Integration of VELMA 2.0 with the MOEA Framework for automatic calibration

**OVERVIEW**

The VELMA 2.0 Simulator has been integrated to work with the MOEA Framework for automatic calibration of VELMA output with observed data measurements. Below is a short explanation of how to configure and calibrate VELMA using the MOEA-VELMA calibration tool.

*What is MOEA?*

The MOEA Framework is Java code that facilitates the use of evolutionary algorithms to solve multi-objective optimization problems. See www.moeaframework.org for more details on the MOEA framework. For our integration with Velma, a complete listing of functionality is provided below. First, we will show just a quick example for how to run a MOEA/VELMA integrative calibration.

**EXAMPLE RUN OF MOEA-VELMA CALIBRATOR**

From a Windows command line, to run the MOEA/VELMA calibrator, you will type and run the following command:

java -cp -Xmx<memory> C:\path\to\MOEA-VELMA.jar;C:\path\to\MOEAFramework-2.6\lib\\* org.moeaframework.examples.ga.Velma.runMOEA\_nsgaii RandomSeed C:\path\to\CalibratorConfiguration.csvFile PopulationSize NumberOfProcessorsToUse

The red text specifies required commands regarding the locations of the source code for the MOEA Framework to run. This includes the following:   
1. The path to the MOEA-VELMA.jar file (e.g., MOEA-VELMArev1955.jar). This includes the entire calibrator integration code.   
2. The path to the bin files of the MOEAFramework-2.6. Remember that MOEAFramework must be in a reachable directory for this to run properly.   
3. The path to the "main" class in the .jar source code, as found in e.g., C:\path\to\MOEAFramework-2.6\lib\\*. This is important for the program to know where to begin.

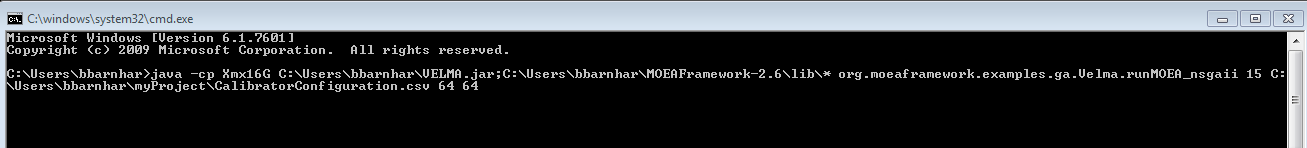
The blue text is information that can be changed by the user depending on your needs. This includes the following:   
1. -Xmx<memory> is the max memory allocation allowed for the run. E.g., -Xmx10G. Note that this can be omitted.   
2. RandomSeed can be any integer, e.g., 10. This number is used for specifying the random seed of the genetic algorithm. Use of the same seed will result in the same initial conditions sent to the genetic algorithm. Different numbers can be used to generate different random initial conditions.   
3. The path to the CalibratorConfiguration.csv file (this will be explained in further detail.)

4. PopulationSize specifies the number of individuals to be used within the genetic algorithm. The more individuals you use, the longer the algorithm will take to test all the model solutions. However, you should consider using at least 16 individuals to ensure genetic diversity of solutions.

5. NumberOfProcessorsUsed: MOEA is parallelized and can run 1 individual for every thread available to it. Previous versions of MOEA utilized all processors/threads available to a machine. However, this led to memory issues and can also cause your computer to be completely dominated by a MOEA run. Therefore, depending on your usage needs, you can set this value to as many processors/threads as you would like. Consider setting this to the max available for your computer at first; then, if you run into memory errors, you may reduce the number of threads used.

This is all you should need to run a MOEA calibration run with VELMA.

Details regarding setting up the CalibratorConfiguration.csv file as well as observed data files are described below.



Details regarding CalibratorConfiguration.csv file

The CalibratorConfiguration.csv file is the primary mechanism to communicate between VELMA2.0 and the MOEA calibration algorithm. This file answers the following questions.

1. What parameