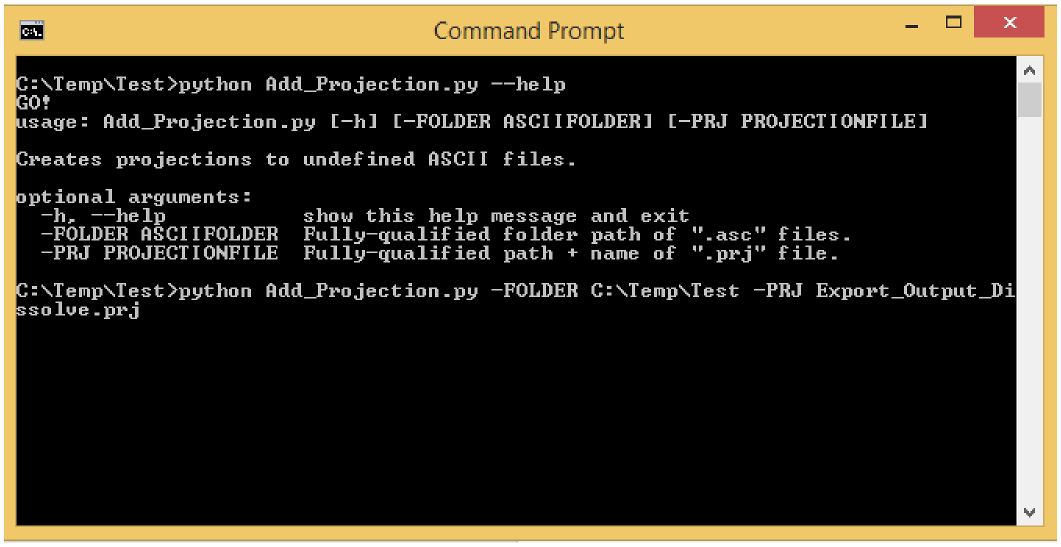
E. Python version 2.x: Current release is 2.7.12
    1. Python comes pre-packaged within ArcGIS and QGIS, so it is likely you already have it installed on your computer. For example, a default install is currently here: C:\Python27\ArcGIS10.2\python.exe. Check to see if you have Python installed before installing a new version. If Python is not installed, you can obtain a copy here: https://www.python.org/download/releases/2.7/ *Note that Python 2.7 is currently considered safe for use on U.S. EPA network and non-network computers.*

F. ASCII Maps requirements: VELMA ASCII raster outputs.

G. A single projection (.prj) file that matches the projection of raster outputs.

H. "**Add_Projection.py**" and can be run with Command Prompt line with Python.
Command prompt input example:
"`python Add_Projection.py -FOLDER C:\Temp\Test -PRJ mypojectionFile.prj`"
"`python Add_Projection.py --help`" for exact arguments (Figure 1).

(continued)


Command Prompt

```
C:\Temp\Test>python Add_Projection.py --help
GO!
usage: Add_Projection.py [-h] [-FOLDER ASCIIFOLDER] [-PRJ PROJECTIONFILE]

Creates projections to undefined ASCII files.

optional arguments:
  -h, --help           show this help message and exit
  -FOLDER ASCIIFOLDER  Fully-qualified folder path of ".asc" files.
  -PRJ PROJECTIONFILE  Fully-qualified path + name of ".prj" file.

C:\Temp\Test>python Add_Projection.py -FOLDER C:\Temp\Test -PRJ Export_Output_Di
ssolve.prj
```

Figure 1. Command line example of running the projection adding script.

Scripts

```
# Add_Projection.py
# Author: Paul Pettus
# Date: 5-1-2017
# Description: Adds projections to raw ASCII VELMA file outputs
#
# VELMA Output files are created without projection.  This script creates
# projections for those output files from a single projection file.
#
# Last updated: 5-2-2017

import os, sys, argparse

# ---------------------------------------------------------------------------------------------------

# Error message class
class Usage(Exception):
    def __init__(self, msg):
        self.msg = msg

def main(argv=None):

    if argv is None:
        argv = sys.argv

    try:
        print("GO!")
        parser = argparse.ArgumentParser(description='Creates projections to undefined ASCII files.'
                                    )
        parser.add_argument('-FOLDER', dest='asciiFolder',
                    help='Fully-qualified folder path of ".asc" files.')

        parser.add_argument('-PRJ', dest='projectionFile',
                    help='Fully-qualified path + name of ".prj" file.')

        args = parser.parse_args()

        # args parsing
        prjFile = os.path.abspath(args.projectionFile)
        inFolder = os.path.abspath(args.asciiFolder)

        print (prjFile)
        print (inFolder)

        # do the work
        addProjections(prjFile, inFolder)

    except Usage as e:
        print(e.msg)
        return 2

    except Exception as e:
        # STUB exception handler
        # Warning: poor programming style.
        # Catches almost any exception (but not KeyboardInterrupt -- which is a Good Thing)
        raise e

def addProjections(prjFile, inFolder):

    # Check that projection file exists
    if not os.path.exists(prjFile):
        raise Usage('Cannot find projection file "' + prjFile + '"')

    # Check that ASCII folder exists
    if not os.path.exists(inFolder):
        raise Usage('Cannot find folder "' + inFolder + '"')

    print ("Starting projection definitions.")

    # read in projection file text

    prjF = open(prjFile,'r')

    prjText = prjF.readlines()

    prjF.close()

    # read in ASCII files

    asciiList = []

    for files in os.listdir(inFolder):
        if files.endswith(".asc"):
            print(files)
            asciiList.append(files)

    # Loop through files and create new projections
    for files in asciiList:

        fileName, fileExtension = os.path.splitext(files)

        outFile = fileName + ".prj"

        print(outFile)

        f = open(outFile, "w")
        for item in prjText:
            f.write(item)
        f.close()
    print("Done!")


if __name__ == "__main__":
    sys.exit(main())
```

D.7a | Create Multiple-Location Weather Drivers – Alternate Methods Using Pseudo Weather Stations

Overview *(Tutorial D.7a – Create Multiple-Location Weather Drivers)*

Daily temperature and precipitation data for multiple weather stations can help VELMA to more accurately simulate climatic effects on streamflow and ecosystem dynamics. However, weather stations are often not optimally located or in sufficient density to support accurate model predictions.

This document describes how to add and locate pseudo-climate stations to a VELMA simulation, based upon daily climate data grids produced by sophisticated national climate models such as PRISM and Daymet. The added pseudo-stations can greatly improve VELMA’s interpolation of climate drivers across the landscape, thereby improving model performance.

The Multiple-Location Weather Model Configuration requires multiple observed weather stations data within the area of interest (AOI), and these data are not always available for the AOI. Alternatively, “station” driver data can be provided to VELMA from other forms of spatially derived daily precipitation and air temperature values for cells in a simulation’s delineated AOI. Unlike actual observed weather station data, these pseudo-station climate data are developed from sophisticated spatial climate models’ datasets such as PRISM (PRISM, 2013) or Daily Gridded Weather Data (Daymet). Both data sets are publicly available daily national climate data, gridded at 1 km x 1 km spatial resolution for Daymet, compared to the 4 km x 4 km resolution data from PRISM. Daily gridded data can be extracted as input climate driver data for a VELMA simulation, by probing latitude and longitude positions for single or multiple locations within an AOI. (Figure 1.)

(continued)

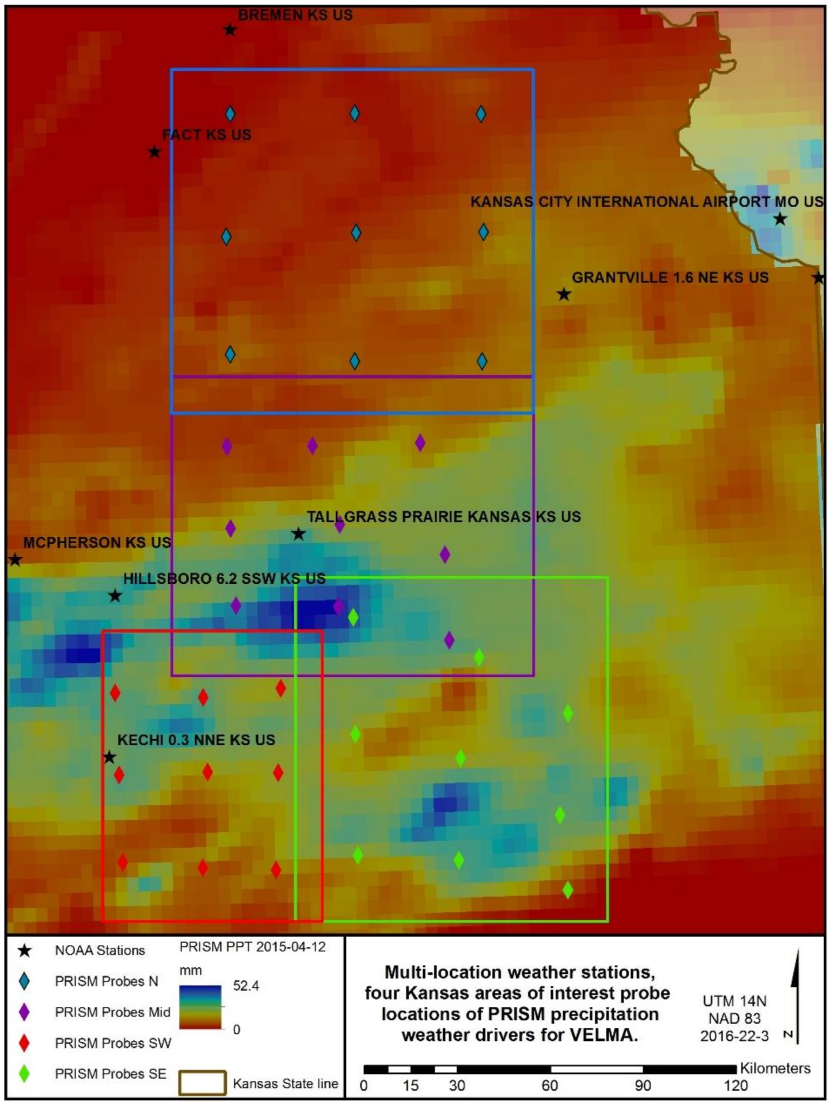


Figure 12 Pseudo weather station example, Flint hills Kansas.

Example Procedure –PRISM, Daymet, or, Observed?

Multiple weather station driver data allows VELMA to more dynamically apply spatially explicit climate drivers to a simulation. For a series of AOI’s in the Kansas’s Flint hills, there are publicly available NOAA observed weather station data that reside across the study region (Figure1). Only two of the simulated AOI’s has a single observation station that resides with its study area, while many observation stations are nearby but are outside these study areas. In this example, many of these NOAA stations had missing data for either precipitation or air temperature, as well as being located out of the AOI’s. Pseudo stations or probes of modeled climate data can, in some effect, cover the spatial dynamic of the daily climate by nearest neighbor assignment to probe station. In these study areas, nine stations were created for each AOI to fit Multi-station data from a combination of PRISM and Daymet daily modeled precipitation and air temperature data (Appendix).

In the Kansas example, the simulation was ran for the years 1998 – 2015. Higher resolution Daymet data was available for the years 1998 – 2014. While PRISM data was available for the year 2015. No continuous data were observed by NOAA in the AOI’s. Daymet data was then supplemented with PRISM data for the missing years, and formatted as required by the “Multiple-Location Weather Configuration” (Figure 2) (Appendix).

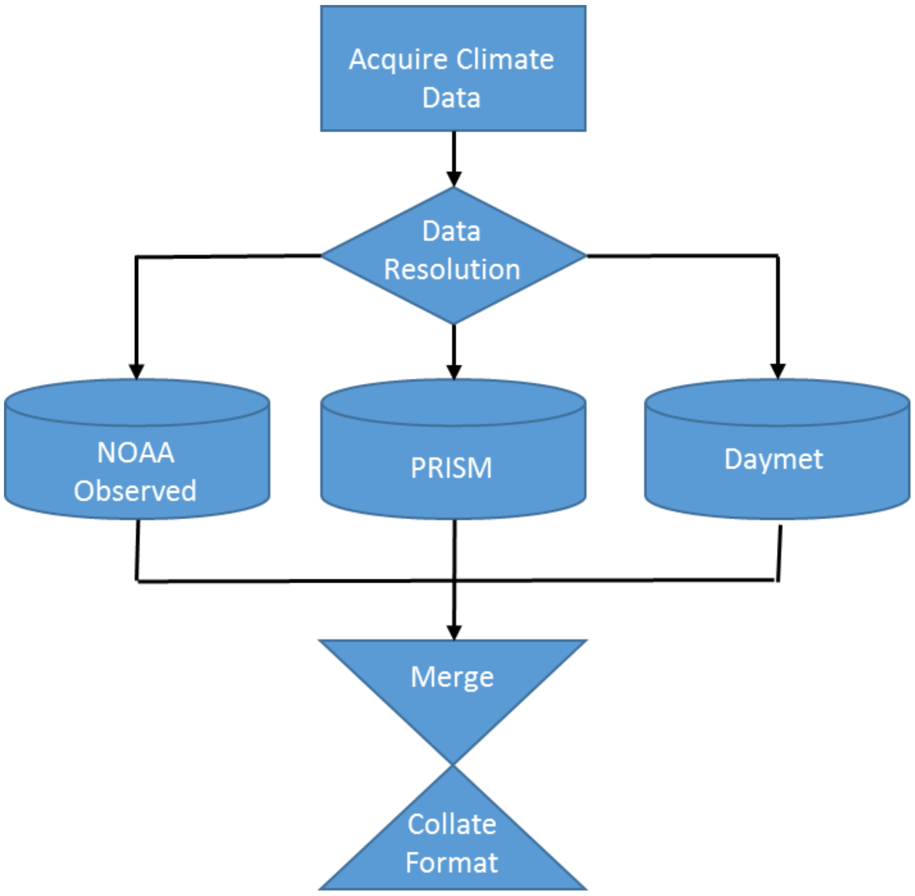
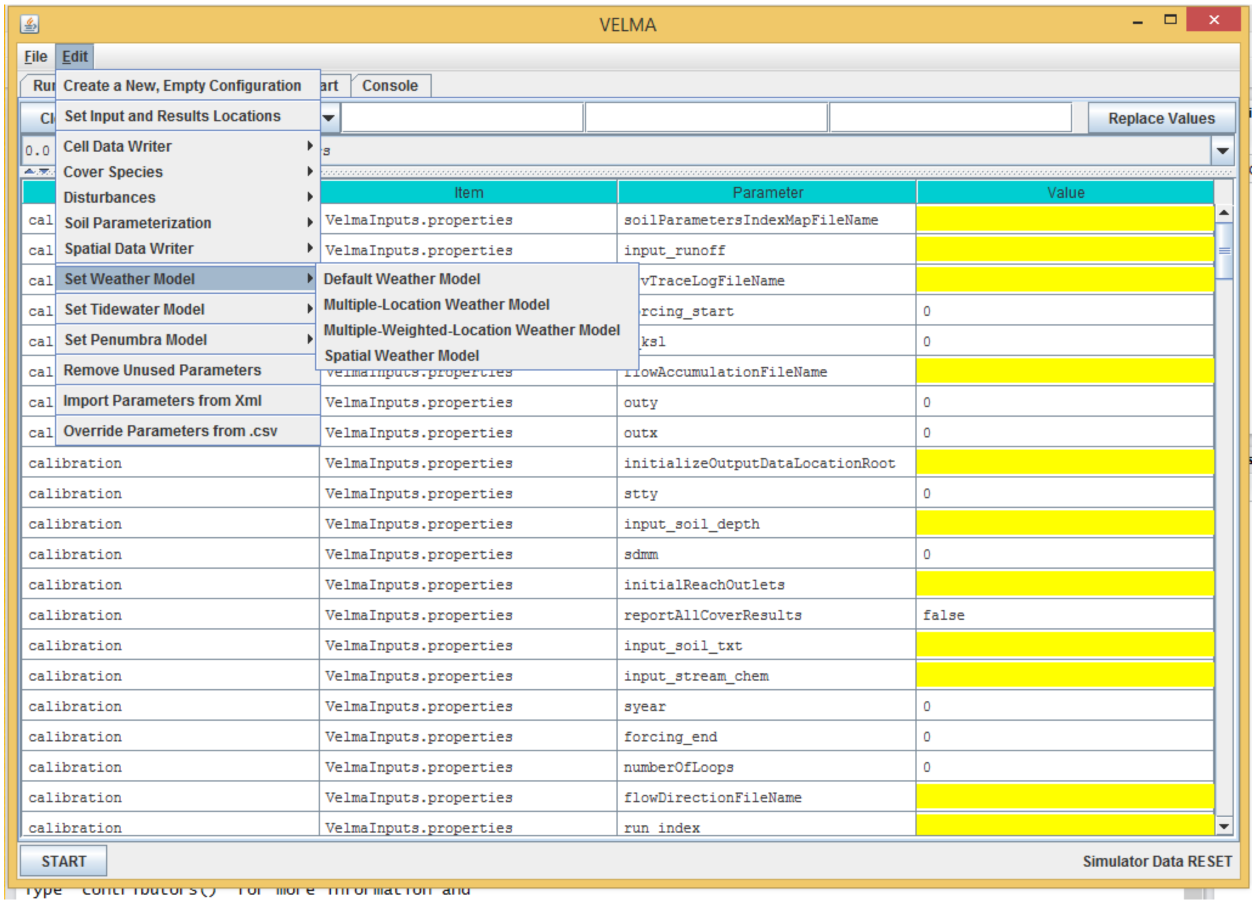


Figure 13 Process flow for multi-station climate data.

Within VELMA there are two choices for the Multi-Location models, “Multiple-Location Weather Model” and “Multiple-Weighted-Locations Weather Model” (Figure 3).

(continued)



	Item	Parameter	Value
cal	VelmaInputs.properties	soilParametersIndexMapFileName	
cal	VelmaInputs.properties	input_runoff	
cal		vTraceLogFileName	
cal		rcing_start	0
cal		ksl	0
cal	VelmaInputs.properties	flowAccumulationFileName	
cal	VelmaInputs.properties	outy	0
cal	VelmaInputs.properties	outx	0
calibration	VelmaInputs.properties	initializeOutputDataLocationRoot	
calibration	VelmaInputs.properties	stty	0
calibration	VelmaInputs.properties	input_soil_depth	
calibration	VelmaInputs.properties	sdmm	0
calibration	VelmaInputs.properties	initialReachOutlets	
calibration	VelmaInputs.properties	reportAllCoverResults	false
calibration	VelmaInputs.properties	input_soil_txt	
calibration	VelmaInputs.properties	input_stream_chem	
calibration	VelmaInputs.properties	syear	0
calibration	VelmaInputs.properties	forcing_end	0
calibration	VelmaInputs.properties	numberOfLoops	0
calibration	VelmaInputs.properties	flowDirectionFileName	
calibration	VelmaInputs.properties	run_index	

Figure 14 Multi-Location Weather Model options with in VELMA simulator.

(continued)

Multiple-Location Weather Model

The "Multiple-Location Weather Model" creates Thiessen Polygons around each of the probe stations Figure 4). This is a very common method used in hydrometeorology for creating a weighted mean from multiple measurement spots in a watershed area.

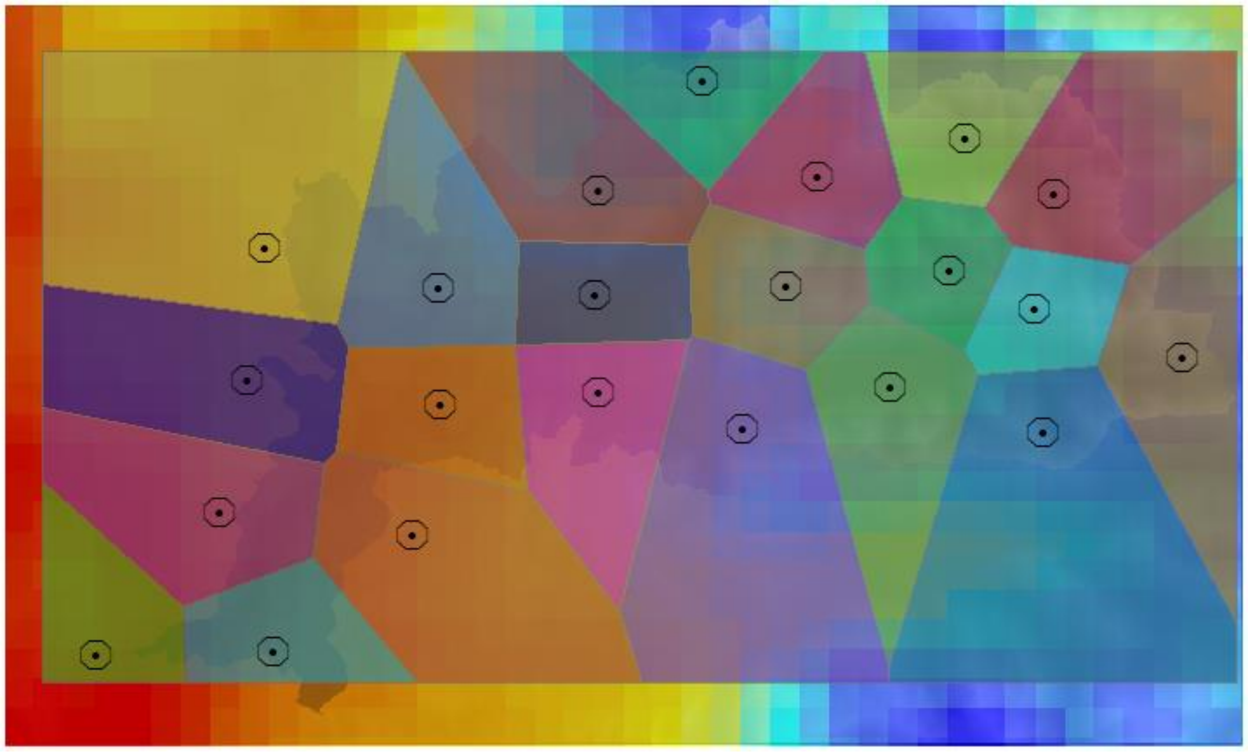


Figure 15 Thiessen Polygons around pseudo station probes.

(continued)

Multiple-Weighted-Locations Weather Model

The second method "Multiple-Weighted-Locations Weather Model" is a nonlinear interpolation or inverse distance weighting technique. The weight a sample point assigns to the averaging calculation value to a cell is based on its distance or closeness it is to the cell. This weighting creates a moving surface, based on inverse distance squared relationship instead of simple inverse (Figure 5).

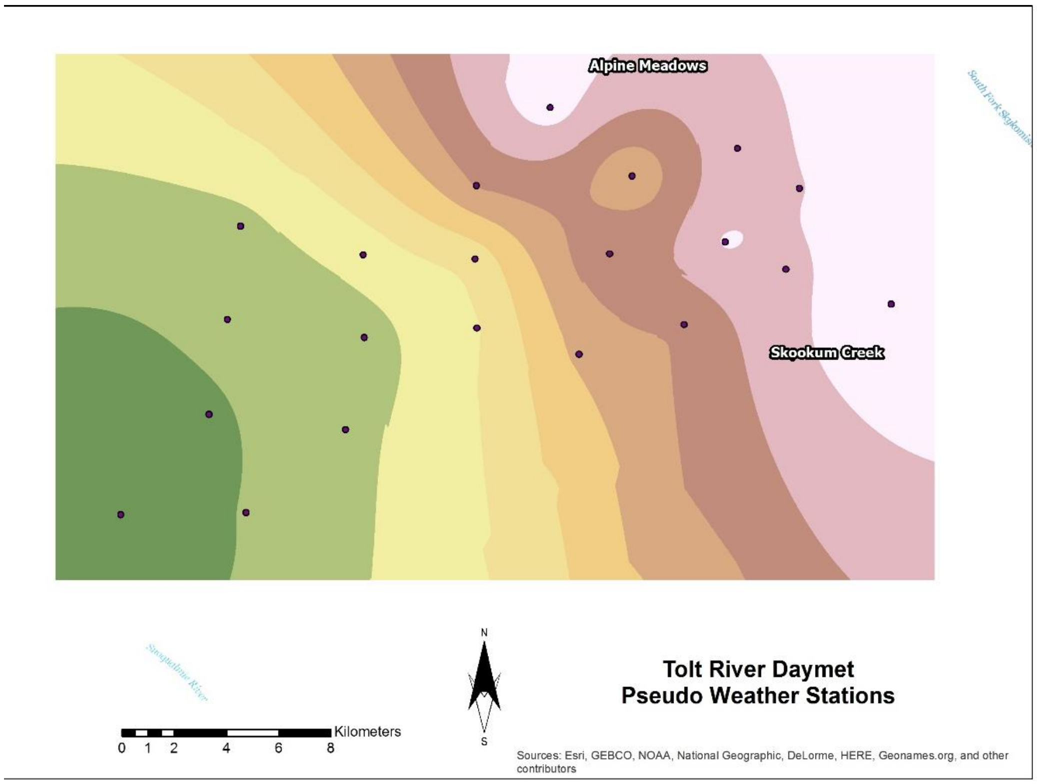


Figure 16 Inverse distance weighted influence of pseudo weather station.

You Must Provide the Weather Model with Location and Driver Data for One or More Weather Locations

The Multiple-Location Weather Model needs the location and driver data file name for each "Weather Location" it will use. A "Weather location" is simply the x and y coordinates of a cell within the bounds of the DEM grid specified for the simulation configuration, plus a file of daily driver values that specifies the precipitation and air temperature at that cell.

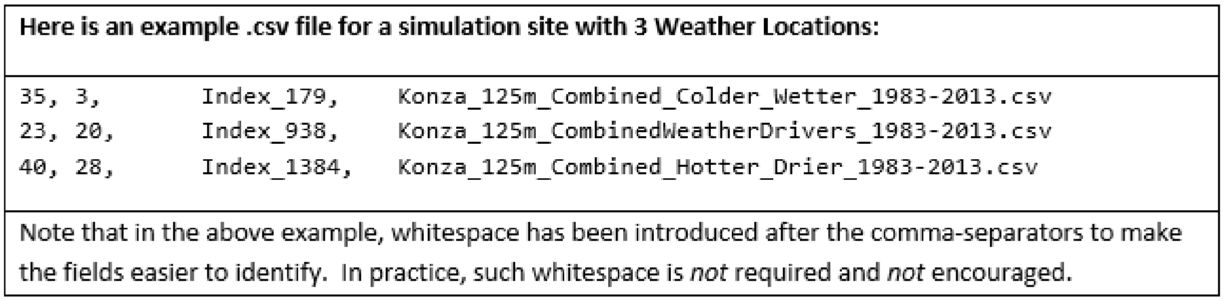
The value for the weatherLocationDataFileName parameter must be the name (or fully-qualified path + name) of a comma-separated (.csv) file with one or more rows. Each row specifies the location and driver data for a specific cell and has the following field layout:

x-coordinate, y-coordinate, uniqueName, driverFileName

The x-coordinate and y-coordinate must specify a location within the bounds of the simulation configuration's DEM grid. (I.e. they must be in the ranges [0, (ncol – 1)] and [0, (nrow – 1)] respectively.

(Recall that ncol and nrow are the DEM grid’s number-of-columns and number-of-rows values – and they must be set to match the ncols and nrows values of the DEM grid’s input_dem (.asc) file.)

As with the weatherLocationDataFileName, the driverFileName may be either a fully-qualified path + name, or simply a file name. In both cases, when presented with a file name without a path, the VELMA simulation engine’s initialization code will assume the file is located in the directory specified by the inputDataLocationRootName/inputDataLocationDirName path.



Here is an example .csv file for a simulation site with 3 Weather Locations:

```
35, 3,       Index_179,    Konza_125m_Combined_Colder_Wetter_1983-2013.csv
23, 20,      Index_938,    Konza_125m_CombinedWeatherDrivers_1983-2013.csv
40, 28,      Index_1384,   Konza_125m_Combined_Hotter_Drier_1983-2013.csv
```

Note that in the above example, whitespace has been introduced after the comma-separators to make the fields easier to identify. In practice, such whitespace is *not* required and *not* encouraged.

Note: Currently, the weatherLocationDataFileName file **cannot** contain a header row.

You Must Provide a Driver Data .csv File Corresponding to Each driverFileName Listed in the Weather Locations Data File.

Each driverFileName specifies a comma-separated (.csv) with one row of data per simulation day. Currently, the number of rows must exactly match the number of days implied by the simulation configuration’s forcing_start and forcing_end parameters. (E.g. if forcing_start = 1999 and forcing_end = 2000, then there must be 365 + 366 = 731 rows of driver data in each specified driverFileName file.)

Each row of a driverFileName file as the following field layout:

Year, Julian Day, Day’s Precipitation (mm), Average Day’s Air Temperature (degrees C)

Here is an example of a few rows from a driver file (these rows are not the first rows in the file):

```
1983,   28,    0,     3.23
1983,   29,    9.6,   1.94
1983,   30,    3.3,   -0.95
1983,   31,    12.5,  -4.6
1983,   32,    12.5,  -4.6
1983,   33,    0,     -5.53
1983,   34,    0,     -9.51
1983,   35,    0.5,   -9.36
```

Again, whitespace has been added above to make the data easier to read. Do *not* include such whitespace in actual data files.

Note: Currently, driverFileName files ***cannot*** contain header rows.

Appendix: GIS Methods

Making probe points and setting up VELMA's multiple station weather model

- In ArcGIS
    - Make point features in watershed
    - Add lat & long columns in attribute table
        - Calculate X and Y locations for the respective columns in degrees (projection?)
    - Add UTM_lat and UTM_long columns (float),
        - Calculate X and Y using NAD1983 UTM and units of meters
    - Add x-cell, y-cell columns (integer)
        - Look up top limit of AOI and left limit
        - Using field calculator:
            - x-cell = (UTM_long-AOI_left)/(cell size)
            - y-cell = (AOI_top-UTM_lat)/(cell size)
    - Export attribute table as .txt, then change to .csv
- Using R Scripts
    - Enter lat-lon values into 'daymetR' R script to download data from Daymet for each station
        - May need to change some parts of script like directories/filenames
        - SNOTEL locations will need to be updated when exact locations are found
            - Bob can identify from satellite imagery
    - Use 'fix_daymet' R script to account for Daymet lack of leap year (averages first and last day of year)
        - Some things in script like file names, directories, and years need to be altered
- Creating VELMA set up files for Multiple-location Weather Configurations (this tutorial)
    - Need to make .csv set up file in Excel for VELMA for multiple weather stations with x, y, cell_index , weather station file name
    - 'fix_daymet' R script gives an output of created file names, can be copied into this
    - for cell_index column can use any unique identifier, we just used 'x_XXXX_y_YYYY' with XXXX and YYYY being the x and y cords
- VELMA set up
    - Edit -> Set Weather Model -> Multiple-Location Weather Model

'weatherLocationsdatafilename' variable is the configuration file made above

(continued)

Appendix (Example R code):

DaymetR_KS.r

```
# Author: Paul Pettus
# Purpose: Download Daymet climate data via lat long probe method


install.packages("rgeos")
install.packages("DaymetR")
library("DaymetR")

setwd("d:/temp/netCDF/")

# Download a probe site by latitude and longitude

download.daymet(site="Northern_KS_1998-2014_X_478_Y_427",lat=39.3031,lon=-96.3158,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_474_Y_759",lat=38.9298,lon=-96.3348,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_474_Y_119",lat=39.65,lon=-96.3074,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_153_Y_121",lat=39.658,lon=-96.7758,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_142_Y_437",lat=39.3019,lon=-96.8018,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_153_Y_741",lat=38.9604,lon=-96.7978,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_800_Y_121",lat=39.6361,lon=-95.8344,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_806_Y_426",lat=39.2931,lon=-95.8401,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Northern_KS_1998-2014_X_803_Y_759",lat=35.8617,lon=-95.8617,start_yr=1998,end_yr=2014,internal=FALSE)


download.daymet(site="Southern_KS_1998-2014_X_258_Y_621",lat=37.4711,lon=-96.9415,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_51_Y_607",lat=37.4914,lon=-97.2331,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_447_Y_626",lat=37.4599,lon=-96.6753,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_32_Y_170",lat=37.9838,lon=-97.249,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_42_Y_381",lat=37.7462,lon=-97.2408,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_271_Y_374",lat=37.7487,lon=-96.9161,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_452_Y_375",lat=37.7423,lon=-96.6586,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_459_Y_157",lat=37.9872,lon=-96.6409,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southern_KS_1998-2014_X_259_Y_181",lat=37.966,lon=-96.9258,start_yr=1998,end_yr=2014,internal=FALSE)

download.daymet(site="Middle_North_KS_1998-2014_X_172_Y_594",lat=38.2307,lon=-96.7979,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_157_Y_396",lat=38.4546,lon=-96.812,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_148_Y_185",lat=38.6926,lon=-96.8179,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_368_Y_183",lat=38.6883,lon=-96.5013,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_722_Y_684",lat=38.113,lon=-96.0174,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_436_Y_598",lat=38.2189,lon=-96.4213,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_439_Y_385",lat=38.4586,lon=-96.4075,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_647_Y_176",lat=38.6873,lon=-96.101,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Middle_North_KS_1998-2014_X_711_Y_463",lat=38.3613,lon=-96.0229,start_yr=1998,end_yr=2014,internal=FALSE)

download.daymet(site="Southeastern_KS_1998-2014_X_154_Y_414",lat=37.848,lon=-96.3725,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_161_Y_727",lat=37.4954,lon=-96.3756,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_149_Y_112",lat=38.1875,lon=-96.3673,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_422_Y_738",lat=37.4738,lon=-96.0081,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_703_Y_360",lat=37.8888,lon=-95.5907,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_426_Y_476",lat=37.7686,lon=-95.9892,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_474_Y_216",lat=38.0601,lon=-95.9096,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_703_Y_815",lat=37.3764,lon=-95.6141,start_yr=1998,end_yr=2014,internal=FALSE)
download.daymet(site="Southeastern_KS_1998-2014_X_682_Y_622",lat=37.5948,lon=-95.6343,start_yr=1998,end_yr=2014,internal=FALSE)
```

fix_Daymet_LeapYear_PPT_TMEAN.r

```
# Author: Paul Pettus
# Purpose: Format Daymet data, fix leap years, create air temp mean.
# Daymet data does not create daily files for leap day on leap years, it drops Dec 31
# data on leap years.  VELMA requires leap days, solution is to average both mean
# air temp and precipitation of julian day 365 and 1 -> julian day 366
# For data sets ending in a leap year, such as 2016, julian day 365 data
# is repeated for julian day 366.

#install.packages("miscTools")
library("miscTools")
library(plyr)

###########################################################################################
# Modify setwd (working directory)
# Modify outDir (output file directory)
setwd("C:/Temp/VELMA_Mobile_Bay/Climate/")
outDir<-"C:/Temp/VELMA_Mobile_Bay/Climate/Output/"
# Modify station output processed list file name (.csv)
stationsFile<-"C:/Temp/VELMA_Mobile_Bay/Climate/Output/Mobile_Stations.csv"
###########################################################################################


# Expected input is "download_daymet" from "daymetr" package
# or "download.daymet" from "DaymetR" package
inFiles <- list.files(pattern="*.csv")
nFiles <-  length(inFiles)

# For multiple probe station files
# For each leap year in the study range input a mean value for julian day 366
for (name in inFiles)
{
```

```
print(name)
# test file
# myfile<-"C:/Temp/VELMA_Mobile_Bay/Climate/Daphne_0.4_SW_AL_US_2012-2016_X_989_Y_-7027_2012_2016.csv"
# will overwrite test file with listed file
myfile<-name
# read in raw daymet file
df = read.csv(myfile,skip=8,header=FALSE,sep=",",col.names=c("year","yday","tday",
                                                            "ppt","srad","swe",
                                                            "tmax","tmin","vp"))

# remove unnecessary Daymet climate columns
# keep ppt, tmax, tmin
parseFile<-df[-c(3,5,6,9)]
head(parseFile)

# add tmean column
parseFile["tmean"]<- NA

# create a mean air temp from max and min daily temp
parseFile$tmean <- (parseFile$tmax+parseFile$tmin)/2

# remove tmax, tmin columns
modDaymet <- parseFile[-c(4,5)]
head(modDaymet)

# return a vector of all unique years in the Daymet file
listYears=unique(modDaymet$year)

# A function to determin if a year is a leap year
# http://en.wikipedia.org/wiki/Leap_year
# A leap year is every 4 years, but not every 100 years, yet then again every 400 years
is.leapyear=function(year){
  return(((year %% 4 == 0) & (year %% 100 != 0)) | (year %% 400 == 0))
}

# empty vector for leap years
leapYears <- c()

# Loop through all the years looking for the leap years and add them to a list of leap years
for (yearNumber in listYears) {
  # Using the is.leapyear function return and add a leap year to the yeap year list
  if (is.leapyear(yearNumber) == TRUE) {
    leapYears <- c(leapYears,yearNumber)
  }
}

# Create a new data.frame that matches the raw Dayment frame input
# this will be filled with the output data.frame and leap day additions
newFrame <- data.frame(year= integer(0), yday=integer(0), ppt=numeric(0), tmean=numeric(0))

# track position of rows processed, start at the end of first year
rowsProcessed <- 0

# Loop through all the years to process
for (yearNumber in listYears) {
  print (yearNumber)
  # check to see if year is a leap year and leap year is not the last year to process
  if (is.element(yearNumber, leapYears) && yearNumber < listYears[length(listYears)] ){
    # Get values of 1 to 365 Julian day of the year
    yearSubset<-subset(modDaymet, year == yearNumber)
    # bind subset of year to newFrame
    newFrame<-rbind(newFrame,yearSubset)
    # increment rows processed position pointer to the end of year (365) on the original raw Daymet frame
    rowsProcessed<-rowsProcessed + 365
    # Get values of Julian day 365
    value_J_365<-modDaymet[rowsProcessed,]
    # Get values of next years Julian day 1
    value_J_1<-modDaymet[rowsProcessed+1,]
    # Average the first and last day
    value_J_366<-(value_J_365+value_J_1)/2
    # copy the year number
    value_J_366[1]=yearNumber
    # create a new 366 Julian day
    value_J_366[2]=366
    # File new data frame with new Julian 366 data
    newFrame<-rbind(newFrame,value_J_366)
  }

  # check if the year is the last year in the listYears and that it is also
  # a leap year
  else if (yearNumber == listYears[length(listYears)] && yearNumber == leapYears[length(leapYears)] ){
    # Get values of 1 to 365 Julian day of the year
    yearSubset<-subset(modDaymet, year == yearNumber)
    # bind subset of year to newFrame
    newFrame<-rbind(newFrame,yearSubset)
    # increment rows processed position pointer to the end of year (365) on the original raw Daymet frame
    rowsProcessed<-rowsProcessed + 365
    # copy the last day of the year to Julian 365
    value_J_365<-modDaymet[rowsProcessed,]
    # copy the last day of the year to Julian 366
    value_J_366<-value_J_365
    # copy the year number
    value_J_366[1]=yearNumber
    # create a new 366 Julian day
    value_J_366[2]=366
```

```
      # File new data frame with new Julian 366 data
      newFrame<-rbind(newFrame,value_J_366)
    }

    # All other years are not leap years, so just add their data to the new frame
    else{
      # Get values of 1 to 365 Julian day of the year
      yearSubset<-subset(modDaymet, year == yearNumber)
      # bind subset of year to newFrame
      newFrame<-rbind(newFrame,yearSubset)
      # increment rows processed position pointer to the end of year (365) on the original raw Daymet frame
      rowsProcessed<-rowsProcessed + 365
    }
  }

  print("Processed Psuedo Station File")
  print(name)

  # Define output file name
  outFile<-paste(outDir,name,sep="")
  # convert data.frame to data.matrix for easier file writing
  matClimate<-data.matrix(newFrame)
  # Create the output file of processed weather data
  write.table(matClimate,outFile,row.names=FALSE,col.names=FALSE,sep=",")

}

# Write out a list of stations processed
write.table(inFiles,stationsFile,row.names=FALSE,col.names=FALSE,sep=",")
print("We did it, time for a coffee break!")
```

Daymet_prism_blender_2-25-2016.r

```
# Author: Paul Pettus
# Date: 2-29-2016
# Purpose:
# Blend Daymet and PRISM probe weather station data into VELMA ready driver input
#

install.packages("prism")
install.packages("zoo")


#updateR()
library("prism")
library("zoo")

# set working dir
setwd("c:/Temp/PRISM/")
mydir<-"c:/Temp/PRISM/"

# check path for prism package
path_check()

# Download study dates for precipitation and mean air temperature, PRISM
get_prism_dailys(type="tmean", minDate = "2015-01-01", maxDate = "2016-2-22", keepZip=FALSE)
get_prism_dailys(type="ppt", minDate = "2015-01-01", maxDate = "2016-2-22", keepZip=FALSE)

# Set input dir of probe data, this data was created from Daymet probes
baseInDir<-"C:/temp/Output/"

# Set output dir for final files
baseOutDir<-"C:/temp/New_Output/"

# CSV file containing the station file names and lat and long of stations
stationCSV<-"C:/temp/kansas_stations.csv"
headNames<-c("filename","lat","long")

stations <- read.csv(stationCSV,header=FALSE,sep=",",col.names=headNames)

# Determine the number of stations
nFiles <-  nrow(stations)

for (i in 1:nFiles)
{
  # Get latitude, longitude, and filename
  lat<-stations$lat[i]
  long<-stations$long[i]
  filename<-stations$filename[i]
  location<-c(long,lat)

  # Define the Daymet input file name
  fullPathFile<-paste(baseInDir,filename,"_1998_2014",".csv",sep='')
  headNames<-c("year","yday","ppt","tmean")
  # Read in Daymet probe station file
  wData<-read.csv(fullPathFile,header=FALSE,sep=",",col.names=headNames)
  # Slice prism cell data by location of cell
  curStation <- prism_slice(location,ls_prism_data()[,1])

  # Retrive precip and mean air temp
  rain<-curStation$data[grep("ppt",rownames(curStation$data)),]
  temp<-curStation$data[grep("tmean",rownames(curStation$data)),]
  rownames(rain)<-c() #remove rownames
  rownames(temp)<-c() #remove rownames
  addLength<-nrow(rain)

  # create a new data frame for the output data
  outFrame <- data.frame(year=integer(addLength),yday=integer(addLength),ppt=double(addLength),tmean=double(addLength))

  # modify the date field to julian day and year
  outFrame$yday<-as.numeric(format(rain$date, "%j"))
  outFrame$year<-as.numeric(format(rain$date, "%Y"))
  outFrame$ppt<-rain$data
  outFrame$tmean<-temp$data

  # combined Daymet frame with PRISM frame climate data
  finalFrame<-rbind(wData,outFrame)

  # Fixing file name of input files
  newFN<-gsub("2014","2015",filename)
  # Define the output file name
  outPathFile<-paste(baseOutDir,newFN,".csv",sep='')

  # write output file to final output dir.
  matClimate<-data.matrix(finalFrame)
  write.table(matClimate,outPathFile,row.names=FALSE,col.names=FALSE,sep=",") }
```

D.7b | Statistical Climate Addendum: Development of Statistical Models Based on Monthly PRISM / Daymet Data

Overview *(Tutorial D.7b – Addendum to Spatial Climate Statistics)*

VELMA requires daily temperature and precipitation climate data to accurately simulate hydrologic processes in watersheds, such as runoff and soil moisture dynamics. Therefore, the accuracy of hydrologic predictions will largely depend upon the proximity of climate a station(s) to the simulated watershed or portions therein. Especially for large watersheds having steep climatic gradients – mountains and coastal areas, for example – climate data need to be at relatively fine spatial resolution (e.g., 1 km^2) to accurately capture snow dynamics, runoff and other processes.

This document describes methods for dealing with sparse climate station data. New methods are introduced for improving the estimation of statistical parameters for the VELMA Spatial Weather Model previously described in Appendix 5 of McKane et al. (2014). These new methods include generating daily pseudo weather station driver data for areas of interest that lack ground-based weather stations.

Reference:
McKane, R.; Brookes, A.; Djang, K.; Stieglitz, M.; Abdelnour, A.; Pan, F.; Halama, J.; Pettus, P.; Phillips, D. (2014) Velma User Manual and Technical Documentation, 2nd ed.; U.S. Environmental Protection Agency Office of Research and Development National Health and Environmental Effects Research Laboratory: Corvallis, OR.

Procedure 1 – Generate statistical parameters

Following the Spatial Weather Model methodology, correlation coefficients were generated from 800m resolution PRISM 30-year monthly temperature and precipitation normals for 1981-2010 (PRISM, 2013). Correlation models were build based on a single independent variable, elevation. Gridded PRISM data were downscaled to match the 30m digital elevation modeled data. Relational models were then generated for each 30 year normal month cell to its corresponding elevation cell of the AOI. Monthly models were developed for both mean air temperature and precipitation, for each AOI. Revised intercept and slope values were used to complete both the “Air Temperature Coefficients File” and the “Precipitation Coefficients File“, which are needed for the VELMA Spatial Weather Configuration. Unfitted parameters such as heat index and flow accumulation were set to zero, since they were unmodeled. The results are summarized in Table 1 and Table 2. All regression equations were created using R statistics package (Appendix 1).

Table 1. Mashel River Basin, WA			
Parameter	Intercept	Slope	R^2
X01_ppt	131.1239	0.165175	0.626577
X02_ppt	86.67763	0.086708	0.61712
X03_ppt	115.3467	0.08018	0.531575
X04_ppt	87.67271	0.069019	0.714808
X05_ppt	64.68459	0.077026	0.848851
X06_ppt	42.64319	0.077571	0.881703
X07_ppt	17.00757	0.036384	0.874514
X08_ppt	25.5493	0.017223	0.763215
X09_ppt	33.77266	0.050396	0.850796
X10_ppt	80.36726	0.09578	0.728162
X11_ppt	151.339	0.157905	0.62436
X12_ppt	128.1081	0.110864	0.534228
X01_mt	5.306282	-0.00422	0.87001
X02_mt	5.815265	-0.00429	0.889654
X03_mt	7.904366	-0.0051	0.919949
X04_mt	10.30037	-0.00507	0.9252
X05_mt	13.6444	-0.00504	0.924011
X06_mt	16.35791	-0.00495	0.925487
X07_mt	18.90406	-0.004	0.907101
X08_mt	19.22824	-0.00397	0.893755
X09_mt	16.23177	-0.00365	0.870096
X10_mt	11.42558	-0.00365	0.872491
X11_mt	7.16671	-0.00427	0.88421
X12_mt	4.495029	-0.00412	0.866289

Table 2. Tolt River Basin, Wa			
Parameter	Intercept	Slope	R^2
X01_ppt	156.8465	0.340983	0.873725
X02_ppt	104.9198	0.209417	0.867981
X03_ppt	118.9242	0.241798	0.878695
X04_ppt	101.0095	0.174351	0.876515
X05_ppt	91.96905	0.123868	0.854035
X06_ppt	70.32923	0.10755	0.854376
X07_ppt	32.10464	0.059888	0.86893
X08_ppt	28.56578	0.062882	0.872201
X09_ppt	59.68145	0.114886	0.865976
X10_ppt	115.1068	0.198013	0.87521
X11_ppt	194.4334	0.356659	0.874651
X12_ppt	152.0708	0.273897	0.868276
X01_mt	4.973876	-0.00426	0.889799
X02_mt	5.770962	-0.00439	0.900742
X03_mt	7.638908	-0.00501	0.918905
X04_mt	9.851594	-0.00459	0.907267
X05_mt	13.03208	-0.00482	0.921168
X06_mt	15.6836	-0.00476	0.920645
X07_mt	18.31654	-0.00379	0.912681
X08_mt	18.68841	-0.00389	0.916803
X09_mt	15.68156	-0.00357	0.896509
X10_mt	11.23656	-0.00382	0.911043
X11_mt	6.95446	-0.00444	0.89201
X12_mt	4.201191	-0.00419	0.884903

Procedure 2 – Daily weather drivers

Pseudo weather stations were created to solve the problem of when there is no ground observational climate data within AOI. These data are also need to properly parametrize the regression driver data of the VELMA spatial weather model. Daily Gridded Weather Data (Daymet) is a national climate data set that was developed in part from NASA funding, for the years of 1980 – 2014, and is widely used in the scientific community (Thorton et al., 2014). It models daily air temperature and precipitation at a higher 1 km x 1 km spatial resolution, compared to the 4 km x 4 km spatial resolution climate data that is publicly available from the PRISM group (PRISM, 2013).

Pseudo weather stations were created by choosing a semi homogenous elevation area within the AOI that was overlaid by one 1 km Daymet cell. Using the latitude and longitude of the cell, daily precipitation and air temperature data was downloaded for the temporal period of interest. This daily Daymet modeled data was then used as an observed "weather station" within the AOI. This cell location would be used as the "Prime Cell" by its center DEM cell x-and-y coordinates as part of the Spatial Weather Model's parameterization.

As an example, the Mowich SNOTEL weather station that is several KM north and east of the Mashel river basin in Washington was used to explore the effectiveness of this process. 1998 – 2014 SNOTEL from this station along with Daymet modeled climate for these days was plotted to explore fit ground observations vs modeled climate. Results are summarized in Figure 1 and Figure 2.

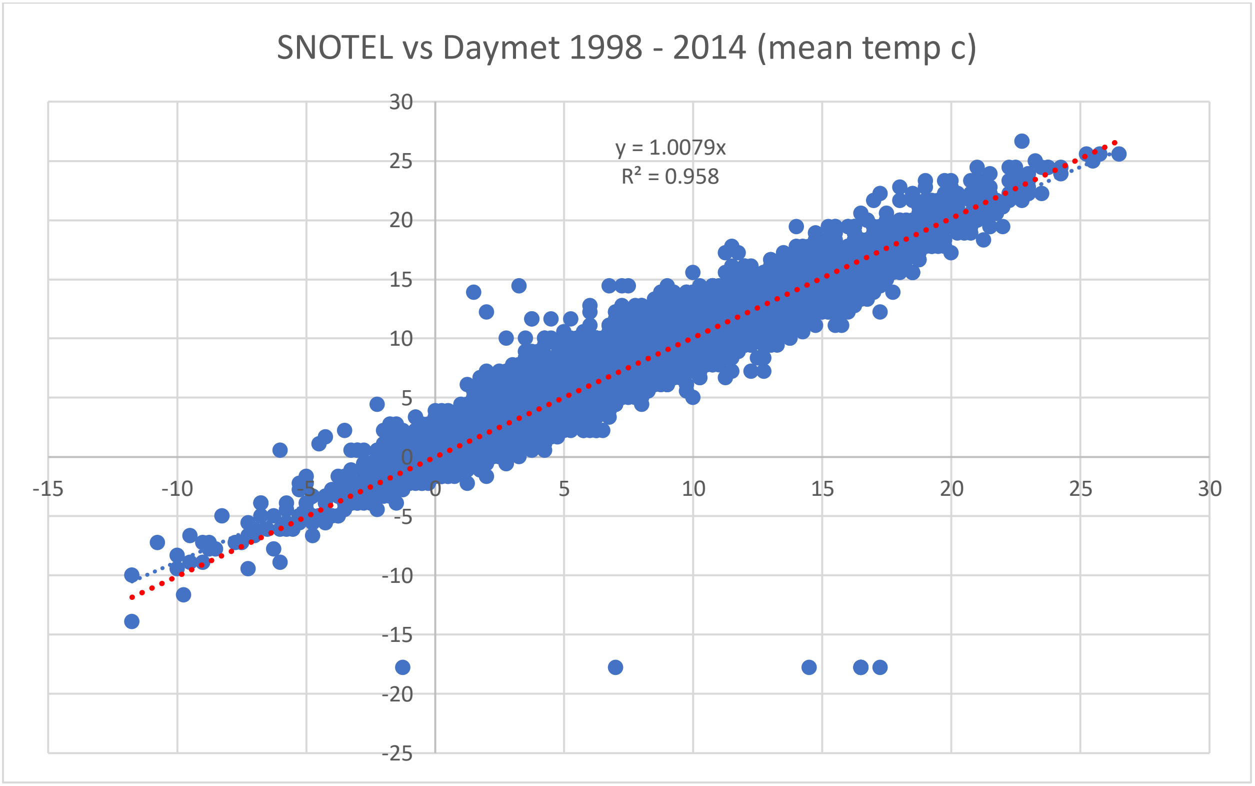


Figure 17 Daymet modeled air temperature vs overserved Mowich, WA SNOTEL data.

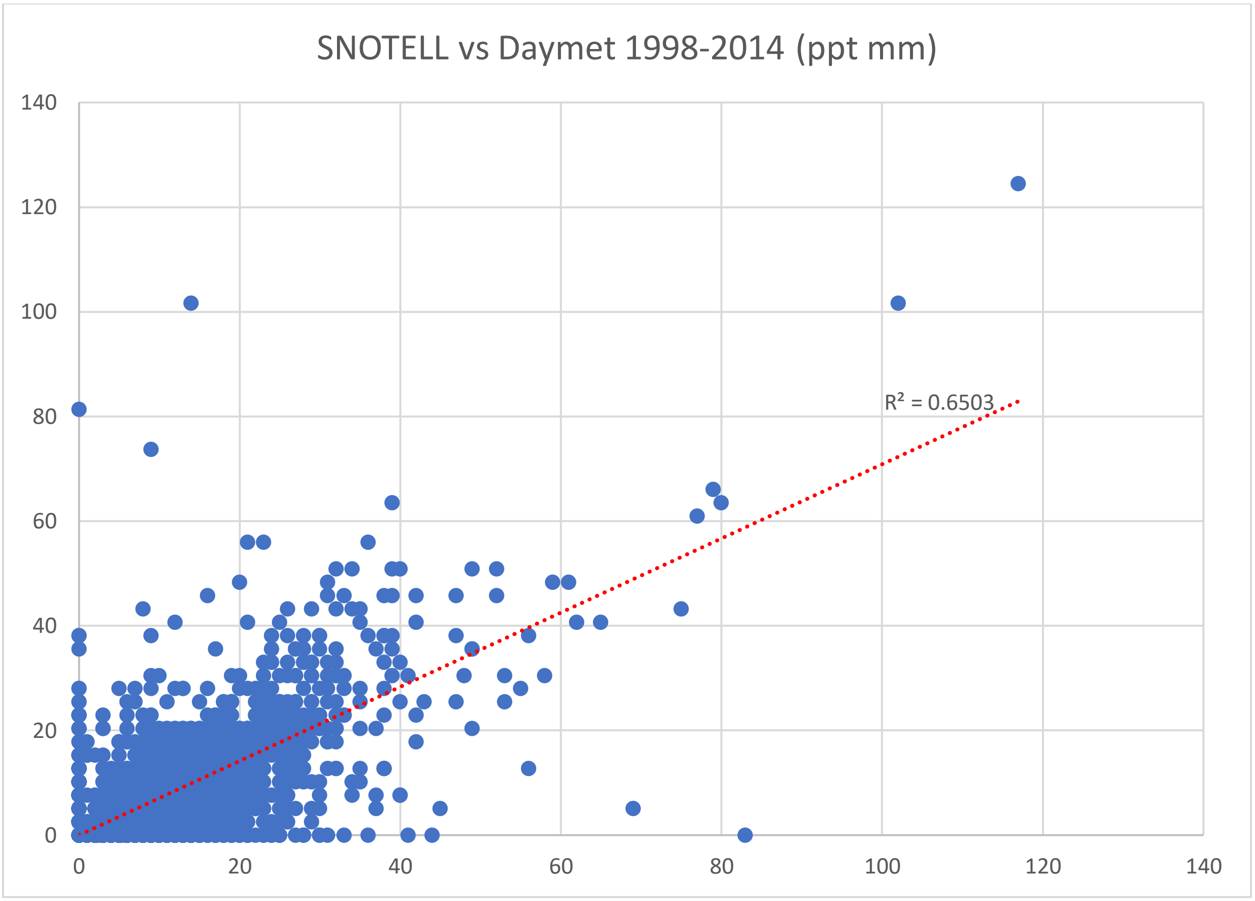


Figure 18. Daymet modeled precipitation vs overserved Mowich, WA SNOTEL data.

Addendum - Procedure 2 – Daily weather drivers

Pseudo station Daymet climate data for water years 1995 and 1999 were compared to two observed SNOTEL climate sites in the Tolt river basin, Alpine Meadows and Skookum. Daily observations of precipitation (mm / day) were accumulated over the water year starting at October 1st and plotted out against Daymet model data at Longitude and Latitudes 1km cells at the location each SNOTEL site. 10 day moving average lines were added to daily comparisons of mean air temperature observations and Daymet models temperatures at those locations. The same mean daily average temperature and precipitation comparison were made between the two SNOTEL sites.

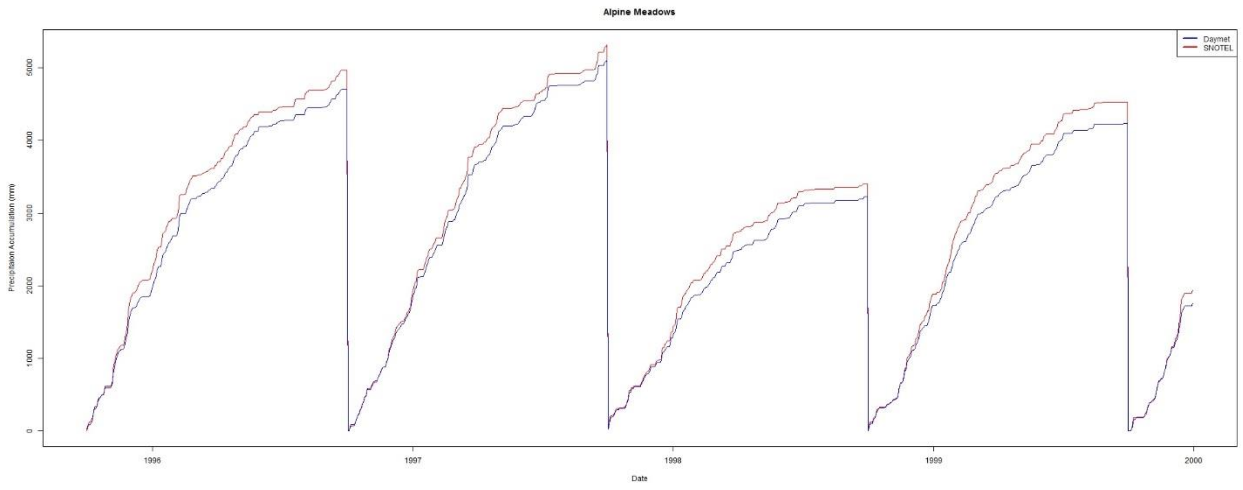


Figure 19 Accumulated precipitation, Daymet vs Alpine Meadows SNOTEL.

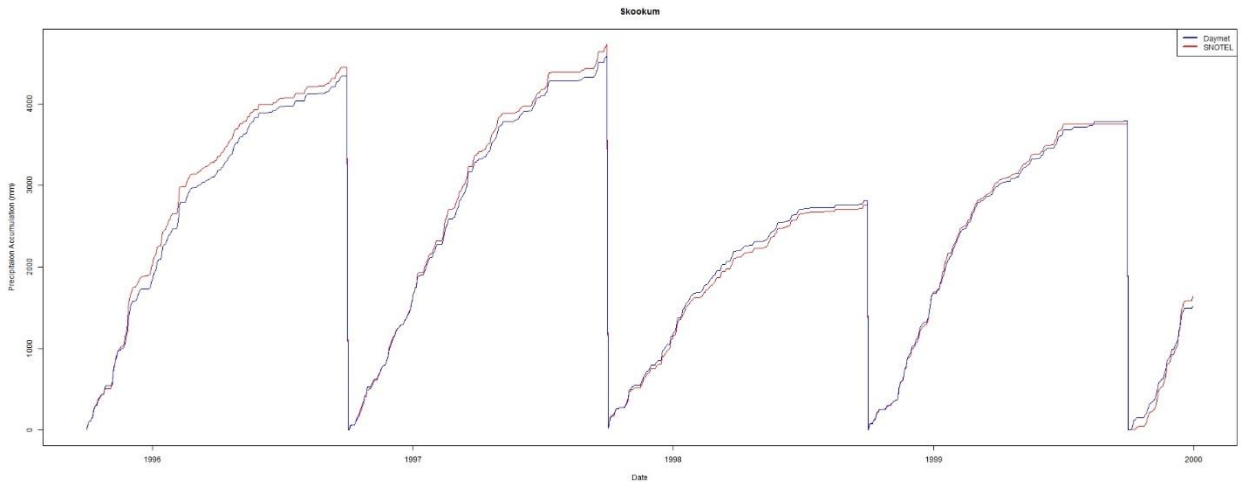


Figure 20 Accumulated precipitation, Daymet vs Skookum SNOTEL.

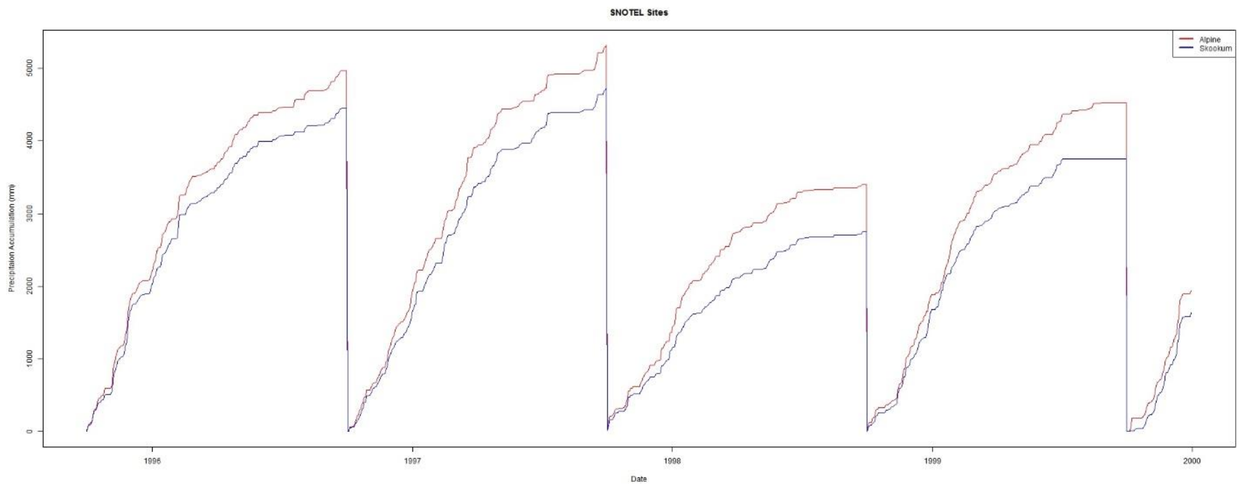


Figure 21 Accumulated precipitation, Alpine Meadows vs Skookum SNOTEL

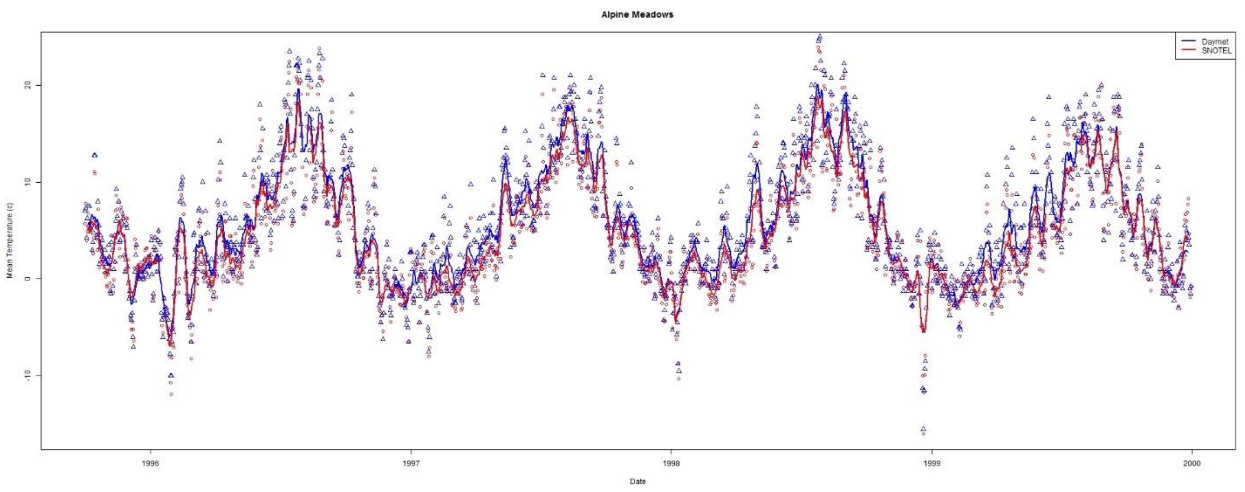


Figure 22 Daily air temperature, Daymet vs Alpine Meadows

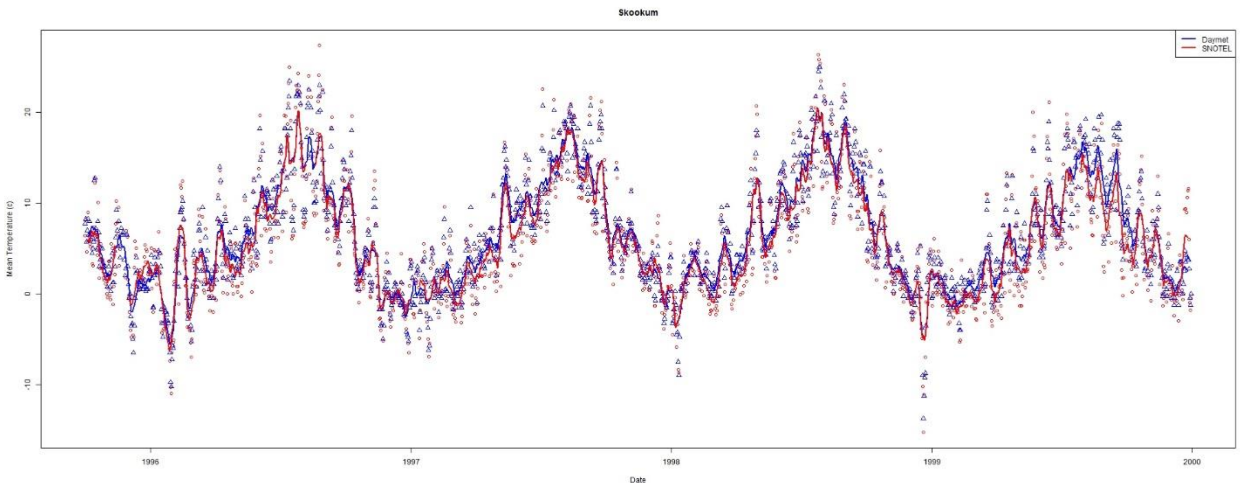


Figure 23 Daily air temperature, Daymet vs Skookum

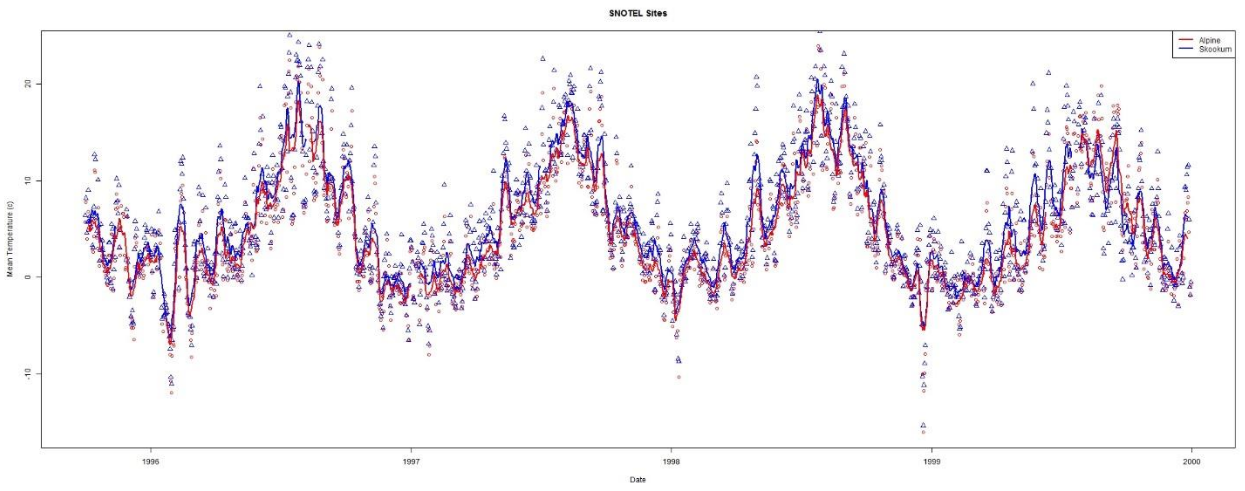


Figure 24 Daily air temperature, Skookum vs Alpine Meadows SNOTEL

References

PRISM Climate Group (2013) Descriptions of PRISM Spatial Climate Datasets for the Conterminous United States. In. PRISM Climate Group, Oregon State University, Corvallis, Oregon, http://prism.oregonstate.edu/documents/PRISM_history_jun2013.pdf

Thornton, P.E., M.M. Thornton, B.W. Mayer, N. Wilhelmi, Y. Wei, R. Devarakonda, and R.B. Cook. 2014. Daymet: Daily Surface Weather Data on a 1-km Grid for North America, Version 2. Data set. Available on-line [https://daac.ornl.gov] from Oak Ridge National Laboratory Distributed Active Archive Center, Oak Ridge, Tennessee, USA.

United States Department of Agriculture, National Water and Climate Center. SNOTEL Precipitation Products. http://www.wcc.nrcs.usda.gov/snow/snotel-precip-reports.html

Appendix

```
# Regression model between the weather variables (temperature and precipitation) and the
# physiographic variables
climate<-read.csv("D:/Temp/Tolt/ToltClimate.csv",header=TRUE)

climVar<-names(climate)

# Retrieve the site names

for (i in climVar) {
  print(i)
}
ln<-length(names(climate))
# Simple plotting of each site, temp vs date
# Modify output/create output directory for plots

m <- list()

for (i in 1:ln) {
  fit <- lm(climate[,i] ~ climate[,25])
  fit <- lm(X01_ppt~ele, data = climate)
  fitAtt<-summary(fit)
  models$climVar[i]<-fitAtt

  directory="D:/Temp/Tolt/"
  r2<- = round(fitAtt$adj.r.squared)
  outname<-paste(directory,climVar[i],".jpg",sep="")

  jpeg(filename=outname,width=675,height=675,quality=100)
  #newdata<-subset(wqdata, Site==i)
  ylabel <- paste(climVar[i]," (mm)", sep="")
  plot(climate[,i] ~ climate[,25],ylab=ylabel,xlab="Elevation (m)")
  abline(coef(fit)[1:2],col="red")

  ## rounded coefficients for better output
  cf <- round(coef(fit), 3)
  ## sign check to avoid having plus followed by minus for negative coefficients
  eq <- paste0(climVar[i], " = ", cf[1],
             ifelse(sign(cf[2])==1, " + ", " - "), abs(cf[2]), " ele    R^2: ", round(r2,3)
  mtext(eq, 3, line=1)
  dev.off()
}


# SNOTEL_Daymet_Climate_Comparison.r
# Author: Paul Pettus
# Purpose:
# Compare Daymet modeled and SNOTEL ground station
# observations of precipitation and mean air temperature
#



##################################################################
# Plots 1995 1999

# Skookum
inFile<-"E:/SNOTEL/NRCS_Daymet_Daily_Skookum_Alpine_1995_1999.csv"

dfdmNRCS = read.csv(inFile,header=TRUE,sep=",")
dfdmNRCS$Date <- as.Date(dfdmNRCS$Date, format="%m/%d/%Y")


jpegFile<-"E:/SNOTEL/Plots/SNOTEL_Skookum_Daymet_Daily_TAVG_1995_1999.jpg"
```

```
jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Mean Temperature (c)"
xLab <- "Date"
mainTitle <- "Skookum"

plot(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Skook_SN,type="p",col="red",pch=1,xlab=xLab, ylab=yLab,main=mainTitle)
points(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Skook_DM,type="p",col="blue",pch=2)

f21 <- rep(1/11,11)

y_sym2 <- filter(dfdmNRCS$TAVG_c_Skook_DM, f21, sides=2)
lines(dfdmNRCS$Date, y_sym2, col="blue",lwd=2)
y_sym <- filter(dfdmNRCS$TAVG_c_Skook_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym, col="red",lwd=2)
legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

jpegFile <- "E:/SNOTEL/Plots/SNOTEL_Daymet_Skookum_PPT_Accumulation_1995_1999.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Precipitaion Accumulation (mm)"
xLab <- "Date"
mainTitle <- "Skookum"

plot(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Skook_SN,type="l",col="red",pch=1, xlab=xLab, ylab=yLab,main=mainTitle)
lines(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Skook_DM,type="l",col="blue",pch=2)

legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

# Alpine Meadows
jpegFile<-"E:/SNOTEL/Plots/SNOTEL_Alpine_Daymet_Daily_TAVG_1995_1999.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Mean Temperature (c)"
xLab <- "Date"
mainTitle <- "Alpine Meadows"

plot(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Alp_SN,type="p",col="red",pch=1,xlab=xLab, ylab=yLab,main=mainTitle)
points(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Alp_DM,type="p",col="blue",pch=2)

f21 <- rep(1/11,11)

y_sym2 <- filter(dfdmNRCS$TAVG_c_Alp_DM, f21, sides=2)
lines(dfdmNRCS$Date, y_sym2, col="blue",lwd=2)
y_sym <- filter(dfdmNRCS$TAVG_c_Alp_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym, col="red",lwd=2)
legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

jpegFile <- "E:/SNOTEL/Plots/SNOTEL_Daymet_Alpine_PPT_Accumulation_1995_1999.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Precipitaion Accumulation (mm)"
xLab <- "Date"
mainTitle <- "Alpine Meadows"

plot(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Alp_SN,type="l",col="red",pch=1, xlab=xLab, ylab=yLab,main=mainTitle)
```

```
lines(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Alp_DM,type="l",col="blue",pch=2)

legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

# Apline Meadows vs Skookum
jpegFile<-"E:/SNOTEL/Plots/SNOTEL_Skookum_Alpine_Daily_TAVG_1995_1999.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Mean Temperature (c)"
xLab <- "Date"
mainTitle <- "SNOTEL Sites"

plot(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Alp_SN,type="p",col="red",pch=1,xlab=xLab, ylab=yLab,main=mainTitle)
points(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Skook_SN,type="p",col="blue",pch=2)

f21 <- rep(1/11,11)

y_sym2 <- filter(dfdmNRCS$TAVG_c_Skook_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym2, col="blue",lwd=2)
y_sym <- filter(dfdmNRCS$TAVG_c_Alp_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym, col="red",lwd=2)
legend('topright', legend=c("Alpine","Skookum"), lwd=c(2.5,2.5), col=c("red","blue"))

dev.off()

jpegFile <- "E:/SNOTEL/Plots/SNOTEL_Skookum_Alpine_PPT_Accumulation_1995_1999.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Precipitaion Accumulation (mm)"
xLab <- "Date"
mainTitle <- "SNOTEL Sites"

plot(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Alp_SN,type="l",col="red",pch=1, xlab=xLab, ylab=yLab,main=mainTitle)
lines(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Skook_SN,type="l",col="blue",pch=2)

legend('topright', legend=c("Alpine","Skookum"), lwd=c(2.5,2.5), col=c("red","blue"))

dev.off()

##########################################################3
# Plots 1994 2015

inFile<-"E:/SNOTEL/NRCS_Daymet_Daily_Skookum_Alpine_1995_2015.csv"

dfdmNRCS = read.csv(inFile,header=TRUE,sep=",")
dfdmNRCS$Date <- as.Date(dfdmNRCS$Date, format="%m/%d/%Y")


jpegFile<-"E:/SNOTEL/Plots/SNOTEL_Skookum_Daymet_Daily_TAVG_1995_2015.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Mean Temperature (c)"
xLab <- "Date"
mainTitle <- "Skookum"

plot(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Skook_SN,type="p",col="red",pch=1,xlab=xLab, ylab=yLab,main=mainTitle)
points(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Skook_DM,type="p",col="blue",pch=2)

f21 <- rep(1/11,11)

y_sym2 <- filter(dfdmNRCS$TAVG_c_Skook_DM, f21, sides=2)
```

```
lines(dfdmNRCS$Date, y_sym2, col="blue",lwd=2)
y_sym <- filter(dfdmNRCS$TAVG_c_Skook_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym, col="red",lwd=2)
legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

jpegFile <- "E:/SNOTEL/Plots/SNOTEL_Daymet_Skookum_PPT_Accumulation_1995_2015.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Precipitaion Accumulation (mm)"
xLab <- "Date"
mainTitle <- "Skookum"

plot(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Skook_SN,type="l",col="red",pch=1, xlab=xLab, ylab=yLab,main=mainTitle)
lines(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Skook_DM,type="l",col="blue",pch=2)

legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

# Alpine Meadows
jpegFile<-"E:/SNOTEL/Plots/SNOTEL_Alpine_Daymet_Daily_TAVG_1995_2015.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Mean Temperature (c)"
xLab <- "Date"
mainTitle <- "Alpine Meadows"

plot(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Alp_SN,type="p",col="red",pch=1,xlab=xLab, ylab=yLab,main=mainTitle)
points(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Alp_DM,type="p",col="blue",pch=2)

f21 <- rep(1/11,11)

y_sym2 <- filter(dfdmNRCS$TAVG_c_Alp_DM, f21, sides=2)
lines(dfdmNRCS$Date, y_sym2, col="blue",lwd=2)
y_sym <- filter(dfdmNRCS$TAVG_c_Alp_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym, col="red",lwd=2)
legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

jpegFile <- "E:/SNOTEL/Plots/SNOTEL_Daymet_Alpine_PPT_Accumulation_1995_2015.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Precipitaion Accumulation (mm)"
xLab <- "Date"
mainTitle <- "Alpine Meadows"

plot(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Alp_SN,type="l",col="red",pch=1, xlab=xLab, ylab=yLab,main=mainTitle)
lines(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Alp_DM,type="l",col="blue",pch=2)

legend('topright', legend=c("Daymet","SNOTEL"), lwd=c(2.5,2.5), col=c("blue","red"))

dev.off()

# Apline Meadows vs Skookum
jpegFile<-"E:/SNOTEL/Plots/SNOTEL_Skookum_Alpine_Daily_TAVG_1995_2015.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Mean Temperature (c)"
xLab <- "Date"
```

```
mainTitle <- "SNOTEL Sites"

plot(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Alp_SN,type="p",col="red",pch=1,xlab=xLab, ylab=yLab,main=mainTitle)
points(dfdmNRCS$Date,dfdmNRCS$TAVG_c_Skook_SN,type="p",col="blue",pch=2)

f21 <- rep(1/11,11)

y_sym2 <- filter(dfdmNRCS$TAVG_c_Skook_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym2, col="blue",lwd=2)
y_sym <- filter(dfdmNRCS$TAVG_c_Alp_SN, f21, sides=2)
lines(dfdmNRCS$Date, y_sym, col="red",lwd=2)
legend('topright', legend=c("Alpine","Skookum"), lwd=c(2.5,2.5), col=c("red","blue"))

dev.off()

jpegFile <- "E:/SNOTEL/Plots/SNOTEL_Skookum_Alpine_PPT_Accumulation_1995_2015.jpg"

jpeg(filename=jpegFile,width=2000,height=800,quality=100)

yLab <- "Precipitaion Accumulation (mm)"
xLab <- "Date"
mainTitle <- "SNOTEL Sites"

plot(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Alp_SN,type="l",col="red",pch=1, xlab=xLab, ylab=yLab,main=mainTitle)
lines(dfdmNRCS$Date,dfdmNRCS$acc_PPT_mm_Skook_SN,type="l",col="blue",pch=2)

legend('topright', legend=c("Alpine","Skookum"), lwd=c(2.5,2.5), col=c("red","blue"))

dev.off()


# tolt_plot_8-11-2016.r
# Author: Paul Pettus
# Purpose:
# Create Daymet 30 year normal regressions of Mean air temperature
# and precipitaion in the Greater Tolt river basin, WA
#

#climate<-read.csv("D:/Temp/Mashel/MashelClimate.csv",header=TRUE)

climate<-read.csv("C:/Temp/Daymet_Tolt/Tolt_Beyond_Climate.csv",header=TRUE)
#climate<-read.csv("C:/Temp/Daymet_Tolt/Tolt_Beyond_Climate_Mean_Elevation.csv",header=TRUE)

#directory="C:/Temp/Daymet_Tolt/Daymet_Regressions_Beyond_Over_1000m/"
#directory="C:/Temp/Daymet_Tolt/Daymet_Regressions_Beyond_Under_1000m/"
#directory="C:/Temp/Daymet_Tolt/Daymet_Regressions_Elevation_Mean_Over_500m/"
#directory="C:/Temp/Daymet_Tolt/Daymet_Regressions_Elevation_Mean_Under_500m/"

directory="C:/Temp/Daymet_Tolt/Daymet_Regressions_Beyond/"
#directory="C:/Temp/Daymet_Tolt/Daymet_Regressions_Elevation_Mean/"

climVar<-names(climate)
head(climate)
####################################################################
# Climate subsets
#climate <- subset(climate, ele >= 1000)
#climate <- subset(climate, ele < 1000)
#climate2 <- subset(climate, ele >= 500)
#climate2 <- subset(climate, ele < 500)

head(climate)

# Calapooia site temperature date times
# Retrieve the site names

for (i in climVar) {
  print(i)
}
ln<-length(names(climate))
# Simple plotting of each site, temp vs date
# Modify output/create output directory for plots


################################################
models <- list()

for (i in 1:ln) {

  fit <- lm(climate[,i] ~ climate[,25])
  #fit <- lm(X01_ppt~ele, data = climate)
  val<-climVar[i]
  models[[val]]<-fit

  fitAtt<-summary(fit)

  #directory="C:/Temp/Daymet_Tolt/Daymet_Regressions/"
  r2<- round(fitAtt$adj.r.squared,3)
  #print(r2)
  #
  outname<-paste(directory,climVar[i],".jpg",sep="")

  jpeg(filename=outname,width=675,height=675,quality=100)
  #newdata<-subset(wqdata, Site==i)
  ylabel <- paste(climVar[i]," (mm)", sep="")
  xlabel<- "Elevation (m)"
  plot(climate[,i] ~ climate[,25],ylab=ylabel,xlab=xlabel)
  abline(coef(fit)[1:2],col="red")

  ## rounded coefficients for better output
  cf <- round(coef(fit), 3)
 # r2 <- round(r2, 2)

  ## sign check to avoid having plus followed by minus for negative coefficients
  eq <- paste0(climate[i], " = ", cf[1],
           ifelse(sign(cf[2])==1, " + ", " - "), abs(cf[2]), " ele    R^2: ", r2)
  eq
  mtext(eq, 3, line=1)
  dev.off()
}

#outFile2<-"C:/Temp/Daymet_Tolt/Daymet_Regressions_Beyond_Over_1000m/Tolt_Model_Summary_Beyond_Over_1000m.csv"
#outFile2<-"C:/Temp/Daymet_Tolt/Daymet_Regressions_Beyond_Under_1000m/Tolt_Model_Summary_Beyond_Under_1000m.csv"
#outFile2<-
"C:/Temp/Daymet_Tolt/Daymet_Regressions_Elevation_Mean_Over_500m/Tolt_Model_Summary_Elevation_Mean_Over_500m.csv"
#outFile2<-
"C:/Temp/Daymet_Tolt/Daymet_Regressions_Elevation_Mean_Under_500m/Tolt_Model_Summary_Elevation_Mean_Under_500m.csv"


outFile2<-"C:/Temp/Daymet_Tolt/Daymet_Regressions_Beyond/Tolt_Model_Summary_Beyond.csv"
#outFile2<-"C:/Temp/Daymet_Tolt/Daymet_Regressions_Elevation_Mean/Tolt_Model_Summary_Beyond_Elevation_Mean.csv"

for (i in 1:length(models)) {
  modname <-names(models[i])
  cf<-coef(models[[i]])
  rsq<-summary(models[[i]])$adj.r.squared
  cat(
    paste(modname,cf[1],cf[2],rsq,sep=',')
    ,'\n',file=outFile2,append=TRUE)
}
```

D.8 | Cell Data Writer Configuration

Overview *(Tutorial D.8 – Cell Data Writer Configuration)*

This document explains how to add optional Cell Data Writers to a VELMA simulation. Cell Data Writers allow you to gather daily simulation results for a specific cell. For example, this is useful if need to compare VELMA soil moisture predictions to observed data recorded by a soil moisture sensor at a particular location.

Cell Data Writers Allow You to Gather and Report Simulation Results for a Specific Grid Location

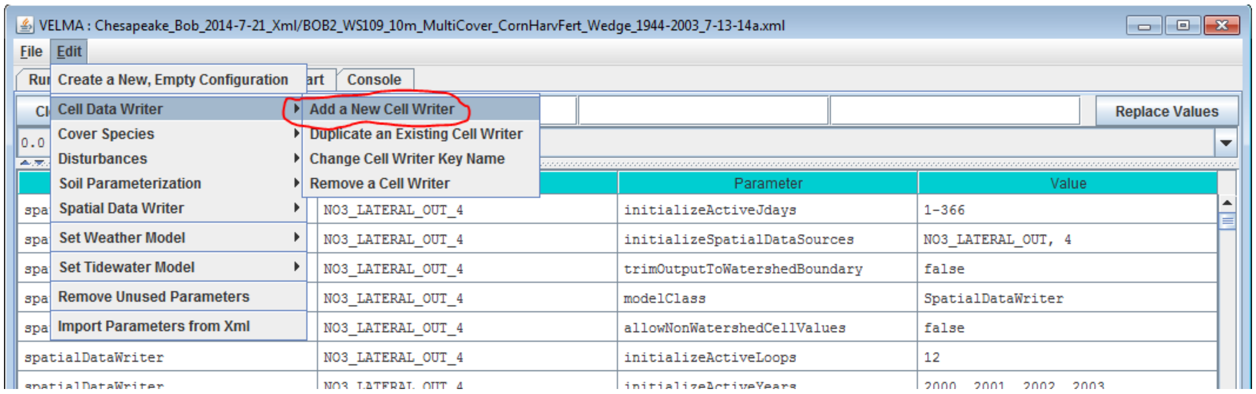
The VELMA simulator automatically provides daily simulation results for various values (e.g. Leaf Biomass), however these daily results are values that are computed by summing individual cell values and then dividing the sum by the number of cells in the simulation’s delineated watershed.

To gather and report daily simulation results for a single, specific cell, you need to add a Cell Data Writer parameterization for that cell to the simulation configuration.

Adding a Cell Data Writer to a Simulation Configuration

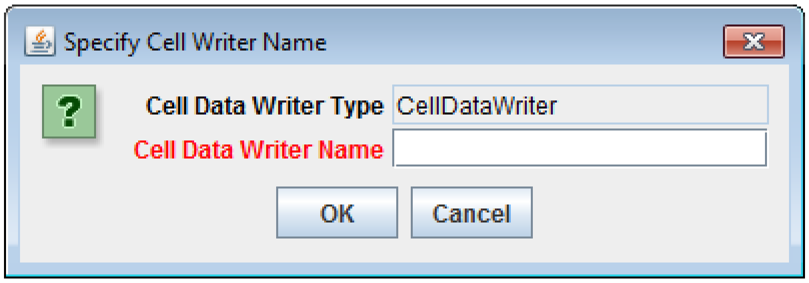
Cell Data Writers are optional. Simulation configurations contain zero instances of them by default. When added, they do not change the simulation computations – they are only involved in reporting results.

To add a Cell Data Writer to your simulation configuration:
Click the “Edit” → “Cell Data Writer” → “Add a New Cell Writer” menu item.


VELMA : Chesapeake_Bob_2014-7-21_Xml/BOB2_WS109_10m_MultiCover_CornHarvFert_Wedge_1944-2003_7-13-14a.xml
File
Edit
Create a New, Empty Configuration
Cell Data Writer
Cover Species
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Remove Unused Parameters
Import Parameters from Xml
Add a New Cell Writer
Duplicate an Existing Cell Writer
Change Cell Writer Key Name
Remove a Cell Writer
Console
Replace Values

		Parameter	Value
spa	NO3_LATERAL_OUT_4	initializeActiveJdays	1-366
spa	NO3_LATERAL_OUT_4	initializeSpatialDataSources	NO3_LATERAL_OUT, 4
spa	NO3_LATERAL_OUT_4	trimOutputToWatershedBoundary	false
spa	NO3_LATERAL_OUT_4	modelClass	SpatialDataWriter
spa	NO3_LATERAL_OUT_4	allowNonWatershedCellValues	false
spatialDataWriter	NO3_LATERAL_OUT_4	initializeActiveLoops	12
spatialDataWriter	NO3_LATERAL_OUT_4	initializeActiveYears	2000, 2001, 2002, 2003

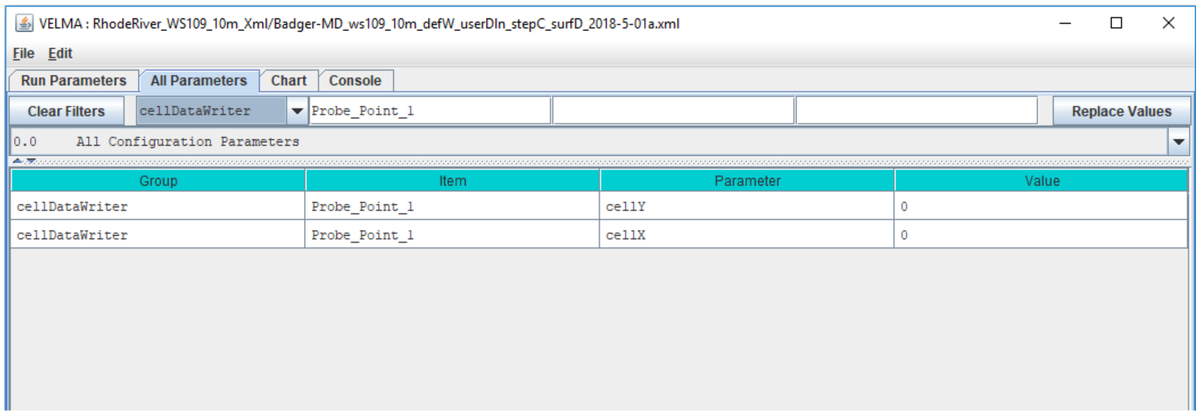
Clicking “Add a New Cell Writer” opens the Cell Writer Name dialog, which looks like this:

screenshot 

Enter a name for your new Cell Data Writer and click "OK".
The name must be unique (i.e. no other Cell Data Writer already specified for this simulation configuration can share the name you specify) and we recommend avoiding whitespace and punctuation characters (e.g. "(" and ")"). An acceptable example name (assuming it's not already in use by another Cell Data Writer might be "Probe_Point_1" or maybe "Outlet_Cell".

Configuring a Cell Data Writer's Parameters

After you click OK in the Cell Data Writer's naming dialog, the VELMA GUI adds the Cell Data Writer to the simulation configuration, and sets the All Parameters tab's filters to display only the parameters of the newly-added Cell Data Writer.
Assuming we named our new Cell Data Writer "Probe_Point_1" and clicked OK, the VELMA GUI would look like this afterwards:


VELMA : RhodeRiver_WS109_10m_Xml/Badger-MD_ws109_10m_defW_userDIn_stepC_surfD_2018-5-01a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
cellDataWriter
Probe_Point_1
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
cellDataWriter	Probe_Point_1	cellY	0
cellDataWriter	Probe_Point_1	cellX	0

Notice, in passing, that the configuration outline does *not* automatically get set by the GUI. In the example screen capture above, it displays "0.0 All Configuration Parameters" – which is not what is being displayed in the parameterization table. This behavior is harmless.

A Cell Data Writer has only 2 parameters, but they must both be set correctly: *none are optional*.

Set the `cellX` parameter's value to the X-coordinate (i.e. column value) of the cell you want data reported for. The valid range is from 0 (the leftmost column) to (number of columns – 1).

Set the `cellY` parameter's value to the Y-coordinate (i.e. row value) of the cell you want data reported for. The valid range is from 0 (the topmost row) to (number of rows – 1).

Cell Data Writers Report Cell-Specific Results for Every Simulation Step

Previous versions of JVelma required each Cell Data Writer parameterization to explicitly state which Julian Days of the year the Cell Data Writer instance should report data for. The parameter that specified this ("`initializeActiveJdays`") has been removed. Each Cell Data Writer instance parameterized for a simulation now implicitly "knows" to report data for each step of the simulation.

Cell Data Writer Output is A Comma-Separated Values File

The VELMA simulator creates 1 .csv file for each valid Cell Data Writer parameterization in the simulation configuration .xml file.

Cell Data Writer output files are written to the Results Data Location folder. Their filenames always begin with the prefix "Cell_" and contain the linear index, x-coordinate and y-coordinate of the cell they contain data for (e.g. `Cell_i2873_x35_y33.csv`).

Each row of a Cell Data Writer output file contains results data for a specific loop, year and Julian day during the simulation run. The file's columns contain the specific results. The header row of the file specifies the contents of each column. Note that some columns contain data that never varies (e.g. the `DEM_Elevation(m)` column reports the cell's elevation – which is always the same value.)

Cell Data Writers and Spatial Data Writers are Different, but Complementary

A Cell Data Writer reports all the results data available for a specific cell on user-specified days.
A Spatial Data Writer reports a specific result for all the cells in the simulation watershed on user-specified days. Both are optional for simulation runs.

D.9 | Set Up VELMA biomass Outputs as Inputs to BlueSky

Overview *(Tutorial D.9 – Set Up VELMA biomass Outputs as Inputs to Bluesky)*

Bluesky is a smoke emissions simulator that is designed to predict the direction and chemistry of smoke produced by forest and rangeland fires. To do this, Bluesky requires spatially-explicit inputs describing the quantity and quality of fuel loads. VELMA is designed to provide this information.

This document describes how to convert VELMA spatial fuel load output to input for BlueSky.

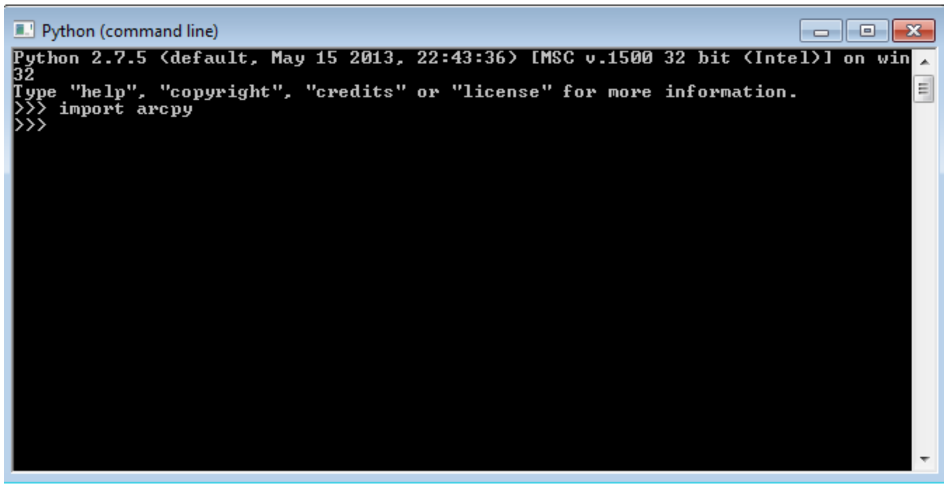
Software Requirements

To run the algorithm that converts VELMA output to input for the Bluesky smoke emissions simulator, you will need the following:

1. **Python version 2.x: Current release is 2.7.11**
2. **Python package 'ArcPy' (Requires ArcGIS license)**
3. **Velma2Bluesky Python Algorithm**

1. Python comes pre-packaged within ArcGIS, so it is likely you already have it installed on your computer. For example, mine is installed currently here: C:\Python27\ArcGIS10.2\python.exe. Check to see if you have Python installed before installing a new version. If Python is not installed, you can obtain a copy here: https://www.python.org/download/releases/2.7/ Note that Python 2.7 is currently considered safe for use on U.S. EPA network and non-network computers.

2. ArcPy is a python package that comes pre-installed with ArcGIS. You can test that you have access to ArcPy by opening your Python executable and typing 'import arcpy' into the command line. If you receive no error, then you have ArcPy installed appropriately.



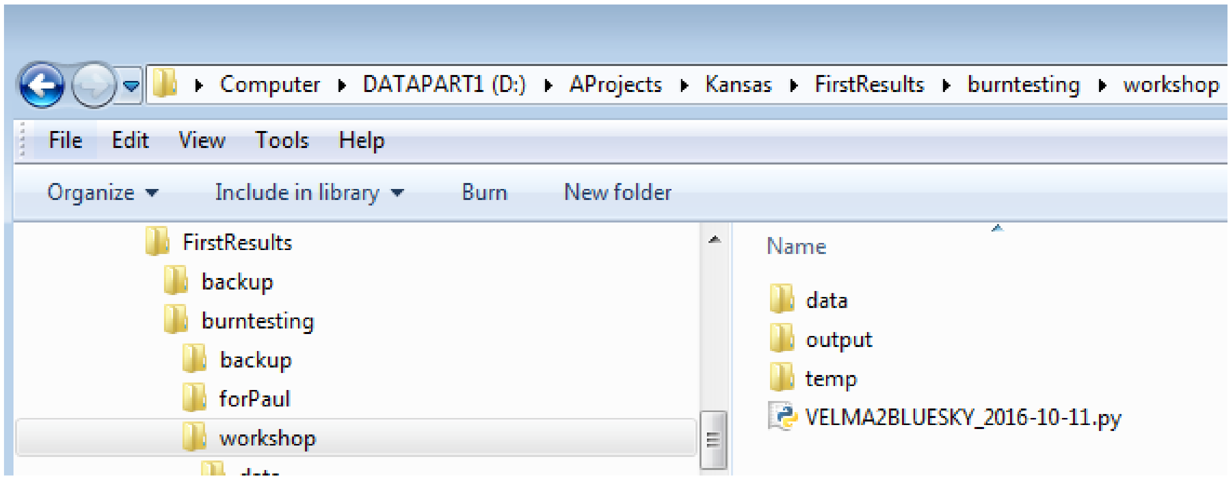
```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> import arcpy
>>>
```

If you have problems importing ArcPy, visit http://pro.arcgis.com/en/pro-app/arcpy/get-started/importing-arcpy.htm

3. **Velma2Bluesky Python Algorithm:** This file is called VELMA2BLUESKY.py and can be run using a command line command: "python C:\Path\To\VELMA2BLUESKY.py". It can also be edited directly with any text editor (e.g., Notepad, EMACS)

Steps for running the VELMA2BLUESKY.py script

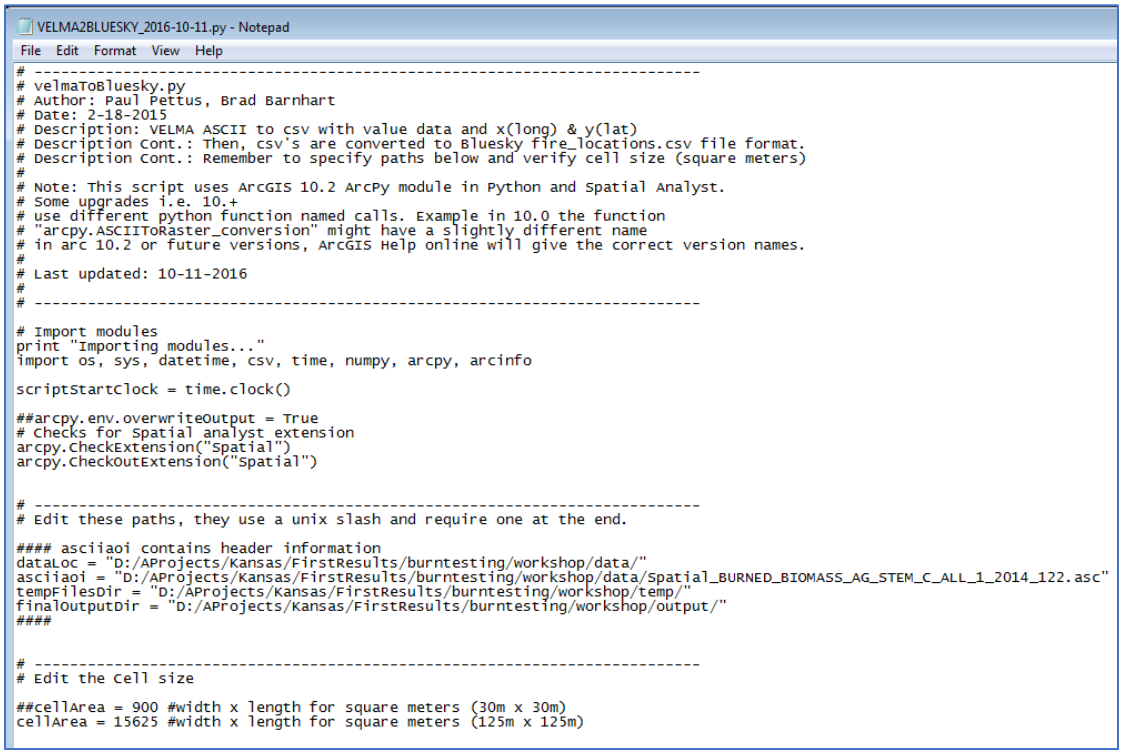
1. First, navigate to the directory that contains the Bluesky script. In this example, the script is called VELMA2BLUESKY_2016-10-11.py, but you can rename it if you would like.



Computer ▸ DATAPART1 (D:) ▸ AProjects ▸ Kansas ▸ FirstResults ▸ burntesting ▸ workshop

data
output
temp
VELMA2BLUESKY_2016-10-11.py

2. You can open the file with any text editor, as shown here:

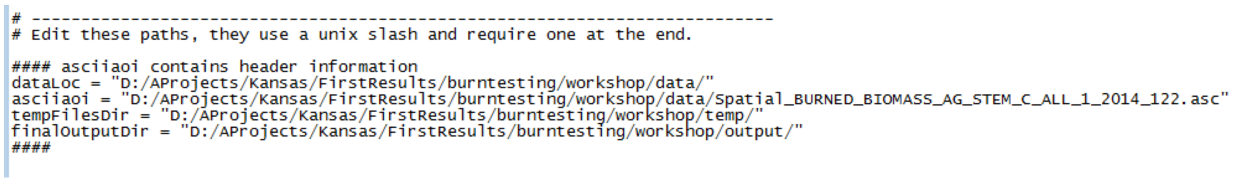

VELMA2BLUESKY_2016-10-11.py - Notepad
File Edit Format View Help

```
# ---------------------------------------------------------------------------
# velmaToBluesky.py
# Author: Paul Pettus, Brad Barnhart
# Date: 2-18-2015
# Description: VELMA ASCII to csv with value data and x(long) & y(lat)
# Description Cont.: Then, csv's are converted to Bluesky fire_locations.csv file format.
# Description Cont.: Remember to specify paths below and verify cell size (square meters)
#
# Note: This script uses ArcGIS 10.2 ArcPy module in Python and Spatial Analyst.
# Some upgrades i.e. 10.+
# use different python function named calls. Example in 10.0 the function
# "arcpy.ASCIIToRaster_conversion" might have a slightly different name
# in arc 10.2 or future versions, ArcGIS Help online will give the correct version names.
#
# Last updated: 10-11-2016
#
# ---------------------------------------------------------------------------

# Import modules
print "Importing modules..."
import os, sys, datetime, csv, time, numpy, arcpy, arcinfo

scriptStartClock = time.clock()

##arcpy.env.overwriteOutput = True
# Checks for Spatial analyst extension
arcpy.CheckExtension("Spatial")
arcpy.CheckOutExtension("Spatial")


# ---------------------------------------------------------------------------
# Edit these paths, they use a unix slash and require one at the end.

#### asciiaoi contains header information
dataLoc = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/data/"
asciiaoi = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/data/Spatial_BURNED_BIOMASS_AG_STEM_C_ALL_1_2014_122.asc"
tempFilesDir = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/temp/"
finalOutputDir = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/output/"
####


# ---------------------------------------------------------------------------
# Edit the cell size

##cellArea = 900 #width x length for square meters (30m x 30m)
cellArea = 15625 #width x length for square meters (125m x 125m)
```

3. You will need to change the paths within the script to make it work on your system.
   - dataLoc = This should be the path to your data directory that contains VELMA output files.
   - asciiaoi = This path points to any single .asc file in the output data directory. The script uses the header of this file to geo-locate the positions of burns.
   - tempFilesDir = This directory must be created to store temporary processing files.
   - finalOutputDir = This directory will hold the Bluesky inputs after the script is run.

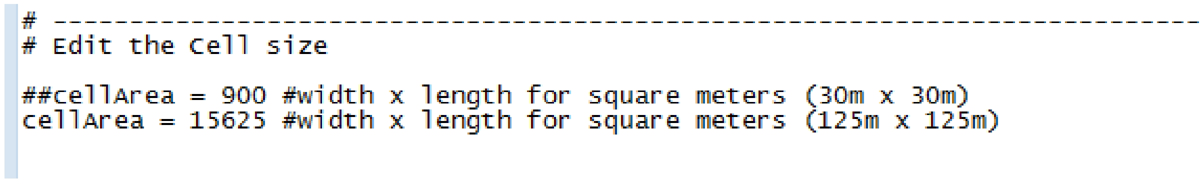
NOTE: The tempFilesDir and finalOutputDir must exist and they must be empty for the script to work properly. If you receive an error, make sure that these paths are correct and that there are no files in the tempFilesDir or finalOutputDir.

Below is an example of a proper specification of paths:



```
# ---------------------------------------------------------------------------
# Edit these paths, they use a unix slash and require one at the end.

#### asciiaoi contains header information
dataLoc = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/data/"
asciiaoi = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/data/Spatial_BURNED_BIOMASS_AG_STEM_C_ALL_1_2014_122.asc"
tempFilesDir = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/temp/"
finalOutputDir = "D:/AProjects/Kansas/FirstResults/burntesting/workshop/output/"
####
```

4. You will also need to set the cellArea in the script. You obtain the cellArea variable by squaring the original size of your VELMA cell. That is, if you have a 30m grid, you will input a cellArea = 30x30 = 900.

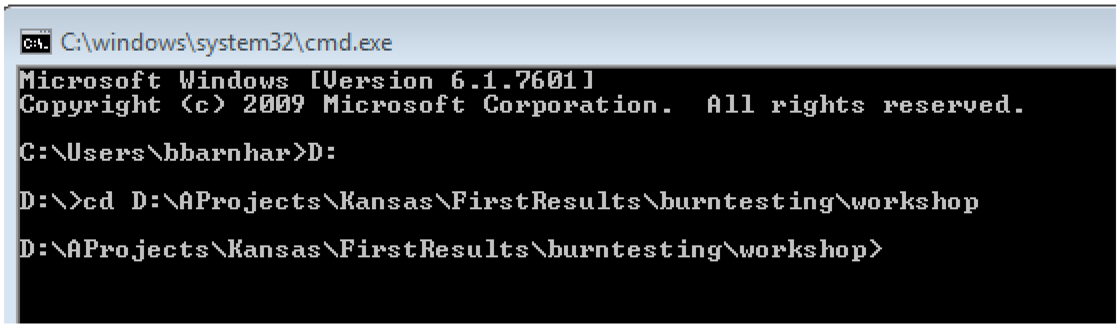


```
# ---------------------------------------------------------------------------
# Edit the Cell size

##cellArea = 900 #width x length for square meters (30m x 30m)
cellArea = 15625 #width x length for square meters (125m x 125m)
```

5. To run the script, start by opening a command window. You can also open the script file in IDLE or another Python user environment if you know how to do this.

For the command line method, on Windows, click Start -> Search for Programs or Files -> type in "cmd" and click on cmd.exe.

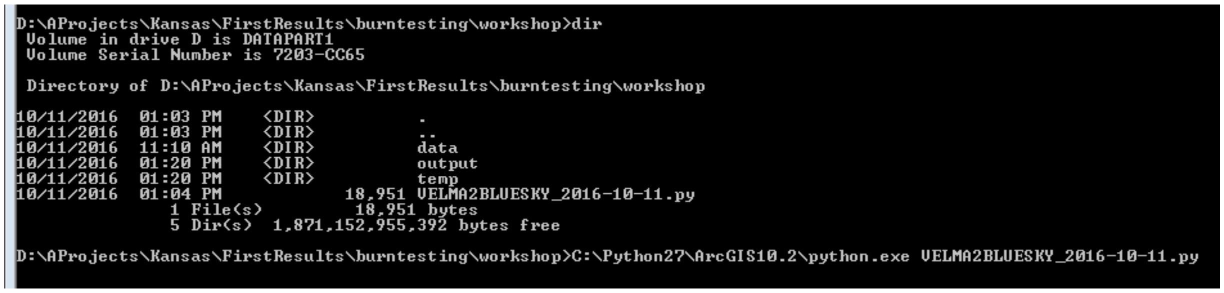
Then, navigate to the directory where you have your VELMA files using the 'cd' command.



```
C:\windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\bbarnhar>D:

D:\>cd D:\AProjects\Kansas\FirstResults\burntesting\workshop

D:\AProjects\Kansas\FirstResults\burntesting\workshop>
```

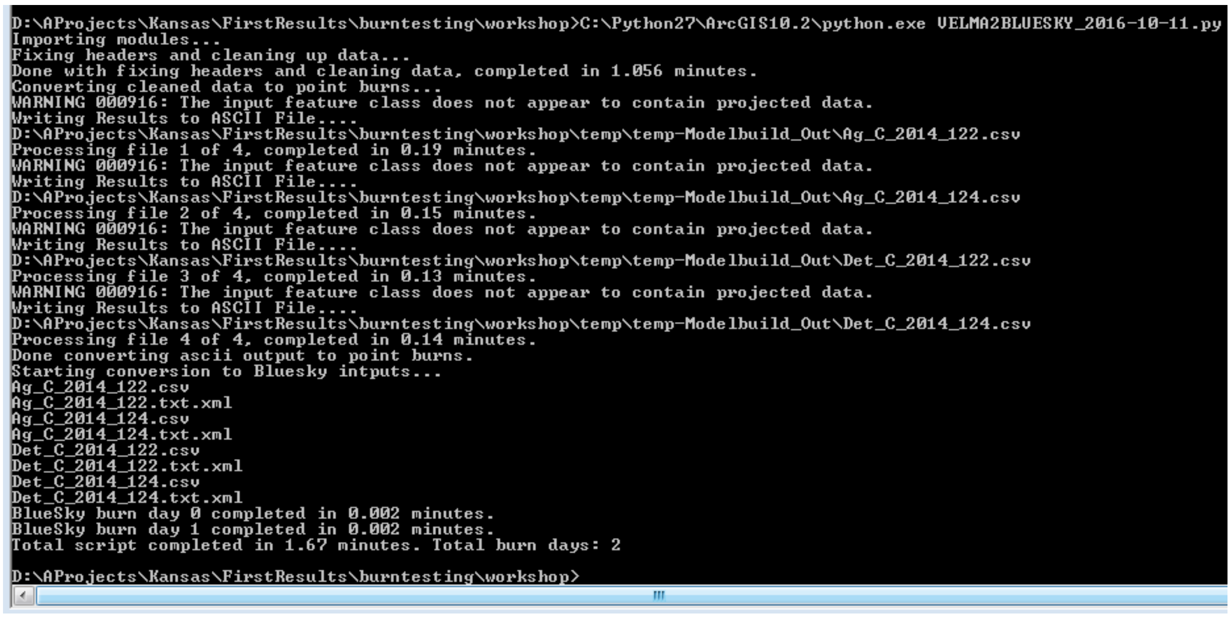
6. Finally, use your python.exe file to run the script using the following command: C:\path\to\python.exe D:\Path\to\VELMA2BLUESKY_2016_10-11.py

An example is shown below. Note that my python.exe program is located at C:\Python27\ArcGIS10.2\python.exe



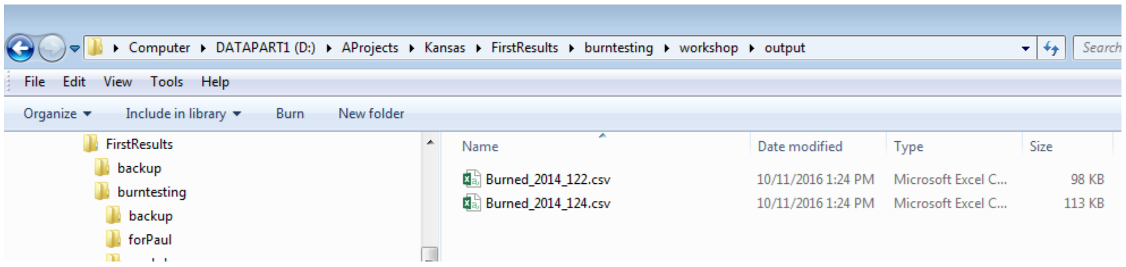
```
D:\AProjects\Kansas\FirstResults\burntesting\workshop>dir
 Volume in drive D is DATAPART1
 Volume Serial Number is 7203-CC65

 Directory of D:\AProjects\Kansas\FirstResults\burntesting\workshop

10/11/2016  01:03 PM    <DIR>          .
10/11/2016  01:03 PM    <DIR>          ..
10/11/2016  11:10 AM    <DIR>          data
10/11/2016  01:20 PM    <DIR>          output
10/11/2016  01:20 PM    <DIR>          temp
10/11/2016  01:04 PM            18,951 VELMA2BLUESKY_2016-10-11.py
               1 File(s)         18,951 bytes
               5 Dir(s)  1,871,152,955,392 bytes free

D:\AProjects\Kansas\FirstResults\burntesting\workshop>C:\Python27\ArcGIS10.2\python.exe VELMA2BLUESKY_2016-10-11.py
```

The program will begin and will display the following output. If you receive an error, check the error carefully and determine whether or not you have properly configured your paths and made sure that you have the proper software configuration (ArcGIS with Spatial Analyst, ArcPy, etc.)



```
D:\AProjects\Kansas\FirstResults\burntesting\workshop>C:\Python27\ArcGIS10.2\python.exe VELMA2BLUESKY_2016-10-11.py
Importing modules...
Fixing headers and cleaning up data...
Done with fixing headers and cleaning data, completed in 1.056 minutes.
Converting cleaned data to point burns...
WARNING 000916: The input feature class does not appear to contain projected data.
Writing Results to ASCII File....
D:\AProjects\Kansas\FirstResults\burntesting\workshop\temp\temp-Modelbuild_Out\Ag_C_2014_122.csv
Processing file 1 of 4, completed in 0.19 minutes.
WARNING 000916: The input feature class does not appear to contain projected data.
Writing Results to ASCII File....
D:\AProjects\Kansas\FirstResults\burntesting\workshop\temp\temp-Modelbuild_Out\Ag_C_2014_124.csv
Processing file 2 of 4, completed in 0.15 minutes.
WARNING 000916: The input feature class does not appear to contain projected data.
Writing Results to ASCII File....
D:\AProjects\Kansas\FirstResults\burntesting\workshop\temp\temp-Modelbuild_Out\Det_C_2014_122.csv
Processing file 3 of 4, completed in 0.13 minutes.
WARNING 000916: The input feature class does not appear to contain projected data.
Writing Results to ASCII File....
D:\AProjects\Kansas\FirstResults\burntesting\workshop\temp\temp-Modelbuild_Out\Det_C_2014_124.csv
Processing file 4 of 4, completed in 0.14 minutes.
Done converting ascii output to point burns.
Starting conversion to Bluesky intputs...
Ag_C_2014_122.csv
Ag_C_2014_122.txt.xml
Ag_C_2014_124.csv
Ag_C_2014_124.txt.xml
Det_C_2014_122.csv
Det_C_2014_122.txt.xml
Det_C_2014_124.csv
Det_C_2014_124.txt.xml
BlueSky burn day 0 completed in 0.002 minutes.
BlueSky burn day 1 completed in 0.002 minutes.
Total script completed in 1.67 minutes. Total burn days: 2

D:\AProjects\Kansas\FirstResults\burntesting\workshop>
```

7. Check your output directory for output files.



Computer ▸ DATAPART1 (D:) ▸ AProjects ▸ Kansas ▸ FirstResults ▸ burntesting ▸ workshop ▸ output

Name	Date modified	Type	Size
Burned_2014_122.csv	10/11/2016 1:24 PM	Microsoft Excel C...	98 KB
Burned_2014_124.csv	10/11/2016 1:24 PM	Microsoft Excel C...	113 KB

File Edit View Tools Help
Organize Include in library Burn New folder
FirstResults
backup
burntesting
backup
forPaul
Search

8. You can view the output files using a text or spreadsheet editor.

Scripts

```
# ---------------------------------------------------------------------------
# velmaToBluesky.py
# Author: Paul Pettus, Brad Barnhart
# Date: 2-18-2015
# Description: VELMA ASCII to csv with value data and x(long) & y(lat)
# Description Cont.: Then, csv's are converted to Bluesky fire_locations.csv file format.
# Description Cont.: Remember to specify paths below and verify cell size (square meters)
#
# Note: This script uses ArcGIS 10.2 ArcPy module in Python and Spatial Analyst.
# Some upgrades i.e. 10.+
# use different python function named calls. Example in 10.0 the function
# "arcpy.ASCIIToRaster_conversion" might have a slightly different name
# in arc 10.2 or future versions, ArcGIS Help online will give the correct version names.
#
# Last updated: 01-31-2017
#
# ---------------------------------------------------------------------------

# Import modules
```

```
print "Importing modules..."
import os, sys, datetime, csv, time, numpy, arcpy, arcinfo

scriptStartClock = time.clock()

##arcpy.env.overwriteOutput = True
# Checks for Spatial analyst extension
arcpy.CheckExtension("Spatial")
arcpy.CheckOutExtension("Spatial")


# ----------------------------------------------------------------------------
# Edit these paths, they use a unix slash and require one at the end.

#### asciiaoi contains header information
dataLoc =
"L:/Priv/CORFiles/Projects/Velma/Kansas_Project/VELMA_SETUPS_WORKSHOP/Velma_Outputs/KS_Flint_Cent
ral_Small_125m_wbFixup_AST2_2017-01-30_historicBurnPlusGraze/"
asciiaoi =
"L:/Priv/CORFiles/Projects/Velma/Kansas_Project/VELMA_SETUPS_WORKSHOP/Velma_Outputs/KS_Flint_Cent
ral_Small_125m_wbFixup_AST2_2017-01-
30_historicBurnPlusGraze/Spatial_BURNED_BIOMASS_AG_STEM_C_ALL_1_2000_88.asc"
tempFilesDir =
"L:/Priv/CORFiles/Projects/Velma/Kansas_Project/VELMA_SETUPS_WORKSHOP/Velma_Outputs/temp/"
finalOutputDir =
"L:/Priv/CORFiles/Projects/Velma/Kansas_Project/VELMA_SETUPS_WORKSHOP/Velma_Outputs/toolOutput/"
####


# ----------------------------------------------------------------------------
# Edit the Cell size

##cellSize = 900 #width x length for square meters (30m x 30m)
cellSize = 15625 #width x length for square meters (125m x 125m)


#-----------------------------------------------------------------------------
print "Starting data clean up..."

outDirStep1 = tempFilesDir + "temp-HeaderFixed/"


# ----------------------------------------------------------------------------
# Read in DEM file, and pull out header
# ----------------------------------------------------------------------------
readFile = open(asciiaoi)

header = readFile.readline() #ncols
header += readFile.readline() #nrows
header += readFile.readline() #xllcorner
header += readFile.readline() #yllcorner
header += readFile.readline() #cellsize
header += readFile.readline() #NODATA_value
readFile.close()
# ----------------------------------------------------------------------------

# ----------------------------------------------------------------------------
# Create temporary directory in tempFilesDir to store fixed headers.
# ----------------------------------------------------------------------------

startClock = time.clock()

## Create a list of all input files
asciiList = []
for files in os.listdir(dataLoc):
    if files.endswith(".asc"):
        testBit = files in asciiList
        if testBit == False:
            asciiList.append(files)

for files in asciiList:
    filename = dataLoc + files
    filesLower = files.lower()

    ## Load in all four above ground biomass files
    if "biomass_ag" in filesLower:
        fileName, fileExtension = os.path.splitext(files)
        Next_poolName, Next_yearID, Next_julianID = fileName.rsplit('_',2)
        stemAgArray = numpy.loadtxt(filename, skiprows=6, dtype= numpy.float64)
        leafAgName = dataLoc+"Spatial_BURNED_BIOMASS_LEAF_C_ALL_1_" + Next_yearID + "_" +
Next_julianID +".asc"
        leafAgArray = numpy.loadtxt(leafAgName, skiprows=6, dtype= numpy.float64)
        stemDetName = dataLoc+"Spatial_BURNED_DETRITUS_AG_STEM_C_ALL_1_" + Next_yearID + "_" +
Next_julianID +".asc"
        stemDetArray = numpy.loadtxt(stemDetName, skiprows=6, dtype= numpy.float64)
        leafDetName = dataLoc+"Spatial_BURNED_DETRITUS_LEAF_C_ALL_1_" + Next_yearID + "_" +
Next_julianID +".asc"
        leafDetArray = numpy.loadtxt(leafDetName, skiprows=6, dtype= numpy.float64)

        row, col = stemAgArray.shape
        sumAgArray = numpy.zeros((row,col))
        sumDetArray = numpy.zeros((row,col))

        ## Data clean-up: This section ensures that near-zero VELMA output is
        ## properly accounted for.
        for i in xrange(row):
            for j in xrange(col):
                cellStem = stemAgArray[i,j]
                cellLeaf = leafAgArray[i,j]
                cellDetStem = stemDetArray[i,j]
                cellDetLeaf = leafDetArray[i,j]

                # Check for negative values -> to 0
                if cellStem < 0 and cellStem != -9999:
                    cellStem = 0

                if cellLeaf < 0 and cellLeaf != -9999:
                    cellLeaf = 0

                if cellDetStem < 0 and cellDetStem != -9999:
                    cellDetStem = 0

                if cellDetLeaf < 0 and cellDetLeaf != -9999:
                    cellDetLeaf = 0

                # Check for values less than 0.001 -> to 0.001
                if cellStem <= 0.001 and cellStem != 0 and cellStem != -9999:
                    cellStem = 0.001

                if cellLeaf <= 0.001 and cellLeaf != 0 and cellLeaf != -9999:
                    cellLeaf = 0.001

                if cellDetStem <= 0.001 and cellDetStem != 0 and cellDetStem != -9999:
                    cellDetStem = 0.001

                if cellDetLeaf <= 0.001 and cellDetLeaf != 0 and cellDetLeaf != -9999:
                    cellDetLeaf = 0.001

                # Check if one cell has a value then all cells must have a value
                if cellStem >= 0.001 or cellLeaf >= 0.001 or cellDetStem >= 0.001 or cellDetLeaf
>= 0.001:

                    if cellStem < 0.001:
                        cellStem = 0.001

                    if cellLeaf < 0.001:
                        cellLeaf = 0.001

                    if cellDetStem < 0.001:
                        cellDetStem = 0.001

                    if cellDetLeaf < 0.001:
                        cellDetLeaf = 0.001

                if cellStem == -9999:
                    sumAgArray[i,j] = -9999.0
                    sumDetArray[i,j] = -9999.0
                else:
                    # Multiply by 1000 because ArcGIS converts raster to shapefile as int not
float
                    sumAgArray[i,j] = (1000*(cellStem + cellLeaf))
                    sumDetArray[i,j] = (1000*(cellDetStem + cellDetLeaf))
        # Writes out live and dead biomass files
        asciiOutAgfn = outDirStep1 + "Ag_C_" + Next_yearID + "_" + Next_julianID +".asc"
        asciiOutDetfn = outDirStep1 + "Det_C_" + Next_yearID + "_" + Next_julianID +".asc"
        if not os.path.exists(os.path.dirname(asciiOutAgfn)):
            os.makedirs(os.path.dirname(asciiOutAgfn))
        f = open(asciiOutAgfn, "w")
        f.write(header)
        numpy.savetxt(f, sumAgArray, fmt="%f")
        f.close()
        f2 = open(asciiOutDetfn, "w")
        f2.write(header)
        numpy.savetxt(f2, sumDetArray, fmt="%f")
        f2.close()
endClock = time.clock()
totalTime = round(((endClock - startClock)/60),3)
print("Done with fixing headers and zero biomass, completed in " + str(totalTime) + " minutes.")

# ---------------------------------------------------------------------------
# Done with fixing headers.
# ---------------------------------------------------------------------------

# ---------------------------------------------------------------------------
# Do not edit these paths. These are temporary directories that
# will be created within the tempFilesDir specified above.
# ---------------------------------------------------------------------------
inDirStep2 = outDirStep1
filesDir = tempFilesDir + "temp-GisFiles/"
outDirStep2 = tempFilesDir + "temp-Modelbuild_Out/"
# ---------------------------------------------------------------------------
print "Starting ascii's to point burn conversions..."
# Process ascii's into point burn data needed for BlueSky input
## Create a list of all input files
asciiList = []
for files in os.listdir(inDirStep2):
    if files.endswith(".asc"):
        testBit = files in asciiList
        if testBit == False:
            asciiList.append(files)

count = 0

for files in asciiList:
    ## Create output file names
    startClock = time.clock()
    fileName, fileExtension = os.path.splitext(files)
    asciifn = inDirStep2 + files
    tiffn = filesDir + fileName + ".tif"
    reclassfn = filesDir + fileName + "_reclass.tif"
    reclassShape = filesDir + fileName + "_reclass.shp"
    zonalfn = filesDir + fileName + "_zonal.tif"
    rastercalc = filesDir + fileName + "_int.tif"
    zonalShape = filesDir + fileName + "_zonal.shp"
    pointfn = filesDir + fileName + ".shp"
    pointGeofn = filesDir + fileName + "_nad83.shp"
    csvfn = outDirStep2 + fileName + ".csv"

    ## Create/check for output dirs
    if not os.path.exists(os.path.dirname(tiffn)):
        os.makedirs(os.path.dirname(tiffn))
    if not os.path.exists(os.path.dirname(csvfn)):
        os.makedirs(os.path.dirname(csvfn))
    # Process: ASCII to Raster
    arcpy.ASCIIToRaster_conversion(asciifn, tiffn, "FLOAT")

    # Process: Define Projection
    arcpy.DefineProjection_management(tiffn,
"PROJCS['NAD_1983_UTM_Zone_14N',GEOGCS['GCS_North_American_1983',DATUM['D_North_American_1983',SP
HEROID['GRS_1980',6378137.0,298.257222101]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199
433]],PROJECTION['Transverse_Mercator'],PARAMETER['False_Easting',500000.0],PARAMETER['False_Nort
hing',0.0],PARAMETER['Central_Meridian',-
99.0],PARAMETER['Scale_Factor',0.9996],PARAMETER['Latitude_Of_Origin',0.0],UNIT['Meter',1.0]]")

    # Process: Reclassify
    arcpy.gp.Reclassify_sa(tiffn, "Value", "0 0;0 100000000 1", reclassfn, "DATA")

    # Process: Raster to Polygon
    arcpy.RasterToPolygon_conversion(reclassfn, reclassShape, "NO_SIMPLIFY", "VALUE")

    # Process: Zonal Statistics
    arcpy.gp.ZonalStatistics_sa(reclassShape, "ID", tiffn, zonalfn, "SUM", "DATA")

    # Process: Raster math convert to int
    myZonalRaster = arcpy.Raster(zonalfn)
##    intRaster = arcpy.sa.Int(myZonalRaster*100000)  ## moved to data clean up section
    intRaster = arcpy.sa.Int(myZonalRaster)
    intRaster.save(rastercalc)

    # Process: Raster to Polygon
    arcpy.RasterToPolygon_conversion(rastercalc, zonalShape, "NO_SIMPLIFY", "VALUE")

    # Process: Add Geometry Attributes
    arcpy.AddGeometryAttributes_management(zonalShape, "AREA", "", "SQUARE_METERS", "")

    # Process: Feature To Point
    arcpy.FeatureToPoint_management(zonalShape, pointfn, "INSIDE")

    # Process: Project
    arcpy.Project_management(pointfn, pointGeofn,
"GEOGCS['GCS_North_American_1983',DATUM['D_North_American_1983',SPHEROID['GRS_1980',6378137.0,298
.257222101]],PRIMEM['Greenwich',0.0],UNIT['Degree',0.0174532925199433]]", "", "")

    # Process: Export Feature Attribute to ASCII  Process: Add XY Coordinates
    arcpy.ExportXYv_stats(pointGeofn, "FID;ID;GRIDCODE;POLY_AREA", "COMMA", csvfn,
"ADD_FIELD_NAMES") # fix this?

    endClock = time.clock()
    totalTime = round(((endClock - startClock)/60),2)
    count = count + 1
    print("Processed VELMA Burn day layer " + str(count) + " completed in " + str(totalTime) + "
minutes.")

print "Done with converting VELMA BURN ASCII OUTPUT TO CSV FILES!"


# ----------------------------------------------------------------------------
# Done with making csv's. Now convert csv to BlueSky format.
# ----------------------------------------------------------------------------
inDirStep3 = outDirStep2
outDir = finalOutputDir
# ----------------------------------------------------------------------------

print "Starting conversion to Bluesky intputs..."

##BLB PROPOSED CHANGE TO GET CSV FILES
asciiList = []
for files in os.listdir(inDirStep3):
    print files
    if files.endswith(".csv"):
        testBit = files in asciiList
        if testBit == False:
            asciiList.append(files)
```

```
#for files in os.listdir(inDirStep3):
#    if files.endswith(".csv"):
#        asciiList.append(files)
##BLB proposed CHANGED TO GET CSV FILES
########################################

## Bluesky header format
header = ["date_time","id","type","latitude","longitude","area","fuel_1hr","fuel_10hr",
          "fuel_100hr","fuel_1khr","fuel_10khr","fuel_gt10khr","shrub","grass","rot",
          "duff","litter","canopy"]

colNumber = 8

# id VELMA + unique id
# File name
idNumber = 0
count = 0


for files in asciiList:
    startClock = time.clock()
    filesLower = files.lower()
    fileName, fileExtension = os.path.splitext(files)
    Next_poolName, Next_yearID, Next_julianID = fileName.rsplit('_',2)
    date  = datetime.datetime(int(Next_yearID),1,1) + datetime.timedelta((int(Next_julianID)-1))
    outDate = date.strftime('%Y%m%d') + "0000Z"
    fileList = []

    if "ag_c_" in filesLower:
        dateList = []
        idList = []
        typeList = []
        areaList = []
        fuel_1hrList = []
        fuel_10hrList = []
        fuel_100hrList = []
        fuel_1khrList = []
        fuel_10khrList = []
        fuel_gt10khrList = []
        shrubList = []
        rotList = []
        litterList = []
        canopyList = []

        typeBurn = "RX"
        idNumber = 0
        fuel_1hr = 0
        fuel_10hr = 0
        fuel_100hr = 0
        fuel_1khr = 0
        fuel_10khr = 0
        fuel_gt10khr = 0
        shrub = 0
        rot = 0
        litter = 0
        canopy = 0

        agArray = numpy.loadtxt(inDirStep3+files, dtype='float', skiprows=1,delimiter=',')
        agFilePathName = inDirStep3+files

        detFilePathName = agFilePathName.replace("Ag_C","Det_C")
        detArray = numpy.loadtxt(detFilePathName, dtype='float', skiprows=1,delimiter=',')

        selectAgArray = agArray[agArray[:,4] > 0]
        outAgArray = numpy.multiply(selectAgArray[:,4],1)
        divAgArray = numpy.divide(outAgArray,1000)  # Adjustment from int(ArcGIS) back to float
        row = divAgArray.shape
        for i in xrange(row[0]):        # Values below 1g/m^2 are returned to zero from pre-
process VELMA-ascii
            cellValue = divAgArray[i]
            if cellValue < 1:
                divAgArray[i] = 0

        sumAgArray = numpy.multiply(2,divAgArray)
        sumAgArray = numpy.multiply(0.00446089,sumAgArray)
        cellCount = numpy.divide(selectAgArray[:,5],cellSize)
        sumAgArray = numpy.divide(sumAgArray,cellCount)
        dataAgList = list(sumAgArray)
        #Notes on conversions. Multiply by 2 to go from Carbon to Biomass
        #Then, divide by m2 area in a single cell.
        #Then, convert g/m2 to tons/acre

        selectDetArray = detArray[detArray[:,4] > 0]
        outDetArray = numpy.multiply(selectDetArray[:,4],1)
        divDetArray = numpy.divide(outDetArray,1000)  # Adjustment from int(ArcGIS) back to float
        row = divDetArray.shape
        for i in xrange(row[0]):        # Values below 1g/m^2 are returned to zero from pre-
process VELMA-ascii
            cellValue = divDetArray[i]
            if cellValue < 1:
                divDetArray[i] = 0

        sumDetArray = numpy.multiply(2,divDetArray)
        sumDetArray = numpy.multiply(0.00446089,sumDetArray)
        cellCount = numpy.divide(selectAgArray[:,5],cellSize)
        sumDetArray = numpy.divide(sumDetArray,cellCount)
        dataDetList = list(sumDetArray)
        #Notes on conversions. Multiply by 2 to go from Carbon to Biomass
        #Then, divide by m2 area in a single cell.
        #Then, convert g/m2 to tons/acre

        inRow2,inCol = selectAgArray.shape

        for i in range(inRow2):
            idName = "VELMA_"+str(idNumber)+"_"+outDate
            idList.append(idName)
            dateList.append(outDate)
            typeList.append(typeBurn)
##            areaList.append(area)
            fuel_1hrList.append(fuel_1hr)
            fuel_10hrList.append(fuel_10hr)
            fuel_100hrList.append(fuel_100hr)
            fuel_1khrList.append(fuel_1khr)
            fuel_10khrList.append(fuel_10khr)
            fuel_gt10khrList.append(fuel_gt10khr)
            shrubList.append(shrub)
            rotList.append(rot)
            litterList.append(litter)
            canopyList.append(canopy)
            idNumber = idNumber + 1

##date_time,id,type,latitude,longitude,area,fuel_1hr,fuel_10hr,fuel_100hr,fuel_1khr,fuel_10khr,fu
el_gt10khr,shrub,grass,rot,duff,litter,canopy

        fileList.append(dateList)
        fileList.append(idList)
        fileList.append(typeList)
        fileList.append(list(selectDetArray[:,1]))
        fileList.append(list(selectDetArray[:,0]))
        areaList = numpy.divide(selectDetArray[:,5],4046.863) # Converts Square meters to acre
        fileList.append(list(areaList))
        fileList.append(fuel_1hrList)
        fileList.append(fuel_10hrList)
        fileList.append(fuel_100hrList)
        fileList.append(fuel_1khrList)
        fileList.append(fuel_10khrList)
        fileList.append(fuel_gt10khrList)
        fileList.append(shrubList)
        fileList.append(dataAgList)
        fileList.append(rotList)
        fileList.append(dataDetList)
        fileList.append(litterList)
        fileList.append(canopyList)

        flippedList = zip(*fileList)


############################################################
## Add two leading zero("00") digits to julian days 1-9
## Add one leading zero("0") digits to julian days 10-99
        if len(Next_julianID) == 1:
            Next_julianID = "00" + Next_julianID

        if len(Next_julianID) == 2:
            Next_julianID = "0" + Next_julianID
############################################################

        outfn = outDir + "Burned_" + Next_yearID + "_" + Next_julianID + ".csv"

        if not os.path.exists(os.path.dirname(outfn)):
            os.makedirs(os.path.dirname(outfn))

        ## Write out Bluesky input file for burn day
        outfile = open(outfn,'w')
        out = csv.writer(outfile, delimiter=',', lineterminator='\n')
        out.writerow(header)
        for i in flippedList:
            out.writerow(i)
        outfile.close()
        endClock = time.clock()
        totalTime = round(((endClock - startClock)/60),3)
        print("ASCII BlueSky Burn day" + str(count) + " completed in " + str(totalTime) + "
minutes.")
        count = count + 1

############################################################
## Create a master list merged burn csv
        if count == 1:
            outfnMerge = outDir + "Burned_Merge_Complete.csv"
            outfileMerge = open(outfnMerge,'w')
            out2 = csv.writer(outfileMerge, delimiter=',', lineterminator='\n')
            out2.writerow(header)
            for i in flippedList:
                out2.writerow(i)
        elif count > 1 and count < (len(asciiList)/2):
            for i in flippedList:
                out2.writerow(i)
        else:
            for i in flippedList:
                out2.writerow(i)
            outfileMerge.close()

############################################################

sciptEndClock = time.clock()
totalTime = round(((sciptEndClock - scriptStartClock)/60),2)
print("Total script completed in " + str(totalTime) + " minutes. Total burn days: " + str(count))
```

D.10 | Default Tidewater Model Configuration

Overview *(Tutorial D.10 – Default Tidewater Model Configuration)*

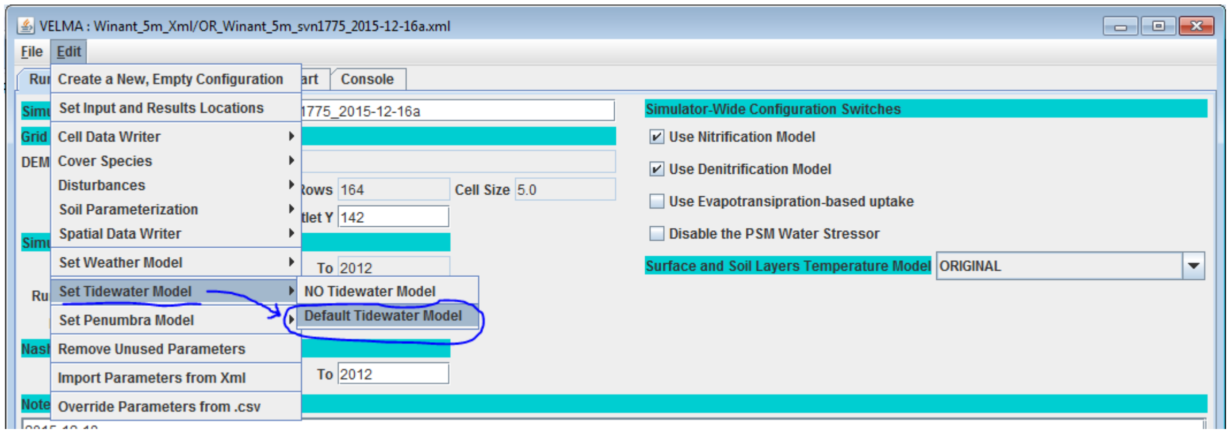
This document describes how to configure VELMA's Tidewater model for simulating the per cell daily amount of tidewater (millimeters of water) and marine nitrogen infiltration (grams nitrogen / meter^2) for coastal watershed applications.

Warning: This model is still under early development.
The functionality described here is likely to change or at least be modified in the future.

The Default Tidewater Model provides a way to mimic water and nitrogen deposition due to tidal overtopping of cells within a VELMA simulation's delineated watershed. The model is currently very simple, and subject to development changes.

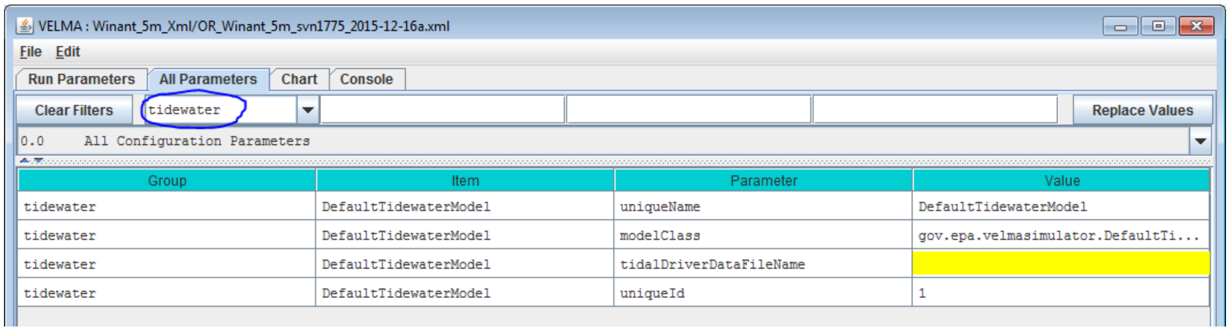
You Must Explicitly Set the Tidewater Model for use by a simulation configuration.

In the VELMA GUI click Edit → "Set Tidewater Model" → "Default Tidewater Model") as shown below:


VELMA : Winant_5m_Xml/OR_Winant_5m_svn1775_2015-12-16a.xml
File
Edit
Create a New, Empty Configuration
Set Input and Results Locations
Cell Data Writer
Cover Species
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Set Penumbra Model
Remove Unused Parameters
Import Parameters from Xml
Override Parameters from .csv
NO Tidewater Model
Default Tidewater Model
Console
Simulator-Wide Configuration Switches
Use Nitrification Model
Use Denitrification Model
Use Evapotranspiration-based uptake
Disable the PSM Water Stressor
Surface and Soil Layers Temperature Model
ORIGINAL
Rows 164
Cell Size 5.0
tlet Y 142
To 2012
To 2012

Note the "NO Tidewater Model" menu option – clicking it removes the Default Tidewater Model from your simulation configuration.

After click-setting the Default Tidewater Model, the VELMA GUI should shift your view to the model's parameters in the "All Parameters" tab:


VELMA : Winant_5m_Xml/OR_Winant_5m_svn1775_2015-12-16a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
tidewater
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
tidewater	DefaultTidewaterModel	uniqueName	DefaultTidewaterModel
tidewater	DefaultTidewaterModel	modelClass	gov.epa.velmasimulator.DefaultTi...
tidewater	DefaultTidewaterModel	tidalDriverDataFileName	
tidewater	DefaultTidewaterModel	uniqueId	1

Notice that the Group selector has "tidewater" selected; you can always return the parameters table to display only tidewater parameters by clicking "Clear Filters" and then selecting "tidewater" in the drop-down list of the Group selector.

You Must Provide the Default Tidewater Model with a Driver Data File

The Default Tidewater Model needs daily values for the following data:

- Highest High Tide Elevation (for the 24-hour period of the day, in meters)
- Lowest Low Tide elevation (for the 24-hour period of the day, in meters)
- Seawater NO_3 Concentration Coefficient (in micro-mols N / liter)
- Seawater NH_4 Concentration Coefficient (in micro-mols N / liter)
- Seawater DON Concentration Coefficient (in micro-mols N / liter)

Set the name of the `tidalDriverDataFileName` to the name of a file that provides the above values.

The name (or fully-qualified path + name) you provide must be a file of comma-separated values (.csv). when presented with a file name without a path, the VELMA simulation engine's initialization code will assume the file is located in the directory specified by the `inputDataLocationRootName/inputDataLocationDirName` path.

Each row of the .csv file specifies one day's worth of tidewater driver data in the following field layout:

```
YEAR, JDAY, HIGH_TIDE, LOW_TIDE, NO3_K, NH4_K, DON_K
```

The year and jday fields must be integers, and the year field must be a full four-digit year. VELMA assumes that first row in the file will have `YEAR` = forcing_start and `JDAY` = 1, and that the last row in the file will have `YEAR` = forcing_end and `JDAY` = 365 (or 366 if forcing_end is a leap year). VELMA further assumes that there will be rows of data for every YEAR, JDAY combination in-between. That is, there can be no missing days in the driver data file.

The `YEAR, JDAY, HIGH_TIDE` and `LOW_TIDE` values are all **required**, but any or all of the nitrogen concentration coefficients are optional and may be left blank. Any nitrogen concentration coefficient value left blank defaults to a value of zero.

Here are the first four rows of a `tidalDriverDataFileName` .csv driver file as an example:

```
YEAR,  JDAY,  HIGH_TIDE,  LOW_TIDE,   NO3_K, NH4_K, DON_K
2008,     1,  2.1858504,  0.38152240,   7.0,   5.0,   3.0
2008,     2,  2.3199558,  0.36018744,   7.0,   5.0,   3.0
2008,     3,  2.5942624,  0.40285736,   7.0,   5.0,   3.0
2008,     4,  2.8990475,  0.60096769,   7.0,   5.0,   3.0
[ . . .]
```

Note that in the above example, whitespace has been introduced after the comma-separators to make the fields easier to identify. In practice, such whitespace is *not* required and *not* encouraged.

The example above contains a header row, which is acceptable but not required. However, if the header row is present, it must start with an alphabet character, not a numeral.

The Default Tidewater Model's Tidewater and Nitrogen Infiltration Amounts are Reported by Cell and Spatial Data Writers

When a DefaultTidewaterModel is part of a simulation configuration, any cell data writers specified automatically include columns reporting the following daily amounts for its specified cell location:

- Tidewater Added (in millimeters of water)
- NO_3-infiltration (in grams nitrogen / meter^2)
- NH_4-infiltration (in grams nitrogen / meter^2)
- DON-infiltration (in grams nitrogen / meter^2)

Spatial Data Writers may be configured for the following Spatial Data keywords:

- `Tidewater`
- `TidalNo3Infiltration`
- `TidalDonInfiltration`
- `TidalNh4Infiltration`
- `TotalTidalNInfiltration`

Configure Spatial Data Writers with the keywords exactly as shown above (the mixed case must also be exact).

The Tidewater values reported in spatial data represent amounts of water (in millimeters) added "as if" they were part of the cell's rainfall on the given day.

All the nitrogen infiltration values reported in Spatial Data output files specified by the above keywords are in units of grams nitrogen / meter^2 and represent amounts added at each cell in the map by the specific nitrogen infiltration type. The TotalTidalNInfiltration values are the sum (NO_3 + NH_4 + DON) of the cells nitrogen infiltration addition for the day.

A simulation configuration that includes Spatial Data Writer parameterizations for any of the above, but does not include the DefaultTidewaterModel parameterization should not crash, but all any spatial data based on the DefaultTidewaterModel will be all zeros.

A Summary Description of the Default Tidewater Model's Behavior and Effects

When a DefaultTidewaterModel is set for a simulation configuration, and provided with valid parameters from the tidal driver data file, the daily water balance code asks the tidewater model instance for a tidewater amount for each cell whose elevation falls below the HIGH_TIDE value for that day, and adds the (zero or more millimeters) resulting tidewater amount to the cell's total rain amount for that day. The tidewater amount added is simply:

```
Tidewater amount = (HIGH_TIDE * 1000) – (cell elevation * 1000)
```

The multiplication by 1000 converts meters to millimeters.

After water balance computes tidewater amounts, the plant soil model (Psm4dLsr) code asks the tidewater model for the NO_3, NH_4 and DON infiltration amounts of every cell. Cells with zero tidewater addition automatically receive zero infiltration amounts, and all cells will receive zero infiltration (regardless of their day's tidewater amount) for any concentration coefficient that was zero (either explicitly or through omission of the concentration constant from the file) in the driver data for that day.

Cells with tidewater amounts > zero, and with non-zero infiltration have their nitrogen infiltration computed as:

```
n-infiltration amount
    = (concentration coefficient * 0.000014) * tidewater amount
```

The conversion factor of 0.000014 converts from the coefficient's units of micro-mols/liter to grams/meter^2.

Each of the three tidal infiltration amounts is computed by the equation above, but the amounts are assigned to different pools within VELMA.

- The NO_3 infiltration amount is added to the NO3 pool's layer 1 amount.
- The NH_4 infiltration amount is added to the NH4 pool's layer 1 amount.
- The DON infiltration amount is added to the DON pool's layer 1 amount.

The amounts are added to their respective data pools prior to any daily nitrification and denitrification calculations. For example, the NO_3 amount is added to the pool prior to denitrification, so the cell's layer 1 denitrification amount for the day will be computed from the NO3 pool's original amount + its tidal infiltration amount.

E.1 | Mapping Surface Layer Permeabilities

Overview *(Tutorial E.1 – Mapping Surface Layer Permeabilities)*

Permeability of surface layers is an important property affecting infiltration of rainfall and surface runoff and accompanying dissolved contaminants and nutrients.

This document describes how to parameterize VELMA's non-soil surface layer to represent the degree of permeability of overlying human structures (roads, roofs, compacted soils, etc.) and natural features (surficial bedrock, etc.) to water and dissolved chemicals.

1.0. Permeability Basics

The VELMA simulator engine allows specification of a spatial data map (grid) with permeability fractions as its cell values. When specified, the grid's permeability fractions are included in determining how much of a cell's total water inflow (from rain, snow melt, lateral surface movement, etc.) penetrates into the first layer of its layered soil column.

Permeability fraction values must be in the range [0.0, 1.0], where 0.0 is completely impermeable (no water penetrates from surface to soil layer 1) and 1.0 is completely permeable (all water penetrates from surface to soil layer 1).

1.1. Permeability Values Are Not Required by the VELMA Simulator

Permeability is *not* the sole factor determining how much water penetrates from surface into the first soil layer. The soil layer's saturation, porosity and other parameters determine the primary amount of water transferred from surface to the layer. A cells permeability fraction is a secondary factor that may further reduce the primary amount.
When a VELMA simulation configuration does not specify a permeability grid, the VELMA simulator engine behaves as if every cell has a permeability of 1.0 – i.e. no secondary reduction of the penetration water amount occurs.

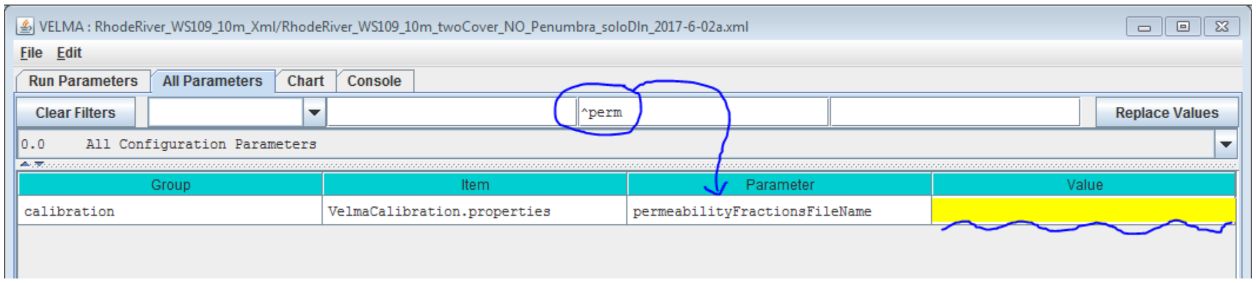
1.2. Specify Permeability Values for Each Simulation Grid Cell in a Permeability Map File

The permeability grid map must be a Grid ASCII (.asc) file with the same header data as the DEM .asc file specified for the simulation configuration. The permeability grid file's data values must be floating point numbers, each one within the range [0.0, 1.0].

1.3. Specify the Simulation's Permeability Map File via a Configuration Parameter

Set the permeability fractions for the cells of the grid by providing the name of the permeability .asc map file as the value of the `permeabilityFractionsFileName` Calibration parameter.

You can find this parameter by entering "^perm" (without the double-quote marks) into the middle filter text box on JVelma's "All Parameters" panel, like so:


VELMA : RhodeRiver_WS109_10m_Xml/RhodeRiver_WS109_10m_twoCover_NO_Penumbra_soloDln_2017-6-02a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
^perm
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
calibration	VelmaCalibration.properties	permeabilityFractionsFileName	

The Value for the `permeabilityFractionsFileName` parameter must be the name of an existing, accessible, valid Grid ASCII (.asc) file containing permeability fractions. If the Value provided is a fully-qualified path, the simulator will look for the .asc file at that location, with that filename. If the Value provided is a partial path or only a file name, the simulator will look for a file with specified name relative to the location specified by the `inputDataLocationRootName/inputDataLocationDirName` parameters.

2.0. How to Establish Pervious Surfaces for a VELMA Application

2.1. Data Sources

Washington Transportation, Proprietary Roads (Statewide) - https://data-wadnr.opendata.arcgis.com/documents/wadnr-active-roads-download/about

  - Washington State Department of Natural Resources (WA DNR)
- National Elevation Dataset (30m) (DEM) – Flat processed by JPDEM
  - U.S. Geological Survey, The National Map, 2015, 3DEP products and services

2.2. Data Processing

A fractional pervious surface map is created by using a combination of impervious layer map(s) and an area of interest (AOI) DEM. The pervious surface maps permeability percent values will range from 0 to 1, where 0 completely impervious and 1 being completely pervious. Layers such as road polylines or raster green infrastructures are converted to a user resolution that is finer or equal to the resolution of the AOI raster, and must have the exact same extent of the AOI raster**.** For instance, a user would determine the appropriate road or impervious surface width (10m, 5m, 3m, 2.5m, or 1m etc.), all of which would evenly divide into a 30m processed DEM / AOI ASCII. The AOI raster may be of any resolution, as long as the impervious surface raster evenly divides into it. To illustrate, a 30m AOI that is 50 rows by 100 columns could be evenly divided by a 10m impervious surface ASCII that was 150 rows by 300 columns. When converting polylines, polygons, or rasters, as with all VELMA layers, they must be locked (snap) to the AOI to ensure proper overlap and alignment of higher resolution cells with the AOI.

The perviousness of finer resolution maps will be weighted by their cell size ratio to the AOI DEM when data is aggregated to the AOI map resolution. For example, to derive a 30m cell's fraction perviousness, the number of higher resolution cells that are analyzed as road are summed together and then divided by the total number of cells. That percentage is then subtracted from 1. 1 being completely pervious and 0 entirely impervious. With the scenario drawn in Figure 1, this single 30m cell would evaluate to 1 –

(15 / 36) = 58% or 1 – (8/9) = 11 % perviousness depending on whether a 10m or 5m road grid were used. An example of a processed permeability fraction map can be seen in Figure 2.

Multiple impervious map layers can be aggregated and weighted together to create the permeability map (Figure 3). When calculating the permeability value for a particular cell with multiple overlapping impervious cell values, the layers must be ordered or ranked by precedence of the layer's influence. In figure 3 the "Roads" layer is ranked first and the "Green I." layer ranked last. The "Roads" layer's imperviousness value is 100% and the "Green I." layer's impervious cells values are 50%, and if any cells overlap each other the 100% value from the "Roads" layer will be used to calculate the partial influence of the 10m area that it represents in the permeability map's 30m cell (Figure 3).

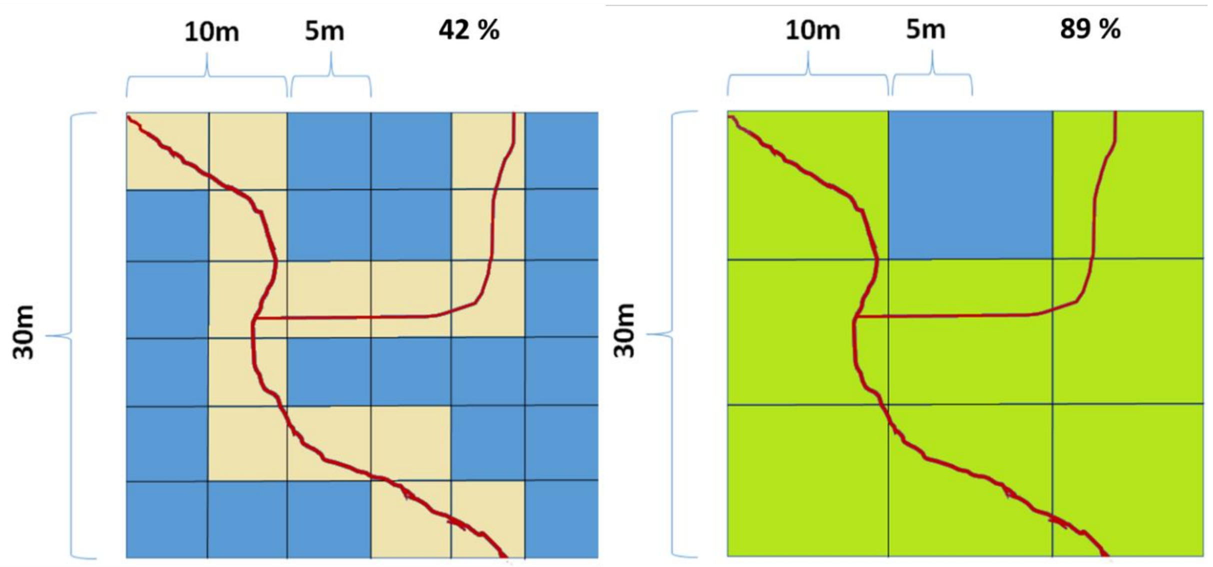


Figure 25. 30m cell overlap of 10m and 5m road grid.

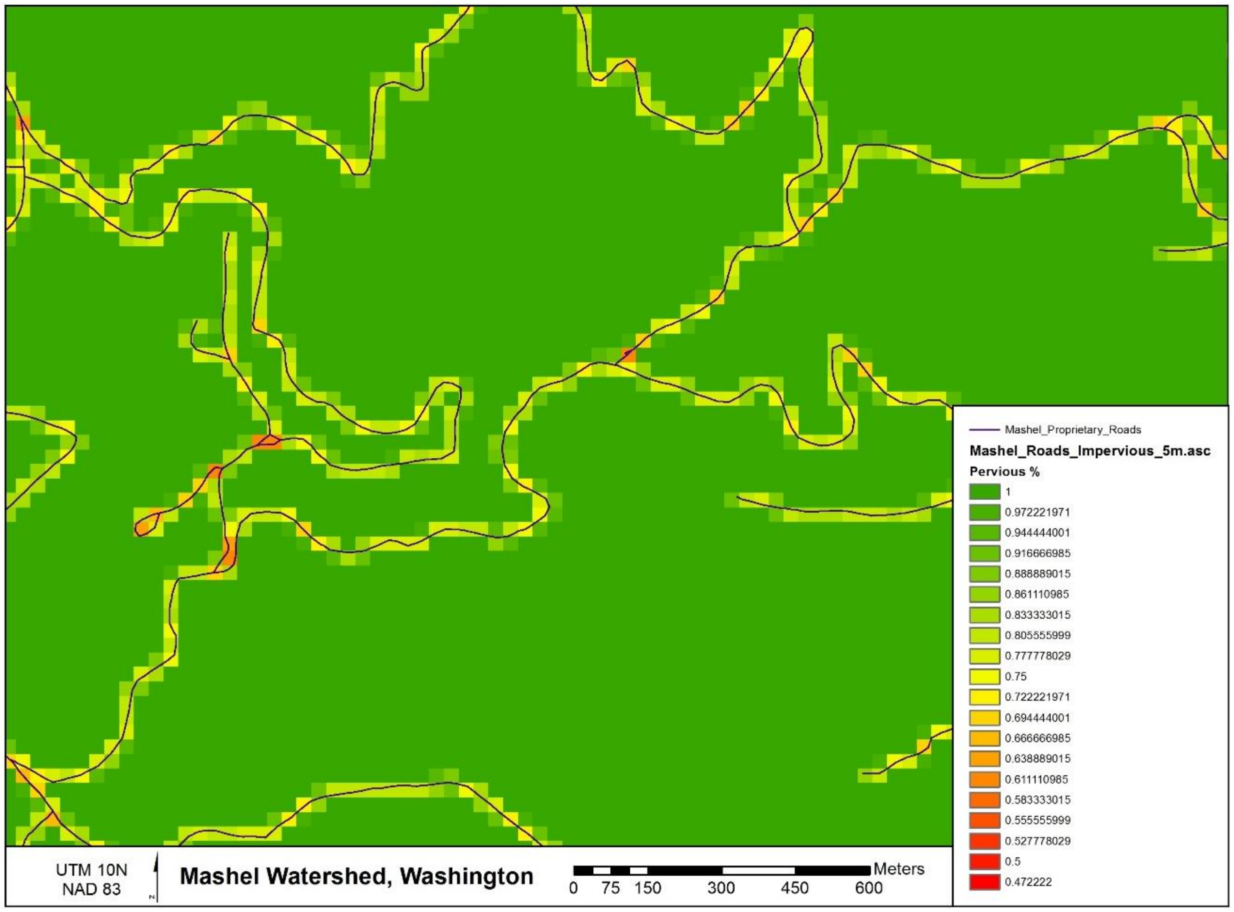


Figure 26. Processed permeability fraction map.

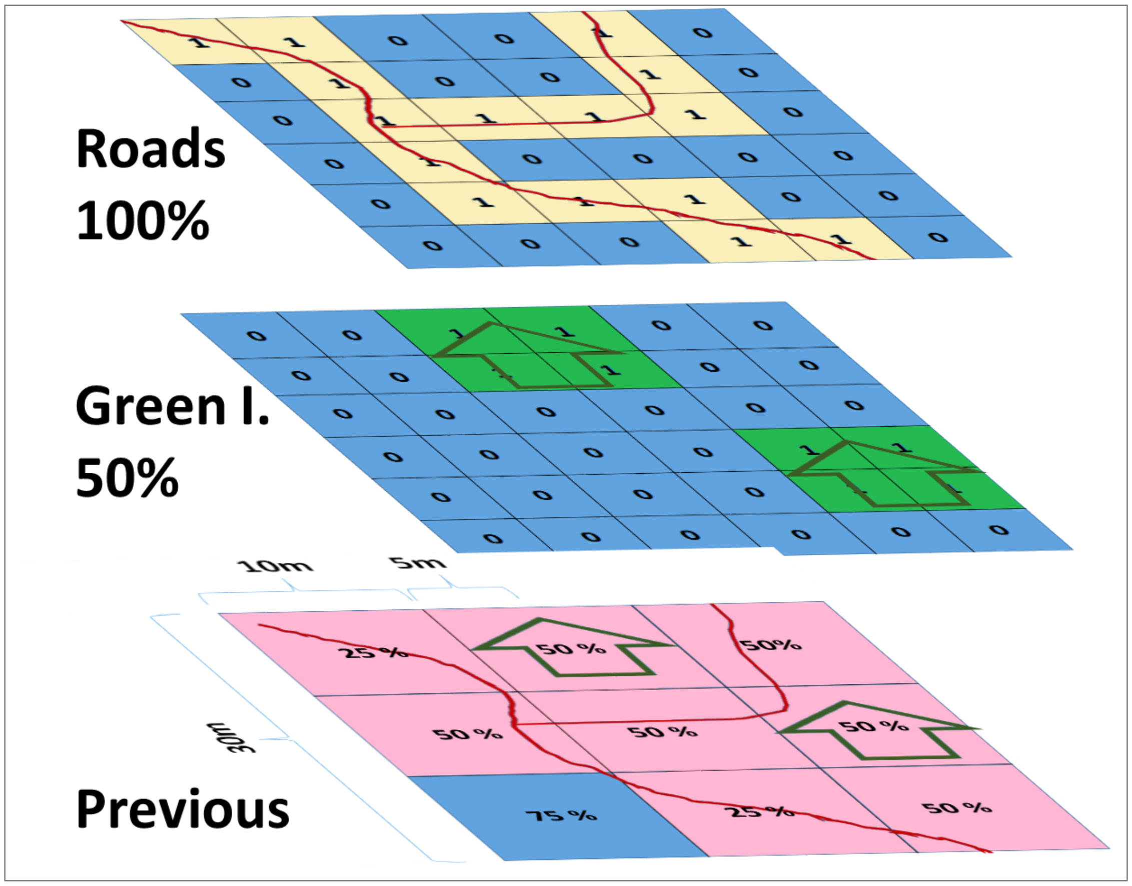


Figure 27 Aggregating multiple impervious surface maps into the permeability layers.

2.3. Software Requirements and Processing

To run and produce the processed permeability fraction map (ASCII) from the "roads" or impervious surface layer (ASCII), you will need the following:

I. Python version 2.x: Current release is 2.7.11
   1. Python comes pre-packaged within ArcGIS, so it is likely you already have it installed on your computer. For example, a default install is currently here: C:\Python27\ArcGIS10.2\python.exe. Check to see if you have Python installed before installing a new version. If Python is not installed, you can obtain a copy here: https://www.python.org/download/releases/2.7/ *Note that Python 2.7 is currently considered safe for use on U.S. EPA network and non-network computers.*

J. ASCII Map requirements:
   1. AOI or processed DEM ASCII raster that matches the VELMA modeling resolution.
   2. Pervious/impervious surface rasters in ASCII format (.asc).
      - Impervious surface cell values set equal to 1 or greater, all other non-pervious and no-data cells values set to -9999, which is the default ERSI ASCII nodata value.

- These map(s) should match the exact extent of the AOI / DEM ASCII regardless if they are higher resolution.

3. For each impervious surface map, the user will have to determine how permeable that surface is. Permeability fraction values must be in the range [0.0, 1.0], where 0.0 is completely impermeable (no water penetrates from surface to soil layer 1) and 1.0 is completely permeable (all water penetrates from surface to soil layer 1).
4. Output filename to be determined by the user.

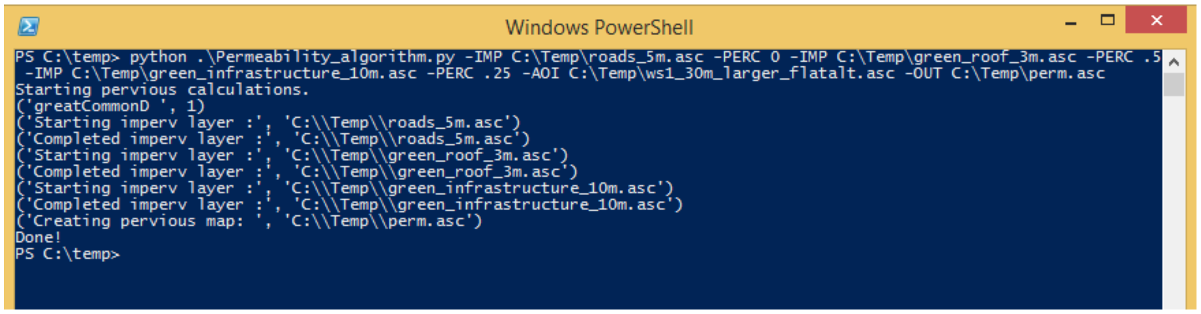
K. The permeability fraction algorithm: "**`Permeability_algorithm.py`**" and can be run on the Command Prompt line with Python. This script requires, at minimum, one impervious surface map with a corresponding percent permeability, one AOI map, and one output file name. The script is designed to allow the user to input multiple impervious surface maps and their corresponding percent permeability.

NOTE – The order of the input impervious maps determines their cell value assignment rank when creating the pervious surface output map. The fist map argument takes precedence over the following maps, and so on for the next map listed. In the example below, if the "`roads_5m.asc`" has cell values that overlap "`green_infrastructure_10m.asc`" values, the "`roads_5m.asc`" pervious percentage will therefor win over the "`green_infrastructure_10m.asc`" values.

Command prompt input example:

```
"python .\Permeability_algorithm.py -IMP C:\Temp\roads_5m.asc -PERC 0 -IMP
C:\Temp\green_roof_3m.asc -PERC .5
 -IMP C:\Temp\green_infrastructure_10m.asc -PERC .25
-AOI C:\Temp\ws1_30m_larger_flatalt.asc -OUT C:\Temp\perm.asc"
```

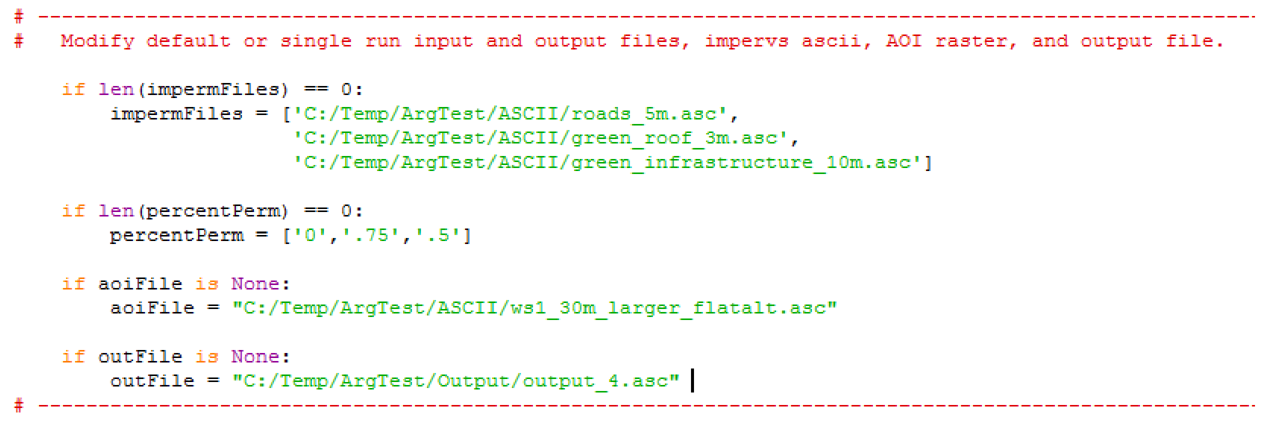
"`python C:\Path\To\Permeability_algorithm.py -help`" for exact arguments (Figure 4).



```
PS C:\temp> python .\Permeability_algorithm.py -IMP C:\Temp\roads_5m.asc -PERC 0 -IMP C:\Temp\green_roof_3m.asc -PERC .5
 -IMP C:\Temp\green_infrastructure_10m.asc -PERC .25 -AOI C:\Temp\ws1_30m_larger_flatalt.asc -OUT C:\Temp\perm.asc
Starting pervious calculations.
('greatCommonD ', 1)
('Starting imperv layer :', 'C:\\Temp\\roads_5m.asc')
('Completed imperv layer :', 'C:\\Temp\\roads_5m.asc')
('Starting imperv layer :', 'C:\\Temp\\green_roof_3m.asc')
('Completed imperv layer :', 'C:\\Temp\\green_roof_3m.asc')
('Starting imperv layer :', 'C:\\Temp\\green_infrastructure_10m.asc')
('Completed imperv layer :', 'C:\\Temp\\green_infrastructure_10m.asc')
('Creating pervious map: ', 'C:\\Temp\\perm.asc')
Done!
PS C:\temp>
```

Figure 28. Command line example of running the pervious surface map algorithm

L. The permeability fraction algorithm also can be run from a Python development environment such as IDLE, by editing the default arguments paths of the inputs and outputs in the python code (Figure 4).



```
# -------------------------------------------------------------------------------------------------
#   Modify default or single run input and output files, impervs ascii, AOI raster, and output file.

    if len(impermFiles) == 0:
        impermFiles = ['C:/Temp/ArgTest/ASCII/roads_5m.asc',
                       'C:/Temp/ArgTest/ASCII/green_roof_3m.asc',
                       'C:/Temp/ArgTest/ASCII/green_infrastructure_10m.asc']

    if len(percentPerm) == 0:
        percentPerm = ['0','.75','.5']

    if aoiFile is None:
        aoiFile = "C:/Temp/ArgTest/ASCII/ws1_30m_larger_flatalt.asc"

    if outFile is None:
        outFile = "C:/Temp/ArgTest/Output/output_4.asc"
# -------------------------------------------------------------------------------------------------
```

Figure 29. Default arguments paths.

Scripts

```
# Pervious_Fraction_Algorithm.py
# Author: Paul Pettus, Kevin Djang
# Date: 10-11-2016
# Description: To create a percentage / fraction permeable surface
#
# Output is a cell size ascii grid that matches the resolution of the AOI
# with percentage values that range from 1 to 0.
#
# Example, either a 5m and 10m imperv map grids are overlaid and divide evenly into
# a 30m grid, and cells that evaluate as imperv are counted up and divided
# by the total overlaid cells under the 30m grid cell.  From which that
# percentage imperv cover is subtract from 1, 1 being completely permeable
# and 0 being completely impermeable.
#
#
# Note: Cells classified as "Roads" or "Impermeable" must have a value >= 1,
# while permeable cells must have a classified value of -9999
#
# Last updated: 6-8-2017

import os, sys, numpy, re, argparse

# ------------------------------------------------------------------------------------------------

# Error message class
class Usage(Exception):
    def __init__(self, msg):
        self.msg = msg

# Import an ascii file with header and array
class myASCIIFile:
    def __init__(self,asciiFile):

        if not os.path.exists(asciiFile):
            raise Usage('Cannot find ASCII "' + asciiFile + '"')

        # Open file and read in header info
        readFile = open(asciiFile)

        self.header = readFile.readline()  #ncols
        self.header += readFile.readline() #nrows
        self.header += readFile.readline() #xllcorner
        self.header += readFile.readline() #yllcorner
        self.header += readFile.readline() #cellsize
        self.header += readFile.readline() #NODATA_value
        readFile.close()

        self.headerList = self.header.rsplit()
        self.ncols = int(self.headerList[1])     #ncols
        self.nrows = int(self.headerList[3])     #nrows
        self.xllcorner = float(self.headerList[5]) #xllcorner
        self.yllcorner = float(self.headerList[7]) #yllcorner
        self.cellsize = float(self.headerList[9])  #cellsize
        self.NODATA = float(self.headerList[11])   #NODATA_value
        self.array = numpy.loadtxt(asciiFile, skiprows=6, dtype= numpy.float64) # ASCII data to numpy array

def main(argv=None):

    if argv is None:
        argv = sys.argv

    try:
        parser = argparse.ArgumentParser(description='Outputs a percent permeability surface map'+
                                         ' from one to many impermeable file layer inputs and' +
                                         ' percentage arguments.')
        parser.add_argument('-IMP', action='append', dest='impermFILE',default=[],
                    help='Fully-qualified path + name of ".asc" impermeable cover file.' +
                    ' Add repeated values to a list.')

        parser.add_argument('-PERC', action='append', dest='impermPERC',default=[],
                    help='Percent permeability of the impermeable surface raster (0-1).' +
                    ' Add repeated values to a list.')

        parser.add_argument('-AOI', action='store', dest='aoiFILE',default='D:/Temp/ArgTest/dem_30m_aoi_flat.asc',
                    help='Fully-qualified path + name of ".asc" AOI / DEM.')

        parser.add_argument('-OUT', action='store', dest='outFILE',default='D:/Temp/ArgTest/Output/output.asc',
                    help='Fully-qualified path + name of ".asc" output file.')

        args = parser.parse_args()

        # args parsing
        impermFileList = args.impermFILE
        percentList = args.impermPERC
        aoiFile = os.path.abspath(args.aoiFILE)
        outFile = os.path.abspath(args.outFILE)

        # do the work
        percentPervious(impermFileList, percentList, aoiFile, outFile)

    except Usage as e:
        print(e.msg)
        return 2

    except Exception as e:
        # STUB exception handler
        # Warning: poor programming style.
        # Catches almost any exception (but not KeyboardInterrupt -- which is a Good Thing)
        raise e

def percentPervious(impermFiles, percentPerm, aoiFile = None, outFile = None):
# -----------------------------------------------------------------------------------------------------------
#   Modify default or single run input and output files, impervs ascii, AOI raster, and output file.

    if len(impermFiles) == 0:
        impermFiles = ['D:/Temp/ArgTest/ASCII/roads_5m.asc',
                  'D:/Temp/ArgTest/ASCII/green_roof_3m.asc',
            'D:/Temp/ArgTest/ASCII/green_infrastructure_10m.asc']

    if len(percentPerm) == 0:
        percentPerm = ['0','.5','.25']

    if aoiFile is None:
        aoiFile = "D:/Temp/ArgTest/ASCII/ws1_30m_larger_flatalt.asc"

    if outFile is None:
        outFile = "D:/Temp/ArgTest/Output/output_9.asc"
# ------------------------------------------------------------------------------------------------------------

    # Check that AOI file exists
    if not os.path.exists(aoiFile):
        raise Usage('Cannot find AOI file "' + aoiFile + '"')

    # Check that all impermeable rasters exist
    for item in impermFiles:
        if not os.path.exists(item):
            raise Usage('Cannot find impermeable file "' + item + '"')

    # Check that the number of permeable layer match the percentage input.
    if len(impermFiles) != len(percentPerm):
        raise Usage('Number of permeable layers do not match percentage inputs!')

    print ("Starting pervious calculations.")

    # read in AOI header and retrive cell size
    aoiAscii = myASCIIFile(aoiFile)

    # Empty list to hold all of the impervious layers cell sizes
    impervCellSizes = []

    for itemAscii in impermFiles:
        # Read in each impervious file
        impermAscii = myASCIIFile(itemAscii)

        # Verify that the columns are evenly divisible into the AOI
        if int(impermAscii.ncols) % int(aoiAscii.ncols) != 0:
            raise Usage("Columns don't divide evenly. " + itemAscii)

        # Verify that the rows are evenly divisible into the AOI
        if int(impermAscii.nrows) % int(aoiAscii.nrows) != 0:
            raise Usage("Rows don't divide evenly. " + itemAscii)

        # Verify that the cell sizes are evenly divisible into the AOI
        if float(aoiAscii.cellsize) % float(impermAscii.cellsize) != 0:
            raise Usage("Cells sizes do not divide evenly. " + itemAscii)

        # read in imperv file header and retrive cell size
        impervCellSizes.append(int(impermAscii.cellsize))

    # Find the finest resolution cell size
    minCellSize = min(impervCellSizes)


    # Get the greatest common divisor of cell sizes
    def gcd(*numbers):
        """Return the greatest common divisor of the given integers"""
        from fractions import gcd
        return reduce(gcd, numbers)

    # Get the greatest common divisor of cell sizes
    greatCommonD = apply(gcd,impervCellSizes)

    print("Greatest Common Divisor cell size: ", greatCommonD)
```

```
# Calculate number or rows and cols for the finest resolution array / raster
maxIpermRow = int(int(aoiAscii.nrows) * (float(aoiAscii.cellsize) / greatCommonD))
maxIpermCol = int(int(aoiAscii.ncols) * (float(aoiAscii.cellsize) / greatCommonD))

# Create the highest resolution array
# This array will hold all the imperious values for all layers
impermOverlapArray = numpy.zeros((maxIpermRow, maxIpermCol))

# Load input files as arrays
aoiArray = aoiAscii.array
impervArray = impermAscii.array

# create empty output array
row, col = aoiArray.shape
outArray = numpy.zeros((row,col))

percentPosition = 0

# Loop through all the impervious files
for itemAscii in impermFiles:

    print("Starting imperv layer :", itemAscii)

    # flip percentPerm to percentage impermeable so that empty or zero values in
    # impermOverlapArray Array values can be evaluated as processed or empty
    # impermeable values will be converted to permeable values after all
    # layers have been processed
    percentImperm = 1 - float(percentPerm[percentPosition])

    # Read in impervious surface file
    impermAscii = myASCIIFile(itemAscii)

    # Calculate the ratio of imperv cells to finest resolution imperv raster cell size
    cellRatio = int(float(impermAscii.cellsize)/float(greatCommonD))

    # Calculate number of imperv cells per current imperv raster cell
    impervCellsToSmall = cellRatio * cellRatio

    # For each imperm layer cell assign a percent impervios value to the impermOverlapArray
    if cellRatio != 1:
        for i in xrange(int(impermAscii.nrows)):
            for j in xrange(int(impermAscii.ncols)):

                cellValue = impermAscii.array[i,j]
                # Check that the cell has a value that is not NODATA
                if cellValue != -9999:
                # Loop through the over lapped impermOverlapArray cells, e.g. 30m / 5m = 6x6 overlap
                    for r in xrange(cellRatio):
                        for c in xrange(cellRatio):
                            impervRow = (i*cellRatio) + (r)
                            impervCol = (j*cellRatio) + (c)

                            # imperm cell has a value of > 0, completely permeable cells have a value of 0
                            # if a impermeable cell values over lap, the first layer argument will take
                            # priority and assign its value first. Poor Programing :(
                            if impermOverlapArray[impervRow,impervCol] == 0:
                                # Assign value to master array
                                impermOverlapArray[impervRow,impervCol] = percentImperm

    # else: Ratio is equal to 1
    else:
        for i in xrange(int(impermAscii.nrows)):
            for j in xrange(int(impermAscii.ncols)):
                cellValue = impermAscii.array[i,j]
                # imperm cell has a value of > 0, completely permeable cells have a value of 0
                # if a impermeable cell values over lap, the first layer argument will take
                # priority and assign its value first. Poor Programing :(
                # Check that the cell has a value that is not NODATA
                if cellValue != -9999:
                    # Assign value to master array
                    impermOverlapArray[i,j] = percentImperv

        # increment the position of the percent list
        percentPosition = percentPosition + 1
        print("Completed imperv layer :", itemAscii)

    # Get the ratio of the impermOverlapArray cells to AOI raster cells
    aoiOverlapRatio = int(float(aoiAscii.cellsize)/float(greatCommonD))

    print("Creating pervious map :", outFile)

    # For each aoi cell calculate the percentage permeable
    for i in xrange(row):
        for j in xrange(col):
            # impermSum holds the total of all imperm cell values set to zero each loop
            impermSum = 0

            # Loop through the over lapped impervious cells, e.g. 30m / 5m = 6x6 overlap
            for r in xrange(aoiOverlapRatio):
                for c in xrange(aoiOverlapRatio):
                    impervRow = (i*aoiOverlapRatio) + (r)
                    impervCol = (j*aoiOverlapRatio) + (c)
                    # retrieve cell value
                    cellValue = impermOverlapArray[impervRow,impervCol]
                    # add cell value to impermSum
                    impermSum = impermSum + cellValue
            # Calculate pervious percentage from impervious values, converted back to pervious
            permeableValue = 1 - (impermSum / (aoiOverlapRatio * aoiOverlapRatio))  # Calculate pervious percentage
            # Assign pervious value to output array
            outArray[i,j] = permeableValue

    # Open output file
    f = open(outFile, "w")
    # Write AOI header to output file
    f.write(aoiAscii.header)
    # Save values array to output file
    numpy.savetxt(f, outArray, fmt="%f")
    f.close()
    print("Done!")


if __name__ == "__main__":
    sys.exit(main())
```

E.2 | VELMA Surface Chemistry Pools

Overview *(Tutorial E.2 – VELMA Surface Chemistry Pools)*

This document describes how to implement surface chemistry pools in VELMA.

“Surface chemistry pools” is a new VELMA feature (version 2.1) that can be used to simulate additions of nutrients and contaminants to road surfaces or other non-soil layer positioned on top of a VELMA soil column (layers 1-4).

In previous model versions, atmospheric deposition of nitrogen, fertilizer additions and so forth had to added to soil layer 1 in VELMA. Now, chemical additions can be more realistically simulated as additions to surfaces that may differ with respect to their permeabilities.

The ability to simulate surface chemistry pools is important because deposited chemicals can potentially reside undissolved on a road or other impermeable surface for some time without being accessible to plant roots and other biogeochemical processes. Or, when rainfall does occur in sufficient amount, the dissolved chemical may run off laterally until reaching a downslope soil column with a more permeable surface.

Note that a new water permeability layer has also been added to VELMA. This feature lies between the surface chemistry pool and soil layer 1. For details see “*Tutorial E.1 – Mapping Surface Layer Permeabilities in VELMA*”.

The Surface Pools “Overlay” Their Corresponding Layered Pools

The surface NH4 pool can be thought of as “overlaying” the four layers of the NH4 chemistry pool, with the water permeability layer lying in between them. Similarly, surface NO3 overlays layered NO3, etc. Surface pool values are in grams of Nitrogen per square meter (g N/m2), the same as the layered pools, or g C/m2, in the case of DOC. Organic Contaminant parameterizations added to VELMA simulation configurations likewise will have a surface pool overlaying a layered pool.

The Nin (Nitrogen Deposition) Surface Pool is Special

Other chemistry surface pools are completely new functionality, but the Nin pool is a refactoring of the already-existing Nin code. Prior Nin behavior continues to be available, and is shunted through the new surface Nin pool.

You can select the type of Nin behavior via the `useExperimentalNin` configuration parameter.
In prior simulation configurations, this parameter was a boolean with the following behavior:

OLD-Style Value	Simulation Behavior
`false`	Use original/legacy Nin code: Constant daily deposition rate. Vertical transport to NH4 pool layer 1. NO lateral transport.

`true`	Use Banked Michaelis-Menten Nin code: Constant daily deposition rate. Vertical transport to NH4 pool layer 1, based on Banked, water-sensitive M-M equation. NO lateral transport.

In new simulation configurations, the useExperimentalNin parameter is an enum with three possible settings:

NEW-Style Value	Simulation Behavior
`NIN_LEGACY`	Use original/legacy Nin code: Constant daily deposition rate. Vertical transport to NH4 pool layer 1. NO lateral transport.
`NIN_BANKED_MICHAELIS_MENTEN`	Use Banked Michaelis-Menten Nin code: Constant daily deposition rate. Vertical transport to NH4 pool layer 1, based on Banked, water-sensitive M-M equation. NO lateral transport.
`NIN_SURFACE_POOL`	Use new, Surface Nin Pool code: Constant daily deposition rate into SURFACE_N_IN pool. Vertical transport to NH4 pool layer 1, based on transportable fraction of SURFACE_N_IN pool and water balance’s current vertical flow value. Lateral transport to adjacent SURFACE_N_IN pool cells based on water balance’s current lateral flow value.

When older, legacy simulation configuration .xml files are loaded into JVelma, the following mapping from old-style to new-style `useExperimentalNin` occurs automatically:

OLD-style Value	NEW-style Value
`false`	`NIN_LEGACY`
`true`	`NIN_BANKED_MICHAELIS_MENTEN`

Notice that there is no way for a legacy simulation configuration to automatically map to the new NIN_SURFACE_POOL mode’s code. This is deliberate. The NIN_SURFACE_POOL mode is currently considered *very* prototype and should not be used without explicit intent.

The Cells of Surface Pools Can Be Empty (and Start Out That Way).

Unlike the surface Nin pool, the other new, surface pools are *not* automatically utilized by any existing VELMA simulation code, submodel, or process.

All Surface Pools Are Affected by Water Movement, Subject to Permeability

All surface pools (Nin pool included, when `useExperimentalNin = SURFACE_N_IN`) are affected by the simulation’s daily, water-based vertical and lateral chemistry transport steps, but if a surface pool cell’s amount is zero, it is ignored for lateral and vertical outflow (it will still accept inflow). This behavior means that simulation configurations created before the introduction of the surface pools should run the same as they did before the surface pools existed. Vertical down-flow from surface pools takes the permeability value between the surface pool cell and its corresponding, underneath, layered pool into account.

The Cells in Surface Pools Stay Empty Unless Explicitly Filled.

Apart from the surface Nin pool, a cell in a surface pool only gains a quantity when you explicitly provide it with one. Once it has a quantity, that quantity may automatically move to surrounding surface cells, or down into layer 1 of the corresponding chemistry’s sub-surface layered pool (via water transport). However, cells only gain amounts when those amounts are explicitly provided.

Put Quantities into Surface Pool Cells via Disturbances.

You can introduce quantities of chemical into their respective surface pool by configuring your simulation to include one or more appropriate Disturbances.

Quick summary/overview of the disturbances that affect Surface Pools

`SetSpatialDataByMapDisturbanceModel`

This is the “standard” Set SpatialData by Map disturbance that assigns all the cells in a given pool via values from a Grid ASCII .asc map file. It is typically used to initialize the various biomass, detritus and chemistry pools. Since surface pools are “like any other pools”, you can set initial values for them with this disturbance the same way you would for any other pool.

`SurfaceDepositionDisturbanceModel`

This is a new disturbance designed to work specifically with surface pools. It allows assignment of specific amounts to specific surface pool cells at specific times during the simulation run. It achieves this pin-point specificity by requiring a more-advanced version of the Historical… disturbance’s “jagged array” input file.

`FertilizeLsrDisturbanceModel`

The “legacy” (i.e. pre-surface pool introduction) Fertilize disturbance model has been updated to include a `depositionTargetOption` parameter. Its default value is “`SOIL`”, which instructs the disturbance to assign specified amounts of NH4, NO3 and/or Humus to those pool’s layer 1 cells. (I.e. the legacy behavior). However, you can set the `depositionTargetOption` parameter to “`SURFACE`”, which will instead assign specified amounts of NH4 and/or NO3 to the respective surface NH4 and/or NO3 pools instead. Regardless of `depositionTargetOption’s` parameter value, Humus amounts are always deposited to Humus Layer 1, because there is, as yet, no surface pool for Humus.

`HistoricalFertilizeLsrDisturbanceModel`

The "legacy" Historical Fertilize disturbance model has the same `depositionTargetOption` as the non-historical `FertilizeLsrDisturbanceModel`. Note that the new option only alters which pool "layer" (layer 1 or surface) receives the deposition. Whether a give cell location gets fertilized at all is unchanged behavior for either type of fertilize disturbance.

NOTE: Fertilize disturbances only act upon NH4 and NO3 surface pools. Other chemistry (e.g. an Organic Contaminant of some parameterization) pools must be initialized/augmented via one of the other methods listed above.

Spatial Data Writers "Know" About Surface Pools

You can output the state of a surface pool by including a spatial data writer parameterization for that pool as part of your simulation configuration. The keywords for the new, core chemistry spatial data pools are: `SURFACE_NH4, SURFACE_NO3, SURFACE_DON, SURFACE_DOC,` and `SURFACE_N_IN`. Contaminant surface pool names have the text "CONTAMINANT_SURFACE_" plus the Contaminant's uniqueName parameter value appended.

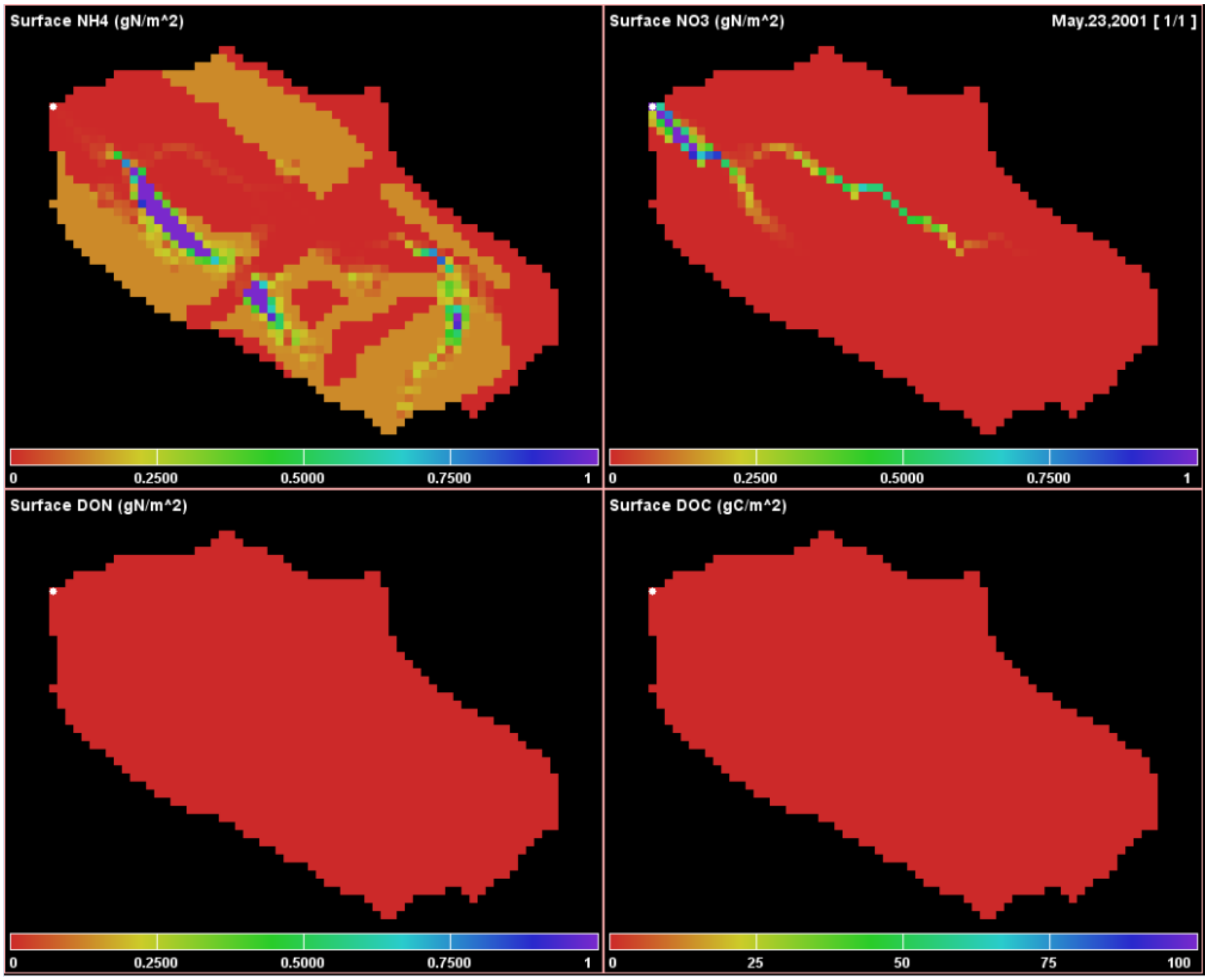
JVelma Provides New Runtime Spatial Displays for the Core Chemistry Surface Pools

There are two new spatial displays available in JVelma's "Display" selector drop-down list: "Surface Chemistry Spatial" and "Surface Nin Spatial".

(continued)

Here are example screen-captures for each these displays

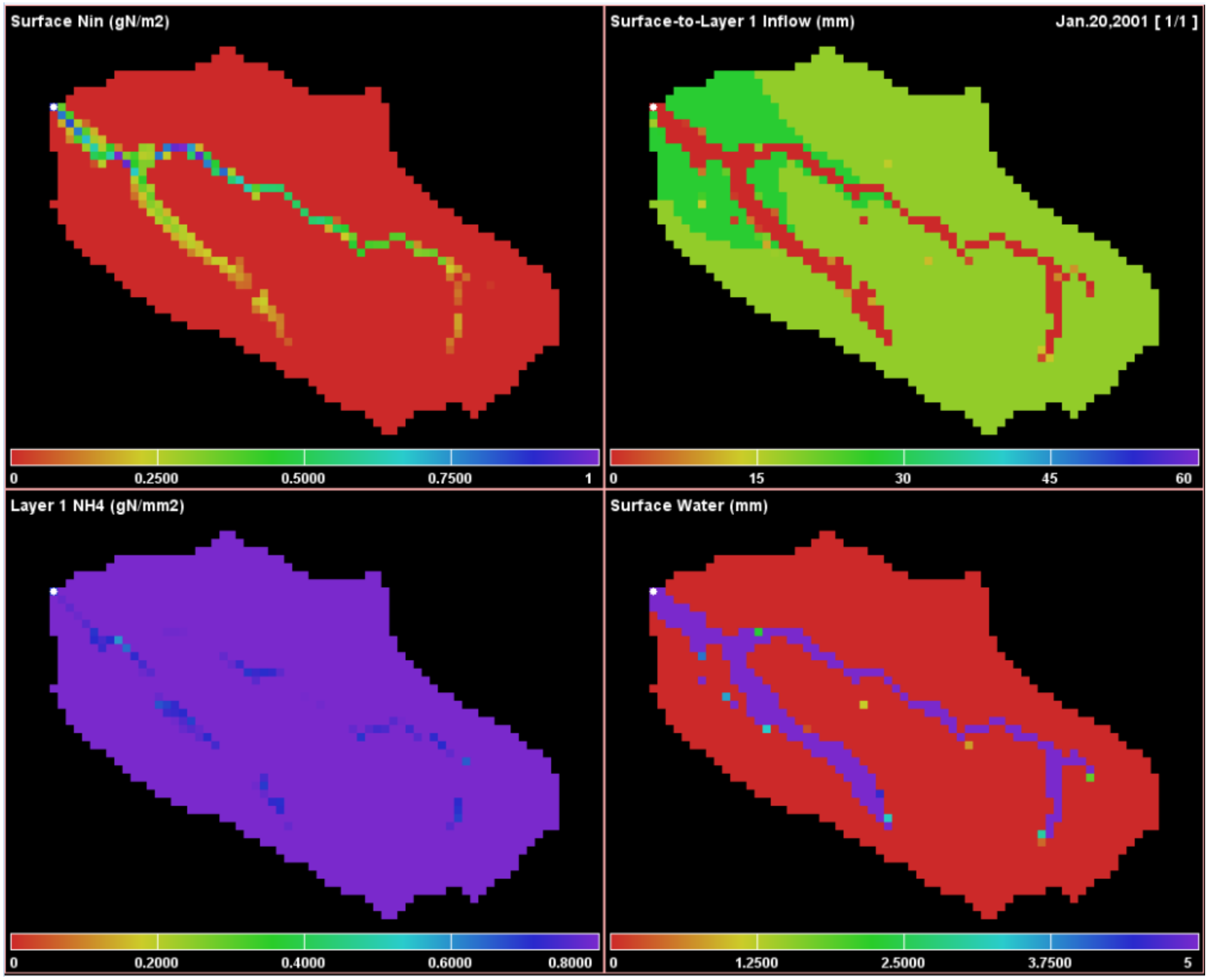
Surface Chemistry Spatial display for a simulation run that includes a Fertilization disturbance depositing NH4 and NO3 to surface layers.



Applicable Display Range Parameters for the Surface Chemistry Spatial display are:

Chart Position	**Display Range Parameter(s)**
UPPER LEFT	`maxSurfaceNh4SaptialDisplay`
UPPER RIGHT	`maxSurfaceNo3SpatialDisplay`
LOWER LEFT	`maxSurfaceDonSpatialDisplay`
LOWER RIGHT	`maxSurfaceDocSpatialDisplay`

Surface Nin Spatial display for simulation run with useExperimentalNin = NIN_SURFACE_POOL



Applicable Display Range Parameters for the Surfacen Nin Spatial display are:

Chart Position	Display Range Parameter(s)
UPPER LEFT	`maxNinSpatialDisplay`
UPPER RIGHT	`maxRainSpatialDisplay`
LOWER LEFT	`maxNh4SpatialDisplay`
LOWER RIGHT	`maxNinSpatialDisplay` (or) `maxStandingWaterSpatialDisplay`

Note that the LOWER RIGHT Surface Nin Spatial Display shows either Accumulated N-in (a “hidden” pool associated with Banked Michaelis-Menton mode) or Standing Water (i.e. Surface Water). The former is displayed when `useExperimentalNin` = NIN_LEGACY or NIN_BANKED_MICHAELIS_MENTEN, the latter when `useExperimentalNin` = NIN_SURFACE_POOL (as was the case for the above screen capture).

E.3 | Deposition of Surface Chemicals

Overview *(Tutorial E.3 – Surface Chemical Deposition)*

This document describes how to use VELMA’s disturbance routine to control the timing, amount and location of a chemical to deposit to the surface pool (the surface pool is located above soil layers 1-4).

This tutorial builds upon two other VELMA surface layer tutorials. We recommend consulting these before applying this tutorial:

- *Tutorial E.1 – Mapping Surface Layer Permeabilities*
- *Tutorial E.2 – VELMA Surface Chemistry Pools*

A Surface Deposition Disturbance increases the amount of chemical in the surface pool of a specific chemical type (e.g. NH4, or a configuration-specific contaminant) at specific times and locations during a VELMA simulation run.

A single Surface Deposition Disturbance targets a single chemical type. If you need to perform surface deposition for more than one chemical (e.g. for NH4 and NO3) you much configure multiple, separate, Surface Deposition Disturbances.

The Surface Deposition Disturbance uses a diver schedule, encoded in a .csv file to specify WHEN (i.e., which simulation steps) and WHERE (i.e., which cells) additional amounts are deposited.
(However, you specify the loops it occurs in via the activeLoops parameter.)

Overview: Configuration Steps for a Surface Deposition Disturbance

Adding one or more Surface Deposition Disturbances to a VELMA simulation configuration via JVelma involves the following two steps:

3. Create the deposition data .csv driver file for the surface deposition.

4. Add a new Surface Deposition disturbance parameterization group to the simulation configuration.

Creating a Surface Deposition Disturbance Driver Data File

Almost all the configuration information a Surface Deposition Disturbance requires is contained within a driver data file. The file consists of one or more rows of comma-separated values. Each row must contain at least four values (“fields”) which denote (in order):

```
iCell, yearRange, jdayRange, depositAmount.
```

Where:

`iCell` = the integer linear index of the cell location within the simulation watershed that will receive deposition amounts.

`yearRange` = the integer year (e.g. 1981) or range of years (e.g. 1981-1991) in which `iCell` will receive the specified `depositAmount`.

`jdayRange` = the integer Julian day of the year (e.g. 30) or range of jdays (e.g. 30-72) in which `iCell` will receive the specified `depositAmount`.

`depositAmount` = the decimal amount of chemical (in g/m2) to deposit in `iCell` on the specified `yearRange`, `jdayRange` (e.g. 0.5).

The `yearRange`, `jdayRange`, and `depositAmount` fields triplet may be repeated within a row, but it is an error to specify a given year, jday pair more than once in a single row (i.e. for a single `iCell`).

The file can contain a header row of comma-separated column header texts as its first row, but this is not necessary, and the header row is ignored if it exists. When a header row is present, it must *not* have a digit as the first character of the leftmost column header's text.

Here is are the rows for a small example deposition driver .csv file:

```
iCell, yearRange, jDayRange, depositAmount
5216,  1980,      30-60,     1.5
5330,  1985-1990, 1,         0.25,          1992, 90-107, 0.3
```

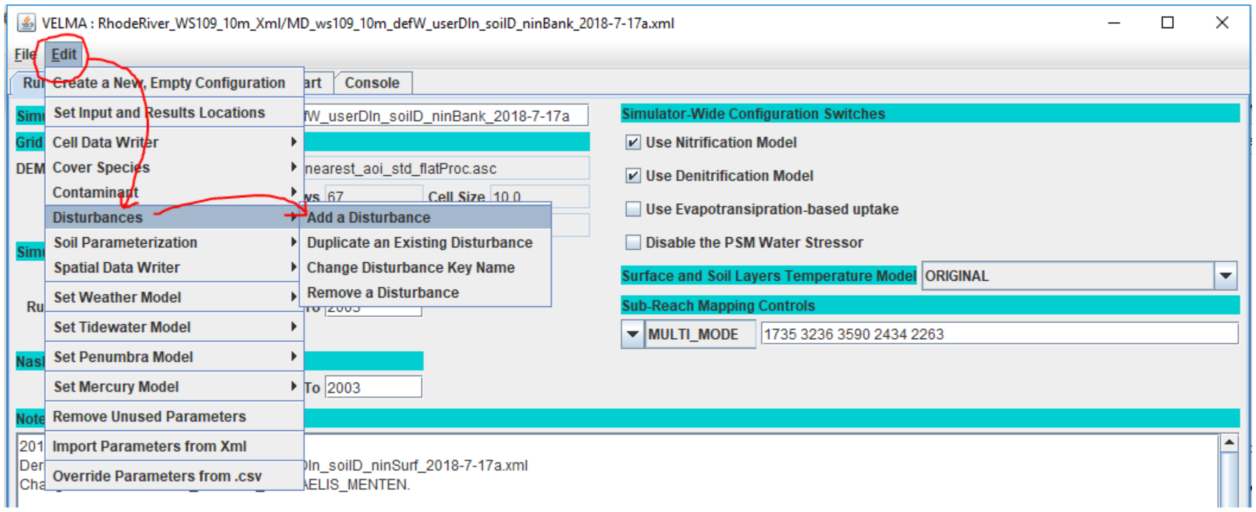
The above example sets deposition times and amounts for two locations (cells 5216 and 5330) of the surface pool specified to the deposition disturbance. Notice that the deposition driver schedule in row 3 (for cell 5330) contains two `(yearRange, jdayRange, depositAmount)` "triplets" – and either row could contain as many more as necessary, so long as the year and Julian day values of any two triplets on the same row do not overlap/collide.
Cell 5216 will receive 1.5 g/m2 to its surface pool during Julian days 30 through 60 of year 1980.
Cell 5330 will receive 0.25 g/m2 to its surface pool on the first day of years 1985 through 1990, and 0.3 g/m2 during Julian days 90 through 107 of year 1992.

Note: whitespace is ignored by the disturbance's file reader, the above spacing is for example readability.

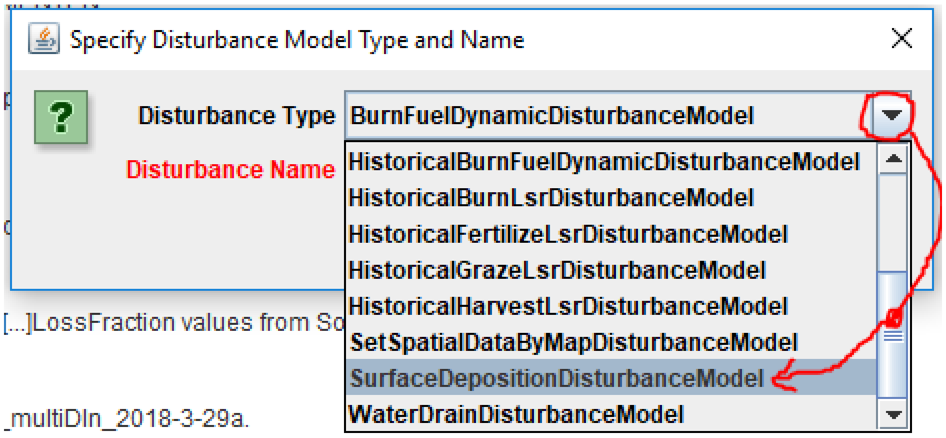
Implicit in the creation of a deposition driver file is the assumption that the iCell values prefixing each row of the file are valid linear indices within the delineated watershed of the VELMA simulation configuration that the Surface Deposition Disturbance and driver data will be added to.
It may be helpful to open VELMA the simulation configuration's flat-processed DEM .asc file in JPDEM, delineate it at the configuration-specified outlet

Adding a Surface Deposition Disturbance Parameterization to a VELMA Configuration

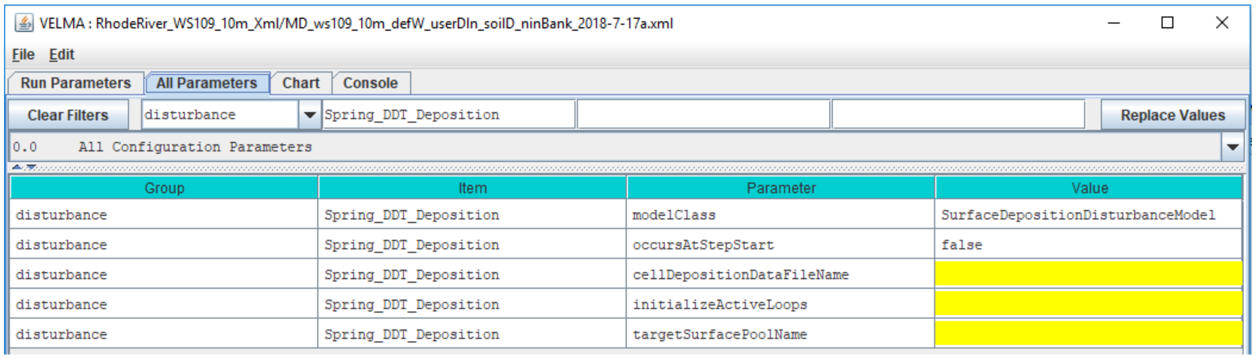
Start with an appropriate VELMA simulation configuration .xml file and deposition driver data .csv file (i.e., the driver data .csv file's cell indices are appropriate for the configuration .xml file's DEM and delineation). Load the VELMA simulation configuration .xml file into JVelma, then click the Edit -> Disturbances -> "Add a Disturbance" menu item, as shown below:


VELMA : RhodeRiver_WS109_10m_Xml/MD_ws109_10m_defW_userDIn_soilD_ninBank_2018-7-17a.xml
File
Edit
Create a New, Empty Configuration
Set Input and Results Locations
Cell Data Writer
Cover Species
Contaminant
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Set Penumbra Model
Set Mercury Model
Remove Unused Parameters
Import Parameters from Xml
Override Parameters from .csv
Add a Disturbance
Duplicate an Existing Disturbance
Change Disturbance Key Name
Remove a Disturbance
Console
Simulator-Wide Configuration Switches
Use Nitrification Model
Use Denitrification Model
Use Evapotranspiration-based uptake
Disable the PSM Water Stressor
Surface and Soil Layers Temperature Model
ORIGINAL
Sub-Reach Mapping Controls
MULTI_MODE
1735 3236 3590 2434 2263

In the "Specify Disturbance model Type and Name" pop-up dialog that opens next, click the drop-down selector, scroll down, and click-select the SurfaceDepositionDisturbanceModel type:


Specify Disturbance Model Type and Name
Disturbance Type
BurnFuelDynamicDisturbanceModel
Disturbance Name
HistoricalBurnFuelDynamicDisturbanceModel
HistoricalBurnLsrDisturbanceModel
HistoricalFertilizeLsrDisturbanceModel
HistoricalGrazeLsrDisturbanceModel
HistoricalHarvestLsrDisturbanceModel
SetSpatialDataByMapDisturbanceModel
SurfaceDepositionDisturbanceModel
WaterDrainDisturbanceModel

Next, type in a unique name for the disturbance in the "Disturbance Name" field of the dialog. Finally click the OK button, which adds a new parameterization group to the simulation configuration, and changes JVelma's display to the "All Parameters" tab, with the Item-level filter set to only display the parameters for the newly-added contaminant.


VELMA : RhodeRiver_WS109_10m_Xml/MD_ws109_10m_defW_userDIn_soilD_ninBank_2018-7-17a.xml
File
Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
disturbance
Spring_DDT_Deposition
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
disturbance	Spring_DDT_Deposition	modelClass	SurfaceDepositionDisturbanceModel
disturbance	Spring_DDT_Deposition	occursAtStepStart	false
disturbance	Spring_DDT_Deposition	cellDepositionDataFileName	
disturbance	Spring_DDT_Deposition	initializeActiveLoops	
disturbance	Spring_DDT_Deposition	targetSurfacePoolName	

Parameters That Must Be Left Alone

The `modelClass` parameter is set by JVelma when you add the Surface Deposition Disturbance to the simulation configuration. Do Not Change this parameter’s value. Ever.

Parameters You Can / Must Provide Values For

The `occursAtStepStart` parameter is default-set to “false”, and indicates that the disturbance is run at the end of each simulation step. This means that the water balance, chemistry and chemical transport for that step has already taken place. Deposition amounts will be part of the next step’s calculations.
The only other value for `occursAtStepStart` is “true” (without the quote marks) and indicates that the disturbance is run at the start of each simulation step, and be part of that step’s chemistry and chemical transport calculations. Neither value is “more correct” but setting `occursAtStepStart` to “true” may make analyzing results a bit simpler. (The default value of “false” is a legacy of earlier VELMA default behavior).

Set the `initializeActiveLoops` to the integer value or integer range of values when the disturbance should be active. For example, if your simulation was configured to run from year 2000 through 2015, and 5 loops, but you only want the Surface Deposition Disturbance to occur for the last 2 loops, set the value to “4, 5” or “4-5” (without the quote marks).

Set the `cellDepositionDataFileName` to the name of the deposition driver data .csv file you prepared for this Surface Deposition Disturbance. You can provide a fully-qualified path and name, a partial-path and name, or just a file name. When the name isn’t fully-qualified, JVelma assumes the file’s location is relative to the input location specified for the simulation run by the `inputDataLocationRootName` and `inputDataLocationFileName` (startups group) parameters.

The `targetSurfacePoolName` parameter specifies the surface pool that the disturbance will add chemical amounts to. You can deposit amounts to any of the VELMA core chemistry pools, or any contaminants parameterized for the simulation configuration.
The values for the core chemistry pools are SURFACE_NH4, SURFACE_NO3, SURFACE_DON and SURFACE_DOC.
To specify a contaminant surface pool, set the value of this parameter to `CONTAMINANT_SURFACE_ +` the value of the contaminant’s `uniqueName` parameter. Example, for a contaminant with `uniqueName` = DDT, the `targetSurfacePoolName` would be CONTAMINANT_SURFACE_DDT.

Capturing Output for a Surface Deposition Disturbance

No additional results are automatically recorded when a Surface Deposition Disturbance occurs.
To record the results, set up a Spatial Data Writer parameterization for the same SURFACE_ pool that the disturbance deposits chemical amounts to, triggering it to write maps on the days you are interested in results for. You can also set Cell Data Writers for cell locations of specific interest.

E.4 | Implementing Green Roof Applications in VELMA

Overview *(Tutorial E.4 – Implementing Green Roof Applications)*

Various green infrastructure (GI) can be included in VELMA watershed setups. This document provides guidance on how to implement green roofs in VELMA. As a user, the primary considerations are to change land cover and soil specifications to match the physical elements used for green roofs.

Intensive vs. Extensive Green Roofs

There are two general categories of green roofs: intensive and extensive. Intensive green roofs have thicker soil columns (6-36”) and allow much larger plant species to be maintained, while extensive green roofs have shallower soil columns (2-6”) and much smaller biomass plant species. Examples are shown below:

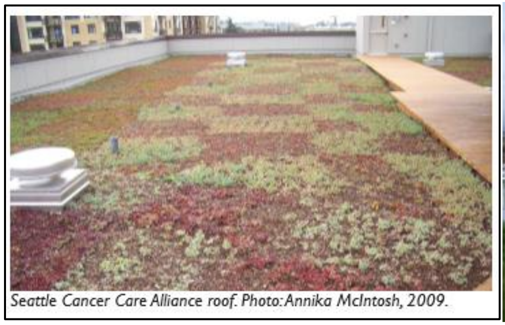
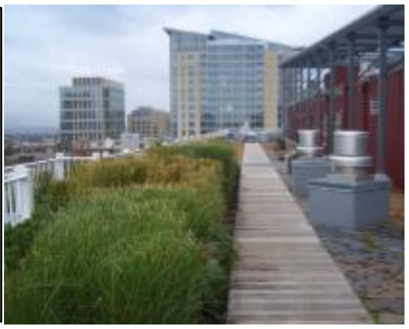

Seattle Cancer Care Alliance roof. Photo:Annika McIntosh, 2009.


Figure 1. Examples of extensive (left) and intensive (right) green roof implementations. Images obtained from “Green roofs in Seattle: A survey of vegetated roofs and rooftop gardens”, which was prepared for the City of Seattle and the University of Washington Green Futures Lab by Annika McIntosh in August 2010 and can be located at http://www.seattle.gov/Documents/Departments/OSE/Green-Roofs-In-Seattle.pdf

VELMA Mechanism for Simulating Green Roofs

A VELMA user will need to make two sets of changes to correctly parameterize VELMA to simulate the implementation of green roofs within a watershed. These include cover type parameterizations and soil type parameterizations. The user will first need to create new cover and soil types that correspond to the locations of green roofs within their watersheds. This is accomplished in the same manner as adding any new cover or soil type, as discussed in other portions of the VELMA manual. Note that if extensive AND intensive green roofs are included in the same model simulation, then they need to have corresponding cover and soil types for each type.

Below is a sketch depicting the general use of green roofs in VELMA.

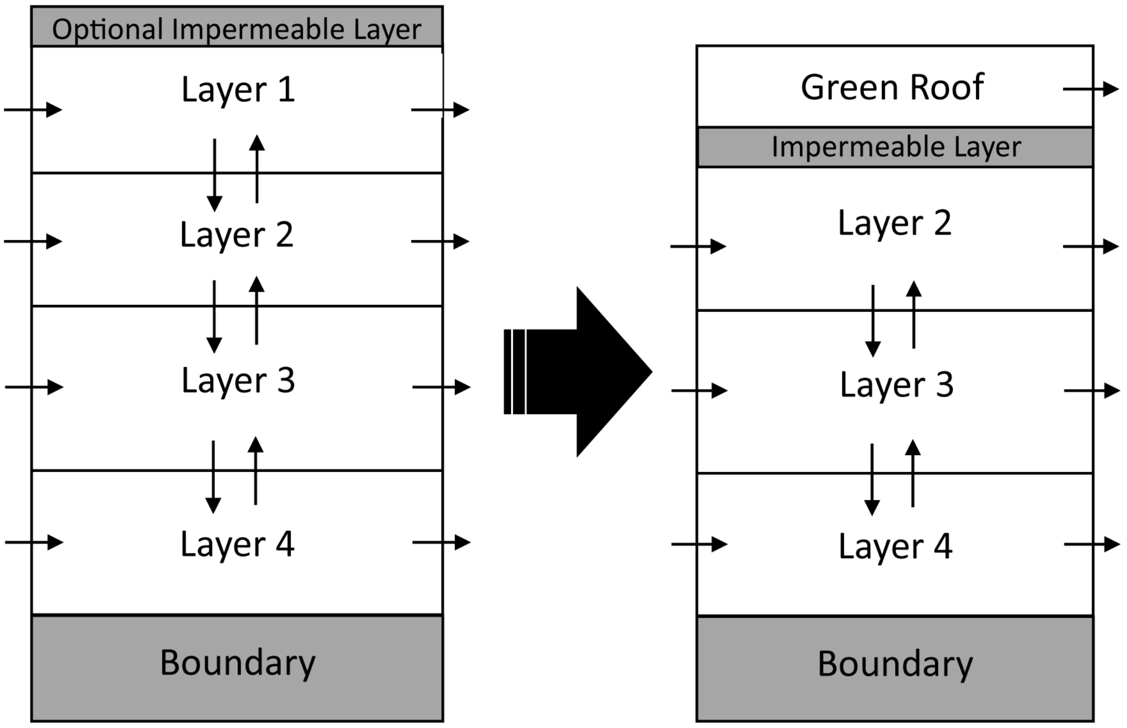


Figure 2. Graphical depiction of simulating green roofs in VELMA. The arrows represent the flow of water and nutrients between adjacent cells.

Note that the first layer is used to describe the media used in the green roof. In the soil parameter section below, we will outline some example parameter specifications for how to implement different soil media types for the first layer of a cell. Also, note that the an impermeable layer must be simulated using parameter choices in order to reduce or eliminate all flow from the green roof to the belowground layers 2-4. This will be specifically described in a following section.

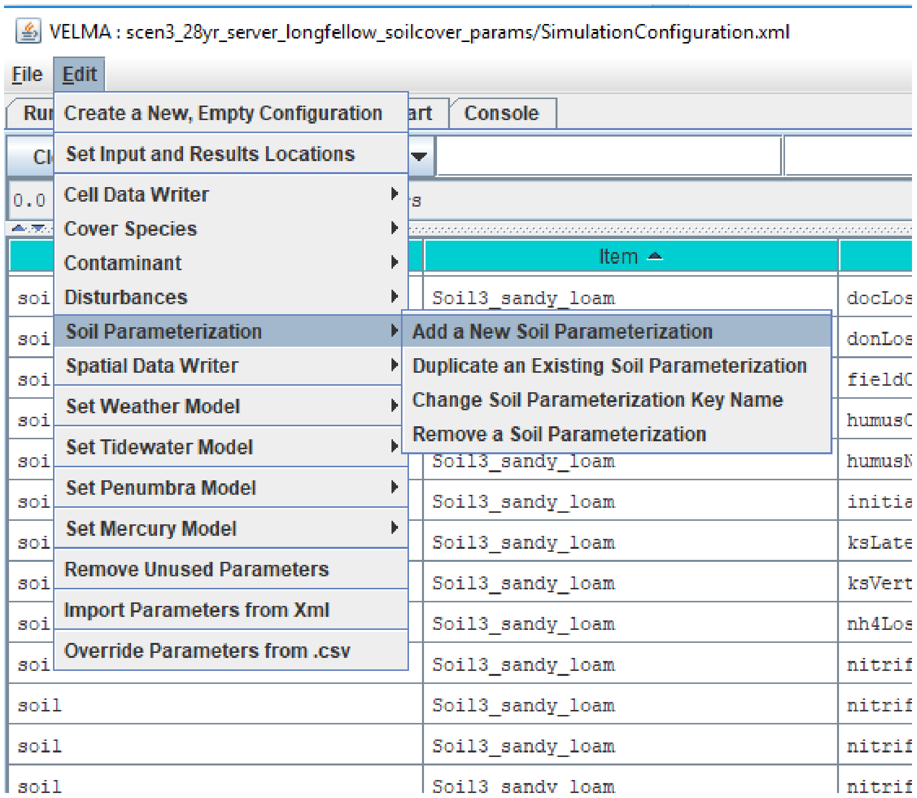
The next two sections will describe how the soil and cover parameters can be altered to implement green roofs on a given cell (or multiple cells). Then, we will specifically describe how to change parameterizations to reduce or eliminate flow from the first layer to the other layers (2-4).

Soil Type Specifications

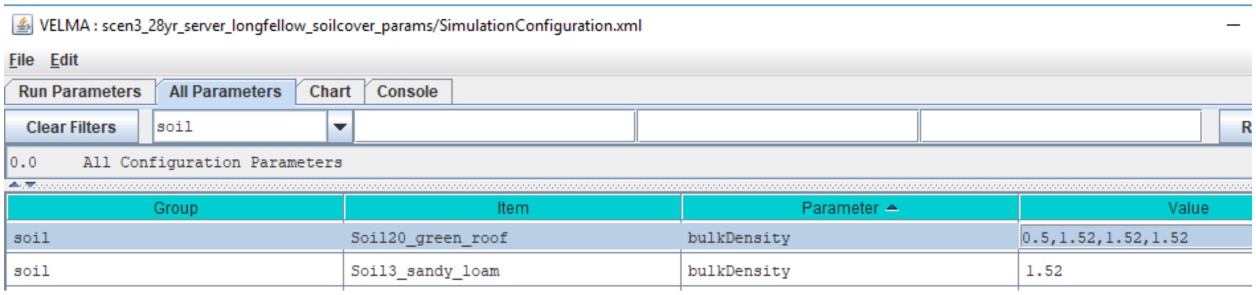
The table below shows example soil type specifications for intensive and extensive green roofs. These data were obtained from Roof-Lite Media Data sheets (Skyland USA, LLC). These media characteristics have been approved for use as green roof media in Seattle, Washington to be eligible for stormwater credits. Note that you will want to verify these soil parameters and characteristics for your particular watershed before implementing these values.

Properties	**General (Sandy Loam)**	**Extensive Roofs (Top Layer)**	**Intensive Roofs (Top Layer)**
Porosity (v/v)	0.453	0.70	0.7
Field Capacity (v/v)	0.207	0.60	0.65
Wilt Point (v/v)	0.095	0.12	0.12
Hydraulic Conductivity (mm/day)	770	87,000	10,000
Bulk Density (g/cm^3)	1.52	0.50	0.75
Depth (mm)	500	100	500

To specify these parameters in VELMA, create a new soil parameterization for Green Roofs.


VELMA : scen3_28yr_server_longfellow_soilcover_params/SimulationConfiguration.xml
File
Edit
Create a New, Empty Configuration
Set Input and Results Locations
Cell Data Writer
Cover Species
Contaminant
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Set Penumbra Model
Set Mercury Model
Remove Unused Parameters
Import Parameters from Xml
Override Parameters from .csv
Add a New Soil Parameterization
Duplicate an Existing Soil Parameterization
Change Soil Parameterization Key Name
Remove a Soil Parameterization
Console
Item
Soil3_sandy_loam

Then, set the parameters for the Green Roof soil to only apply to the 1st soil layer, as shown below.


VELMA : scen3_28yr_server_longfellow_soilcover_params/SimulationConfiguration.xml
File
Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
soil
0.0 All Configuration Parameters

Group	Item	Parameter	Value
soil	Soil20_green_roof	bulkDensity	0.5,1.52,1.52,1.52
soil	Soil3_sandy_loam	bulkDensity	1.52

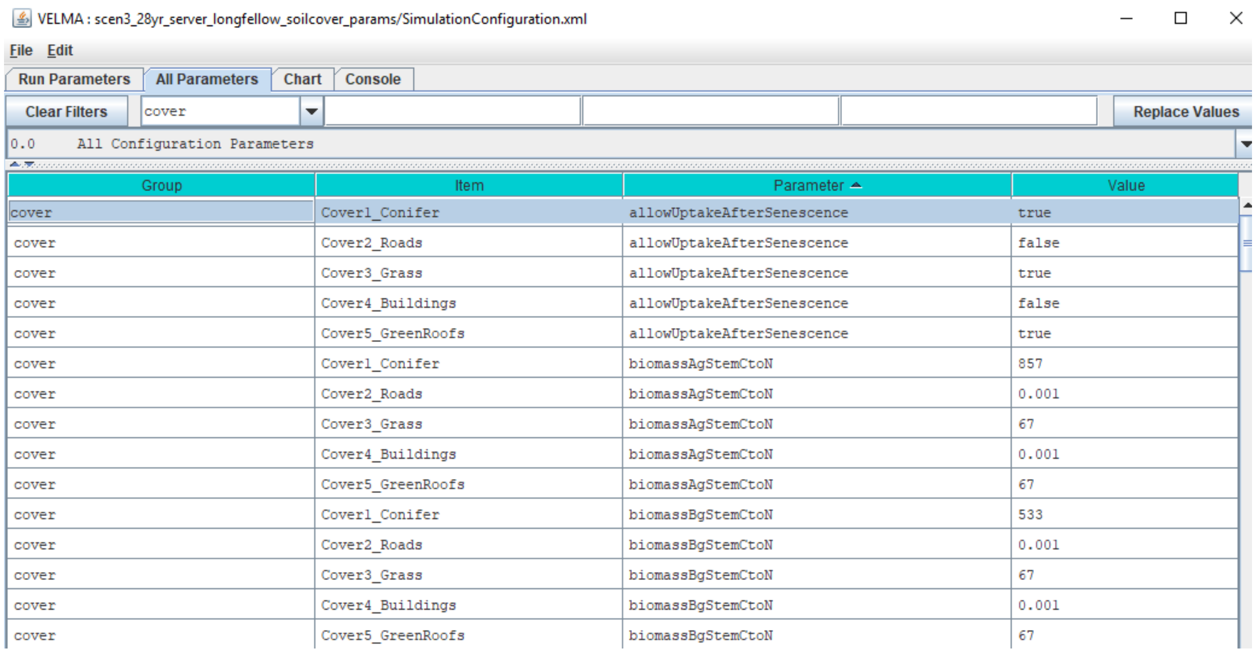
In the above screenshot, we have two soil layers (Soil20_green_roof and Soil3_sandy_loam). Note however, that the change in the bulkDensity parameter for the Green Roof soil type only pertains to the first soil layer. This can be done for the remaining soil parameters, e.g., porosity, field capacity, depth.

For a list of all relevant soil parameters that are parameterized to use Green Roofs, please see the VELMA .xml file of the Longfellow watershed in Seattle, Washington, which is provided as an example.

Cover Type Specifications

Extensive green roofs can support low-level biomass growth, while intensive green roofs can support much higher biomass values. We used Sedum album data from Getter et al. (2009), which provided average estimates of 240 and 1000 gCm^2yr^{-1} for extensive and intensive green roofs, respectively.

There a large number of parameters that need to be properly parameterized to match the growth of aboveground and belowground biomass for a particular cell. Below is an example showing fiver cover types, including Conifer, Roads, Grass, Buildings, and Green Roofs. In some cases – for example, with extensive green roofs – you will want to match your parameterizations of the “Grass” cover type with the green roof cover. In other cases, you may want to use blends of parameterizations to represent species mixtures.


VELMA : scen3_28yr_server_longfellow_soilcover_params/SimulationConfiguration.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
cover
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
cover	Cover1_Conifer	allowUptakeAfterSenescence	true
cover	Cover2_Roads	allowUptakeAfterSenescence	false
cover	Cover3_Grass	allowUptakeAfterSenescence	true
cover	Cover4_Buildings	allowUptakeAfterSenescence	false
cover	Cover5_GreenRoofs	allowUptakeAfterSenescence	true
cover	Cover1_Conifer	biomassAgStemCtoN	857
cover	Cover2_Roads	biomassAgStemCtoN	0.001
cover	Cover3_Grass	biomassAgStemCtoN	67
cover	Cover4_Buildings	biomassAgStemCtoN	0.001
cover	Cover5_GreenRoofs	biomassAgStemCtoN	67
cover	Cover1_Conifer	biomassBgStemCtoN	533
cover	Cover2_Roads	biomassBgStemCtoN	0.001
cover	Cover3_Grass	biomassBgStemCtoN	67
cover	Cover4_Buildings	biomassBgStemCtoN	0.001
cover	Cover5_GreenRoofs	biomassBgStemCtoN	67

For a list of all relevant cover parameters that are parameterized to use Green Roofs, please see the VELMA .xml file of the Longfellow watershed in Seattle, Washington, which is provided as an example.

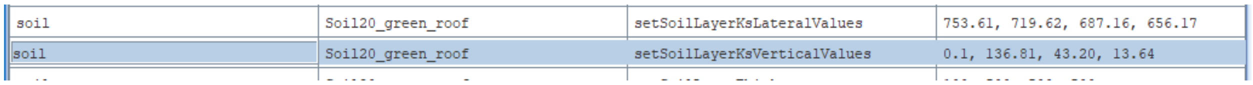
Limiting Flow from the 1st Layer to the Underground Layers

The final step to implementing green roofs in VELMA requires the user to reduce the amount of flow from the first layer to the second through fourth layers to a near-zero value. Note that the first layer represents the media contained in the green roof, and the other layers represent the soil material underneath the buildings; that is why we wish to remove the vertical flow from the first to the other layers.

The mechanism to perform this operation is the following VELMA parameter: */soil/<GreenRoofSoilName>/setSoilLayerKsVerticalValues.*

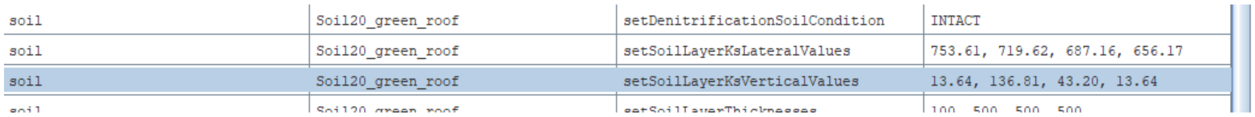
As shown in the following screenshot, the first of four values should be reduced to a near-zero value. Note that using 0 (or even 0.1) will oftentimes cause the model to crash, so the user will need to test different possibilities to find a solution that will not crash the model yet simulate the desired mechanism.

An example of a parameterization that resulted in a crashed model run.



soil	Soil20_green_roof	setSoilLayerKsLateralValues	753.61, 719.62, 687.16, 656.17
soil	Soil20_green_roof	setSoilLayerKsVerticalValues	0.1, 136.81, 43.20, 13.64

An example of a parameterization that resulted in a successful model run.



soil	Soil20_green_roof	setDenitrificationSoilCondition	INTACT
soil	Soil20_green_roof	setSoilLayerKsLateralValues	753.61, 719.62, 687.16, 656.17
soil	Soil20_green_roof	setSoilLayerKsVerticalValues	13.64, 136.81, 43.20, 13.64
soil	Soil20_green_roof	setSoilLayerThicknesses	100, 500, 500, 500

Ultimately, the user will need to validate the use of particular ksVert and ksLat values by comparing simulated with observed data.

Conclusions

This document has outlined some basic ways to incorporate green roofs within VELMA. Much testing and validation is needed to verify that this mechanistic approach is suitable for simulating green roofs in particular watersheds. We emphasize the need to validate VELMA with observed data – not only at a single gauge at the watershed’s outlet but also with other site-specific or average observed data (e.g., soil moisture, evapotranspiration) that can be used to further constrain the model for particular green roof implementations.

References

Getter, K.L., Rowe, D.B., Robertson, G.P., Cregg, B.M. and Andresen, J.A., 2009. Carbon sequestration potential of extensive green roofs. *Environmental science & technology*, *43*(19), pp.7564-7570.

Skyland USA, LLC, Data sheets for parameterization information for different types of green roofs. https://www.rooflitesoil.com/downloads/

E.5 | Irrigation Disturbance

Overview *(Tutorial E.5 – Add Irrigation Disturbance)*

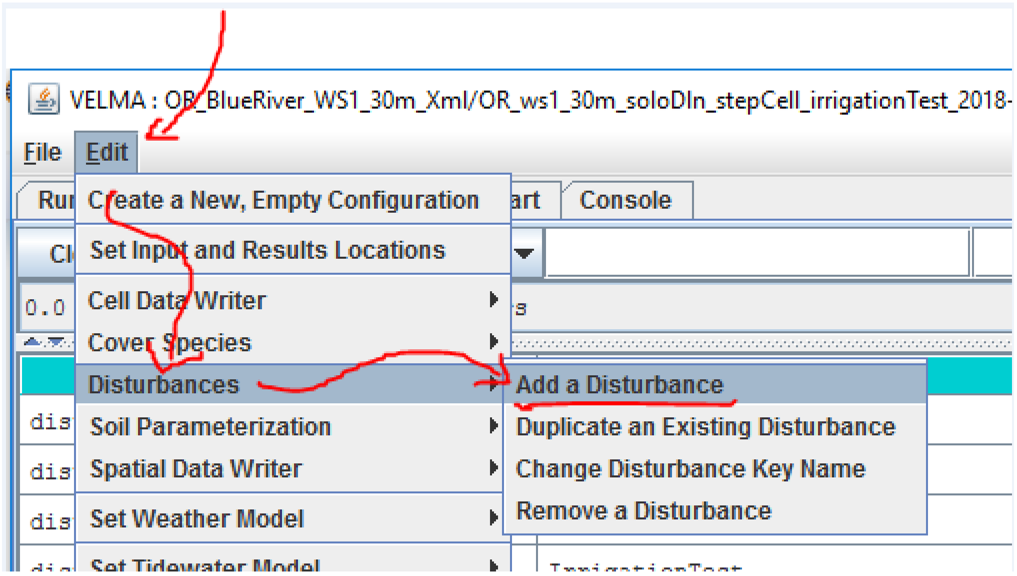
This document describes how to build an Irrigation Disturbance in VELMA. This provides a way to add specific amounts of water to specific categories of cells within the simulated watershed, at specific steps during the simulation run, and for specific precipitation conditions.

Use the Irrigation Disturbance routine in VELMA to specify when (Julian days) and where (mapped cell locations) you want to add irrigation water. The irrigation water is added "as-if" it is a rain addition for the specified cell(s). This prevents irrigation water additions from perturbing VELMA's water balance mechanisms.

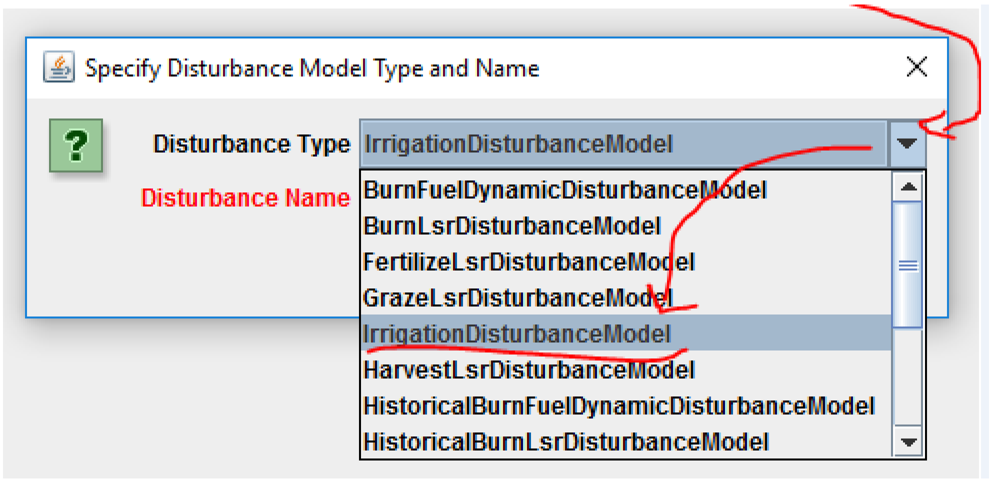
Note that cell locations to be irrigated are limited to the Cover Types you specify. For example, suppose you wish to irrigate only a subset of grassland areas amongst all "Grassland" cover type areas within the simulated watershed. In that case, you will need to create a cover type named "IrrigatedGrassland" (or something similar), and assign a unique integer to all cells for that new cover type in the watershed's Cover Type map. Otherwise all cells designated in the Cover Type map as "Grassland" will be irrigated.

Using the Irrigation Disturbance

Add an Irrigation Disturbance to your simulation configuration in JVelma via the menu Edit -> Disturbances -> Add a Disturbance item.


VELMA : OR_BlueRiver_WS1_30m_Xml/OR_ws1_30m_soloDIn_stepCell_irrigationTest_2018-
File
Edit
Create a New, Empty Configuration
Set Input and Results Locations
Cell Data Writer
Cover Species
Disturbances
Soil Parameterization
Spatial Data Writer
Set Weather Model
Set Tidewater Model
Add a Disturbance
Duplicate an Existing Disturbance
Change Disturbance Key Name
Remove a Disturbance
Console

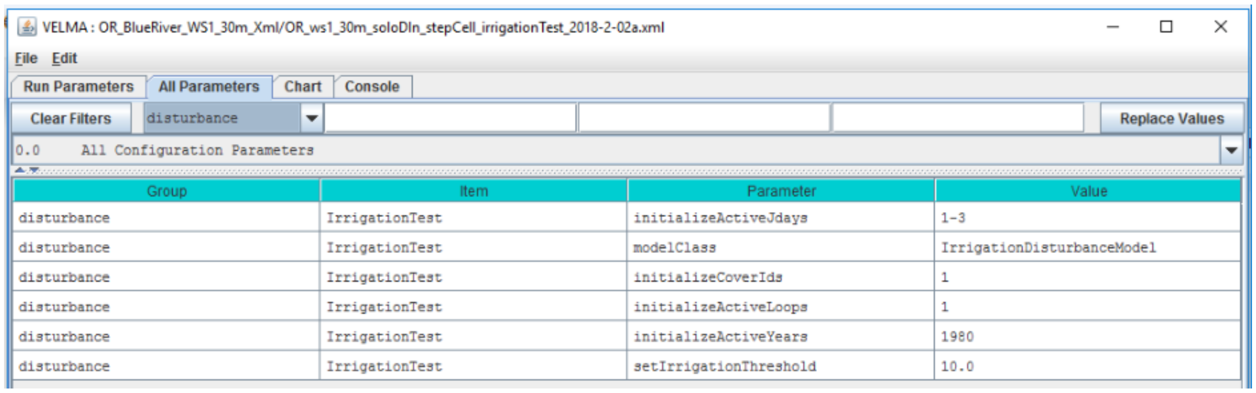
Clicking the "Add a Disturbance" item opens the "Specify Disturbance Model Type and Name" dialog. Select "IrrigationDisturbanceModel" in the dropdown selector as the model type.


Specify Disturbance Model Type and Name
Disturbance Type
IrrigationDisturbanceModel
Disturbance Name
BurnFuelDynamicDisturbanceModel
BurnLsrDisturbanceModel
FertilizeLsrDisturbanceModel
GrazeLsrDisturbanceModel
IrrigationDisturbanceModel
HarvestLsrDisturbanceModel
HistoricalBurnFuelDynamicDisturbanceModel
HistoricalBurnLsrDisturbanceModel

The Disturbance Name can be any meaningful text name, but it can only be composed of letters of the alphabet, the digits 0-9, and the underscore ("_") character.
It must also be unique from any other disturbances named in the same simulation configuration.

After entering a name for the new Irrigation Disturbance, click the OK button, and JVelma will create a new parameterization group and filter the parameters table view to show only its parameters in the "All Parameters" tab.

Here is an example of a new Irrigation Disturbance, created with the name "Irrigation Test":


VELMA : OR_BlueRiver_WS1_30m_Xml/OR_ws1_30m_soloDIn_stepCell_irrigationTest_2018-2-02a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
disturbance
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
disturbance	IrrigationTest	initializeActiveJdays	1-3
disturbance	IrrigationTest	modelClass	IrrigationDisturbanceModel
disturbance	IrrigationTest	initializeCoverIds	1
disturbance	IrrigationTest	initializeActiveLoops	1
disturbance	IrrigationTest	initializeActiveYears	1980
disturbance	IrrigationTest	setIrrigationThreshold	10.0

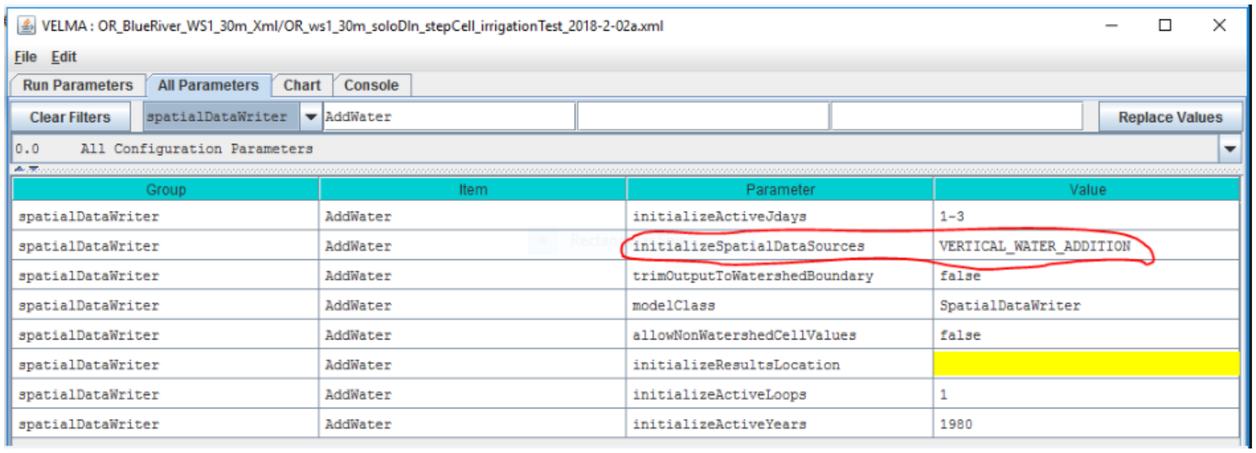
The example parameterization above schedules irrigation to occur for cells with Cover Type = 1 during the first simulation loop, on the first 3 days of year 1980.

Even during the 3 days of 1980 when irrigation *can* occur, it will *only* occur for Cover Type 1 cells in which (`irrigationThreshold` - `totalPrecipitation`) > 0.0.
(Where `totalPrecipitation` is the total precipitation for a given cell on a given day – e.g. if `irrigationThreshold` = 10.0, but the `totalPrecipitation` = 5.0, no irrigation occurs even if the cell is otherwise-eligible.)

Capturing Irrigation Details in VELMA Simulation Results

To observe how much irrigation occurs (and when it occurs), add a `SpatialDataWriter` parameterization for the `VERTICAL_WATER_ADDITION` pool to your simulation configuration.

Here is an example, configured to emit .asc maps on the same days as the example Irrigation Disturbance configuration given above.


VELMA : OR_BlueRiver_WS1_30m_Xml/OR_ws1_30m_soloDIn_stepCell_irrigationTest_2018-2-02a.xml
File Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
spatialDataWriter
AddWater
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
spatialDataWriter	AddWater	initializeActiveJdays	1-3
spatialDataWriter	AddWater	initializeSpatialDataSources	VERTICAL_WATER_ADDITION
spatialDataWriter	AddWater	trimOutputToWatershedBoundary	false
spatialDataWriter	AddWater	modelClass	SpatialDataWriter
spatialDataWriter	AddWater	allowNonWatershedCellValues	false
spatialDataWriter	AddWater	initializeResultsLocation	
spatialDataWriter	AddWater	initializeActiveLoops	1
spatialDataWriter	AddWater	initializeActiveYears	1980

Be aware that the `VERTICAL_WATER_ADDITION` pool is shared by other disturbances and simulator engine mechanisms.

If you have Groundwater Storage, or Water Drain Disturbance configured and active in your simulation, the `VERTICAL_WATER_ADDITION` amounts represent the sum of the contributors than share the pool. (Currently, however, there are no other results reporting mechanisms for irrigation amounts.)

E.6 | Groundwater Storage Mechanism

Overview *(Tutorial E.6_Groundwater Storage Box)*

This document describes how to configure VELMA’s groundwater storage mechanism, an optional method for simulating vertical drainage of excess (over saturation) water to a pool of deep groundwater. Drainage to groundwater only occurs when all the layers (1-4) in a cell’s soil column are fully saturated.

When the groundwater storage mechanism is not invoked – for example when bedrock or other impermeable layer is present below layer 4 – no drainage to deep groundwater occurs and VELMA simulates upwelling of water within the saturated soil column to the surface.

The VELMA simulator can handle over-saturation water amounts in two ways

Due to its original model design, and by its default implementation, the VELMA Simulator behaves as if there is an impermeable bedrock layer underlying the bottom layer of each cell’s soil column.
When all the layers in a cell’s column are fully saturated, any additional water input on a given day “rebounds” up the water column to the surface of the cell, and – if the soil layer column is completely saturated -- becomes standing (i.e. surface) water for the cell.

However, the VELMA simulator also provides an alternate mechanism – the groundwater “storage box” – for handling water amounts over the fully-saturated amount a cell’s soil column can contain.
When the groundwater storage mechanism is enabled (as part of a simulation’s configuration) a specified fraction [0.0 to 1.0] of the excess water that reaches the bottom of a saturated soil column is removed from the column and accounted for in a single “storage box” amount accumulator. The remaining excess water “rebounds” up the soil column. The fraction of the excess water transferred to the ‘storage box” is, in effect, removed from the watershed’s water dynamics.

The amount of water transferred to the groundwater storage box is tracked and reported, but is *not* spatially explicit, and is not returnable to the watershed after it is transferred to the “box”.
In effect, water in the storage box has been removed from the simulation system.

However, the fraction that the determines the amount of water transferred may be either global or spatially explicit. I.e. different groundwater fractions may be specified for each cell in the watershed.

In testing, enabling the groundwater storage mechanism has the general effects of reducing watershed total runoff, and “dampening” the reactivity of the day-to-day runoff relative to water input (precipitation, snow melt, etc.).

Enabling the Groundwater Storage Mechanism

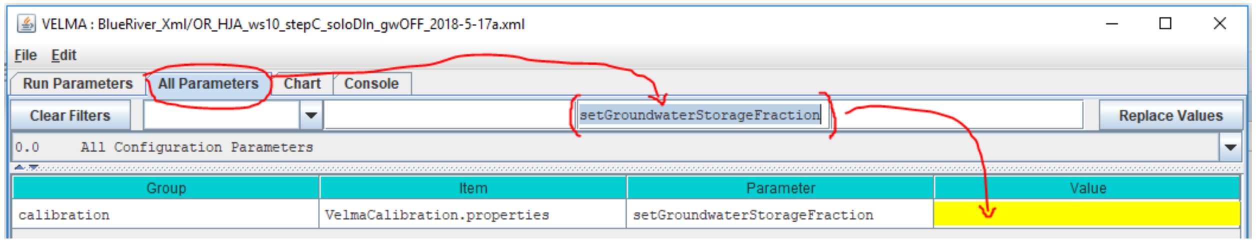
The groundwater storage mechanism is enabled/disabled by the `setGroundwaterStorageFraction` parameter’s value, which also specifies the global fraction or map of cell-specific fractions the mechanism employs.

The following table indicates the three types of values the parameter accepts:

Value of setGroundwaterStorageFraction	**Groundwater Mechanism Status and Behavior**
Empty/blank	**Inactive** No groundwater storage occurs.
Single numeric value in range [0, 1]	**Active** Specified value is fraction used for every cell’s groundwater calculation.
Filename of an .asc map containing values, each in range [0, 1]	**Active** Specified map’s values provide cell-specific fractions for groundwater calculations.

To find and set the `setGroundwaterStorageFraction` parameter in JVelma:

1. Click the “`All Parameters`” tab.
2. Type the (case-sensitive) text `setGroundwaterStorageFraction` into the middle text-entry filter field.
3. Double-click the “Value” field of the `setGroundwaterStorageFraction` parameter row, and enter a value.


VELMA : BlueRiver_Xml/OR_HJA_ws10_stepC_soloDln_gwOFF_2018-5-17a.xml
File
Edit
Run Parameters
All Parameters
Chart
Console
Clear Filters
setGroundwaterStorageFraction
Replace Values
0.0 All Configuration Parameters

Group	Item	Parameter	Value
calibration	VelmaCalibration.properties	setGroundwaterStorageFraction	

To ***disable*** the groundwater storage mechanism set the `setGroundwaterStorageFraction` paremeter’s value to empty/blank.
(Note: The value 0.0 has the same effect, but use empty/blank, because this clearly signals “disable the groundwater mechanism” to the VELMA simulator.)

Water “Lost” to the Groundwater Storage Box is Reported in Simulation Results

When the `enableGroundwaterStorageBox` parameter is set to `true`, over-saturation water amounts are transferred to the storage box, and are effectively removed from the watershed. (Currently, there is no mechanism for reintroducing water transferred to the storage box back into the simulation.)

The VELMA simulator reports the groundwater storage amount in the following results:

1. In the simulation runtime log file.
   Usually named `GlobalStateLog.txt` (unless user-configured for a different name), the runtime log contains an entry reporting the total amount of water (in mm) transferred to (and thus accumulated in) the groundwater storage box over the course of the entire simulation.

Here is an example of the output:

```
INFO 12:02:39 VelmaSimulatorEngine: Groundwater Storage true : storage
amount=4.203496685912282E8
```

2. In the `DailyResults.csv` file.
   Simulation runs with `enableGroundwaterStorageBox == true` write the per-day amount of water to a column named `Groundwater_Storage_Addition(mm/day).`
   The amount is not an averaged-per-cell; it is the total amount of water in the storage box.
   To determine the addition (as opposed to the total) amount of water added to the storage box on a given day, subtract the given day’s amount from the prior day’s amount.

3. In Cell Data Writer `.csv` results files.
   If a groundwater storage-enabled simulation run also has one or more Cell Data Writers configured, each Cell Data Writer’s results file will contain a column named `Groundwater_Storage_Addition`. This column reports (in mm) the amount of water transferred from to cell to groundwater storage, per day. The values are not cumulative. To determine the cumulative amount over a span of days, sum the amounts in this data column for the days in question.

E.7 | Spatial Data Writer Key Words for Generating ASCII VELMA Outputs for Data Visualization

Overview *(Tutorial E.7_Spatial Data Writer Key Words for ASCII Data Visualization)*

The VELMA GUI can be used to set up spatial data writer keywords corresponding to specific hydrological and biogeochemical VELMA outputs. VELMA has a spatial data writer disturbance submodel that users can configure to produce ASCII output files (georeferenced grid cell arrays) that are saved to the VELMA results folder. See the VELMA 2.0 User Manual (McKane et al. 2014) for details on how to set up VELMA spatial data writer disturbances.

The purpose of the present tutorial (E.8) is to update available spatial data writer keywords that must be specified if a user wants VELMA to produce ASCII files for outputs of interest. After a simulation ends, the user can load the ASCII files into GIS-based data visualization tools such as ArcGIS or VISTAS (http://blogs.evergreen.edu/vistas/vistas-software/).

Note: The 2014 VELMA User Manual is downloadable from the EPA VELMA website: https://www.epa.gov/water-research/visualizing-ecosystem-land-management-assessments-velma-model-20

How to set up VELMA spatial data writers with user-specified keywords

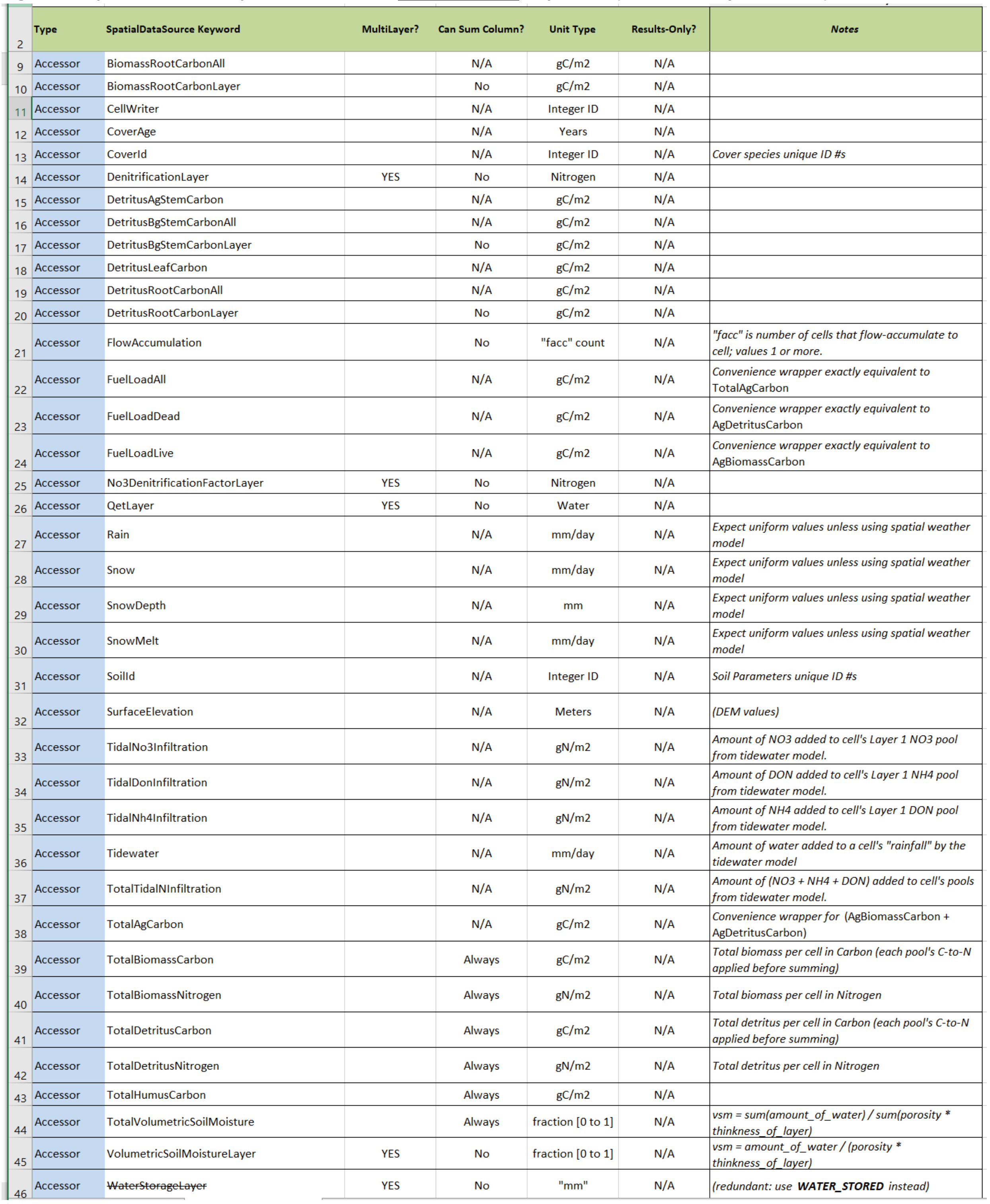
Section 23.4 of the VELMA 2.0 User Manual (McKane et al. 2014) provides instructions for implementing spatial data writers in VELMA 2.0 or 2.1. This tutorial is limited to updating the “Table of Spatial Data Sources” (keywords) described in the 2014 User Manual.

Spatial Data Writer Keyword Data Types

There are four kinds of keyword data types that users can specify when configuring spatial data writer VELMA outputs. These codes appear in the first column of Figure 1.

- Accessor = Spatial Data that is not directly stored in a spatial data pool object
- Direct = Spatial Data that is directly stored in a spatial data pool object
- Disturbance = Spatial Data **automatically written** (when enabled) by disturbances.
- Buffered = Spatial Data directly stored in a spatial data pool object, but only when a SpatialDataWriter requires it.

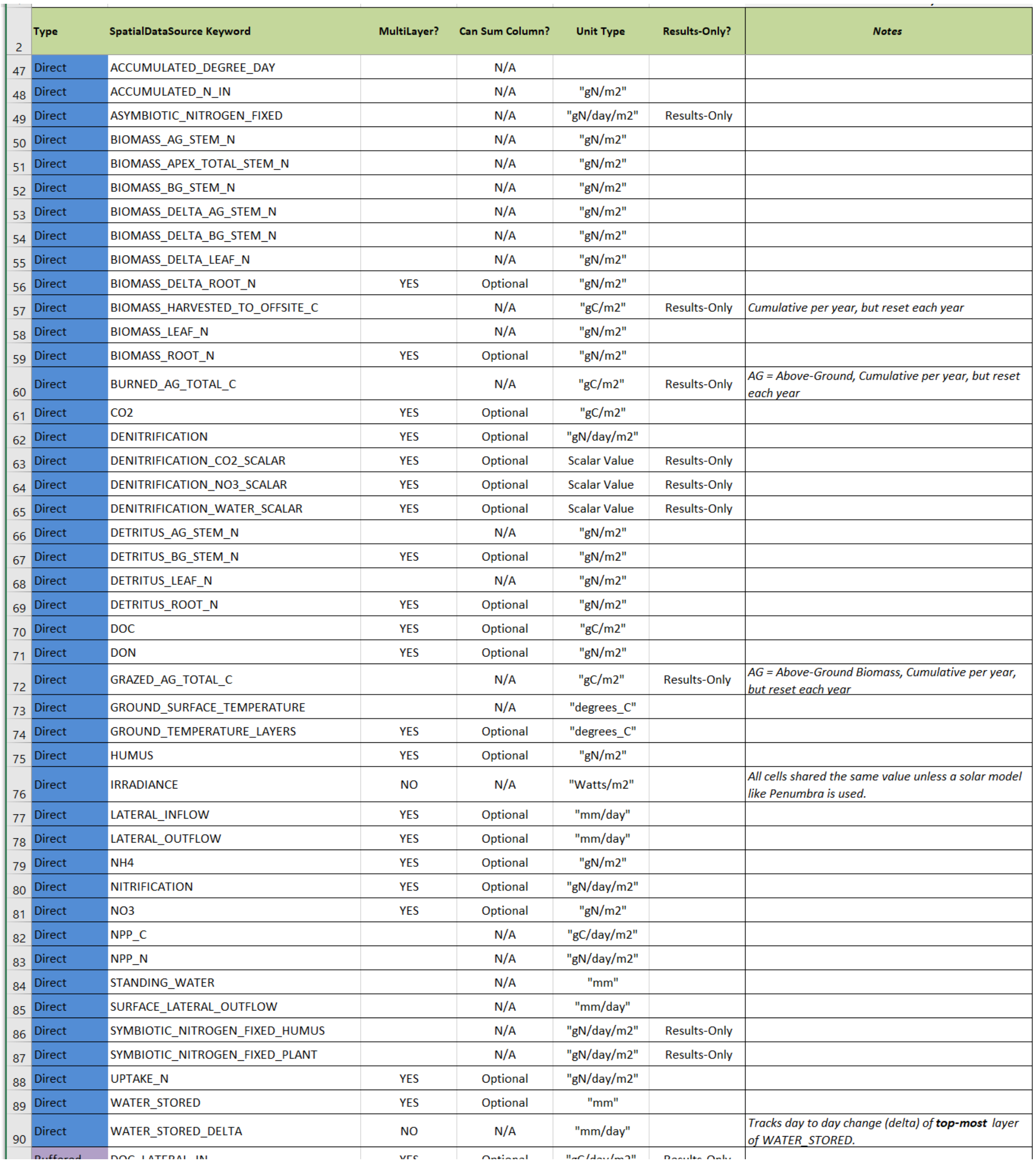
Figure 1. Updated VELMA spatial data writer case sensitive keywords (enter exactly as shown)



	Type	SpatialDataSource Keyword	MultiLayer?	Can Sum Column?	Unit Type	Results-Only?	*Notes*
2							
9	Accessor	BiomassRootCarbonAll		N/A	gC/m2	N/A	
10	Accessor	BiomassRootCarbonLayer		No	gC/m2	N/A	
11	Accessor	CellWriter		N/A	Integer ID	N/A	
12	Accessor	CoverAge		N/A	Years	N/A	
13	Accessor	CoverId		N/A	Integer ID	N/A	*Cover species unique ID #s*
14	Accessor	DenitrificationLayer	YES	No	Nitrogen	N/A	
15	Accessor	DetritusAgStemCarbon		N/A	gC/m2	N/A	
16	Accessor	DetritusBgStemCarbonAll		N/A	gC/m2	N/A	
17	Accessor	DetritusBgStemCarbonLayer		No	gC/m2	N/A	
18	Accessor	DetritusLeafCarbon		N/A	gC/m2	N/A	
19	Accessor	DetritusRootCarbonAll		N/A	gC/m2	N/A	
20	Accessor	DetritusRootCarbonLayer		No	gC/m2	N/A	
21	Accessor	FlowAccumulation		No	"facc" count	N/A	*"facc" is number of cells that flow-accumulate to cell; values 1 or more.*
22	Accessor	FuelLoadAll		N/A	gC/m2	N/A	*Convenience wrapper exactly equivalent to* TotalAgCarbon
23	Accessor	FuelLoadDead		N/A	gC/m2	N/A	*Convenience wrapper exactly equivalent to* AgDetritusCarbon
24	Accessor	FuelLoadLive		N/A	gC/m2	N/A	*Convenience wrapper exactly equivalent to* AgBiomassCarbon
25	Accessor	No3DenitrificationFactorLayer	YES	No	Nitrogen	N/A	
26	Accessor	QetLayer	YES	No	Water	N/A	
27	Accessor	Rain		N/A	mm/day	N/A	*Expect uniform values unless using spatial weather model*
28	Accessor	Snow		N/A	mm/day	N/A	*Expect uniform values unless using spatial weather model*
29	Accessor	SnowDepth		N/A	mm	N/A	*Expect uniform values unless using spatial weather model*
30	Accessor	SnowMelt		N/A	mm/day	N/A	*Expect uniform values unless using spatial weather model*
31	Accessor	SoilId		N/A	Integer ID	N/A	*Soil Parameters unique ID #s*
32	Accessor	SurfaceElevation		N/A	Meters	N/A	*(DEM values)*
33	Accessor	TidalNo3Infiltration		N/A	gN/m2	N/A	*Amount of NO3 added to cell's Layer 1 NO3 pool from tidewater model.*
34	Accessor	TidalDonInfiltration		N/A	gN/m2	N/A	*Amount of DON added to cell's Layer 1 NH4 pool from tidewater model.*
35	Accessor	TidalNh4Infiltration		N/A	gN/m2	N/A	*Amount of NH4 added to cell's Layer 1 DON pool from tidewater model.*
36	Accessor	Tidewater		N/A	mm/day	N/A	*Amount of water added to a cell's "rainfall" by the tidewater model*
37	Accessor	TotalTidalNInfiltration		N/A	gN/m2	N/A	*Amount of (NO3 + NH4 + DON) added to cell's pools from tidewater model.*
38	Accessor	TotalAgCarbon		N/A	gC/m2	N/A	*Convenience wrapper for* (AgBiomassCarbon + AgDetritusCarbon)
39	Accessor	TotalBiomassCarbon		Always	gC/m2	N/A	*Total biomass per cell in Carbon (each pool's C-to-N applied before summing)*
40	Accessor	TotalBiomassNitrogen		Always	gN/m2	N/A	*Total biomass per cell in Nitrogen*
41	Accessor	TotalDetritusCarbon		Always	gC/m2	N/A	*Total detritus per cell in Carbon (each pool's C-to-N applied before summing)*
42	Accessor	TotalDetritusNitrogen		Always	gN/m2	N/A	*Total detritus per cell in Nitrogen*
43	Accessor	TotalHumusCarbon		Always	gC/m2	N/A	
44	Accessor	TotalVolumetricSoilMoisture		Always	fraction [0 to 1]	N/A	*vsm = sum(amount_of_water) / sum(porosity * thinkness_of_layer)*
45	Accessor	VolumetricSoilMoistureLayer	YES	No	fraction [0 to 1]	N/A	*vsm = amount_of_water / (porosity * thinkness_of_layer)*
46	Accessor	~~WaterStorageLayer~~	YES	No	"mm"	N/A	*(redundant: use* ***WATER_STORED*** *instead)*

(continued)

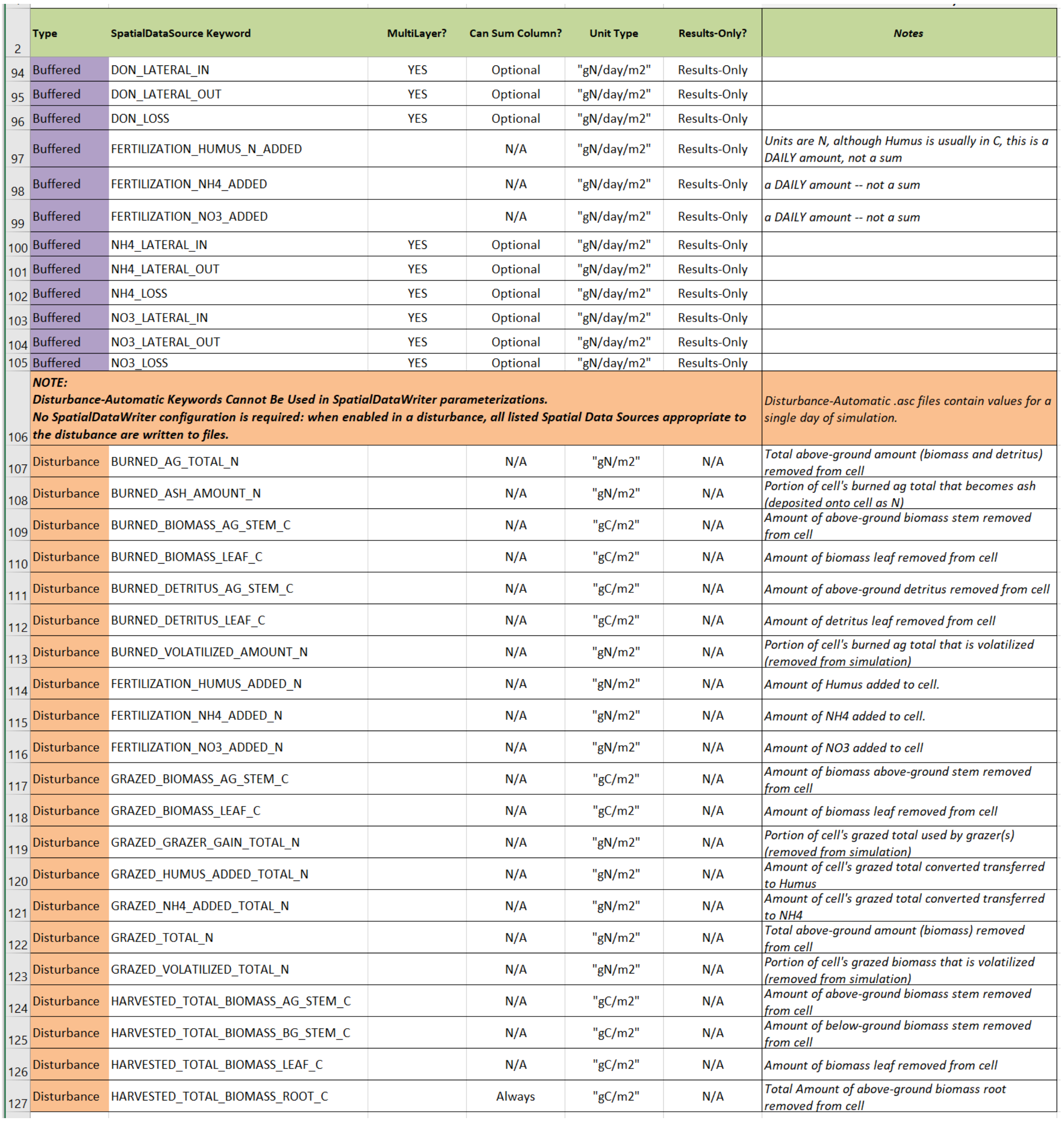
Figure 1 (continued). Updated VELMA spatial data writer case sensitive keywords (enter exactly as shown)



2	Type	SpatialDataSource Keyword	MultiLayer?	Can Sum Column?	Unit Type	Results-Only?	Notes
47	Direct	ACCUMULATED_DEGREE_DAY		N/A			
48	Direct	ACCUMULATED_N_IN		N/A	"gN/m2"		
49	Direct	ASYMBIOTIC_NITROGEN_FIXED		N/A	"gN/day/m2"	Results-Only	
50	Direct	BIOMASS_AG_STEM_N		N/A	"gN/m2"		
51	Direct	BIOMASS_APEX_TOTAL_STEM_N		N/A	"gN/m2"		
52	Direct	BIOMASS_BG_STEM_N		N/A	"gN/m2"		
53	Direct	BIOMASS_DELTA_AG_STEM_N		N/A	"gN/m2"		
54	Direct	BIOMASS_DELTA_BG_STEM_N		N/A	"gN/m2"		
55	Direct	BIOMASS_DELTA_LEAF_N		N/A	"gN/m2"		
56	Direct	BIOMASS_DELTA_ROOT_N	YES	Optional	"gN/m2"		
57	Direct	BIOMASS_HARVESTED_TO_OFFSITE_C		N/A	"gC/m2"	Results-Only	*Cumulative per year, but reset each year*
58	Direct	BIOMASS_LEAF_N		N/A	"gN/m2"		
59	Direct	BIOMASS_ROOT_N	YES	Optional	"gN/m2"		
60	Direct	BURNED_AG_TOTAL_C		N/A	"gC/m2"	Results-Only	*AG = Above-Ground, Cumulative per year, but reset each year*
61	Direct	CO2	YES	Optional	"gC/m2"		
62	Direct	DENITRIFICATION	YES	Optional	"gN/day/m2"		
63	Direct	DENITRIFICATION_CO2_SCALAR	YES	Optional	Scalar Value	Results-Only	
64	Direct	DENITRIFICATION_NO3_SCALAR	YES	Optional	Scalar Value	Results-Only	
65	Direct	DENITRIFICATION_WATER_SCALAR	YES	Optional	Scalar Value	Results-Only	
66	Direct	DETRITUS_AG_STEM_N		N/A	"gN/m2"		
67	Direct	DETRITUS_BG_STEM_N	YES	Optional	"gN/m2"		
68	Direct	DETRITUS_LEAF_N		N/A	"gN/m2"		
69	Direct	DETRITUS_ROOT_N	YES	Optional	"gN/m2"		
70	Direct	DOC	YES	Optional	"gC/m2"		
71	Direct	DON	YES	Optional	"gN/m2"		
72	Direct	GRAZED_AG_TOTAL_C		N/A	"gC/m2"	Results-Only	*AG = Above-Ground Biomass, Cumulative per year, but reset each year*
73	Direct	GROUND_SURFACE_TEMPERATURE		N/A	"degrees_C"		
74	Direct	GROUND_TEMPERATURE_LAYERS	YES	Optional	"degrees_C"		
75	Direct	HUMUS	YES	Optional	"gN/m2"		
76	Direct	IRRADIANCE	NO	N/A	"Watts/m2"		*All cells shared the same value unless a solar model like Penumbra is used.*
77	Direct	LATERAL_INFLOW	YES	Optional	"mm/day"		
78	Direct	LATERAL_OUTFLOW	YES	Optional	"mm/day"		
79	Direct	NH4	YES	Optional	"gN/m2"		
80	Direct	NITRIFICATION	YES	Optional	"gN/day/m2"		
81	Direct	NO3	YES	Optional	"gN/m2"		
82	Direct	NPP_C		N/A	"gC/day/m2"		
83	Direct	NPP_N		N/A	"gN/day/m2"		
84	Direct	STANDING_WATER		N/A	"mm"		
85	Direct	SURFACE_LATERAL_OUTFLOW		N/A	"mm/day"		
86	Direct	SYMBIOTIC_NITROGEN_FIXED_HUMUS		N/A	"gN/day/m2"	Results-Only	
87	Direct	SYMBIOTIC_NITROGEN_FIXED_PLANT		N/A	"gN/day/m2"	Results-Only	
88	Direct	UPTAKE_N	YES	Optional	"gN/day/m2"		
89	Direct	WATER_STORED	YES	Optional	"mm"		
90	Direct	WATER_STORED_DELTA	NO	N/A	"mm/day"		*Tracks day to day change (delta) of **top-most** layer of WATER_STORED.*
	Buffered	DOC_LATERAL_IN	YES	Optional	"gC/day/m2"	Results-Only	

(continued)

Figure 1 (continued). Updated VELMA spatial data writer case sensitive keywords (enter exactly as shown)



2	Type	SpatialDataSource Keyword	MultiLayer?	Can Sum Column?	Unit Type	Results-Only?	Notes
94	Buffered	DON_LATERAL_IN	YES	Optional	"gN/day/m2"	Results-Only	
95	Buffered	DON_LATERAL_OUT	YES	Optional	"gN/day/m2"	Results-Only	
96	Buffered	DON_LOSS	YES	Optional	"gN/day/m2"	Results-Only	
97	Buffered	FERTILIZATION_HUMUS_N_ADDED		N/A	"gN/day/m2"	Results-Only	*Units are N, although Humus is usually in C, this is a DAILY amount, not a sum*
98	Buffered	FERTILIZATION_NH4_ADDED		N/A	"gN/day/m2"	Results-Only	*a DAILY amount -- not a sum*
99	Buffered	FERTILIZATION_NO3_ADDED		N/A	"gN/day/m2"	Results-Only	*a DAILY amount -- not a sum*
100	Buffered	NH4_LATERAL_IN	YES	Optional	"gN/day/m2"	Results-Only	
101	Buffered	NH4_LATERAL_OUT	YES	Optional	"gN/day/m2"	Results-Only	
102	Buffered	NH4_LOSS	YES	Optional	"gN/day/m2"	Results-Only	
103	Buffered	NO3_LATERAL_IN	YES	Optional	"gN/day/m2"	Results-Only	
104	Buffered	NO3_LATERAL_OUT	YES	Optional	"gN/day/m2"	Results-Only	
105	Buffered	NO3_LOSS	YES	Optional	"gN/day/m2"	Results-Only	
106	***NOTE: Disturbance-Automatic Keywords Cannot Be Used in SpatialDataWriter parameterizations. No SpatialDataWriter configuration is required: when enabled in a disturbance, all listed Spatial Data Sources appropriate to the distubance are written to files.***						*Disturbance-Automatic .asc files contain values for a single day of simulation.*
107	Disturbance	BURNED_AG_TOTAL_N		N/A	"gN/m2"	N/A	*Total above-ground amount (biomass and detritus) removed from cell*
108	Disturbance	BURNED_ASH_AMOUNT_N		N/A	"gN/m2"	N/A	*Portion of cell's burned ag total that becomes ash (deposited onto cell as N)*
109	Disturbance	BURNED_BIOMASS_AG_STEM_C		N/A	"gC/m2"	N/A	*Amount of above-ground biomass stem removed from cell*
110	Disturbance	BURNED_BIOMASS_LEAF_C		N/A	"gC/m2"	N/A	*Amount of biomass leaf removed from cell*
111	Disturbance	BURNED_DETRITUS_AG_STEM_C		N/A	"gC/m2"	N/A	*Amount of above-ground detritus removed from cell*
112	Disturbance	BURNED_DETRITUS_LEAF_C		N/A	"gC/m2"	N/A	*Amount of detritus leaf removed from cell*
113	Disturbance	BURNED_VOLATILIZED_AMOUNT_N		N/A	"gN/m2"	N/A	*Portion of cell's burned ag total that is volatilized (removed from simulation)*
114	Disturbance	FERTILIZATION_HUMUS_ADDED_N		N/A	"gN/m2"	N/A	*Amount of Humus added to cell.*
115	Disturbance	FERTILIZATION_NH4_ADDED_N		N/A	"gN/m2"	N/A	*Amount of NH4 added to cell.*
116	Disturbance	FERTILIZATION_NO3_ADDED_N		N/A	"gN/m2"	N/A	*Amount of NO3 added to cell*
117	Disturbance	GRAZED_BIOMASS_AG_STEM_C		N/A	"gC/m2"	N/A	*Amount of biomass above-ground stem removed from cell*
118	Disturbance	GRAZED_BIOMASS_LEAF_C		N/A	"gC/m2"	N/A	*Amount of biomass leaf removed from cell*
119	Disturbance	GRAZED_GRAZER_GAIN_TOTAL_N		N/A	"gN/m2"	N/A	*Portion of cell's grazed total used by grazer(s) (removed from simulation)*
120	Disturbance	GRAZED_HUMUS_ADDED_TOTAL_N		N/A	"gN/m2"	N/A	*Amount of cell's grazed total converted transferred to Humus*
121	Disturbance	GRAZED_NH4_ADDED_TOTAL_N		N/A	"gN/m2"	N/A	*Amount of cell's grazed total converted transferred to NH4*
122	Disturbance	GRAZED_TOTAL_N		N/A	"gN/m2"	N/A	*Total above-ground amount (biomass) removed from cell*
123	Disturbance	GRAZED_VOLATILIZED_TOTAL_N		N/A	"gN/m2"	N/A	*Portion of cell's grazed biomass that is volatilized (removed from simulation)*
124	Disturbance	HARVESTED_TOTAL_BIOMASS_AG_STEM_C		N/A	"gC/m2"	N/A	*Amount of above-ground biomass stem removed from cell*
125	Disturbance	HARVESTED_TOTAL_BIOMASS_BG_STEM_C		N/A	"gC/m2"	N/A	*Amount of below-ground biomass stem removed from cell*
126	Disturbance	HARVESTED_TOTAL_BIOMASS_LEAF_C		N/A	"gC/m2"	N/A	*Amount of biomass leaf removed from cell*
127	Disturbance	HARVESTED_TOTAL_BIOMASS_ROOT_C		Always	"gC/m2"	N/A	*Total Amount of above-ground biomass root removed from cell*

End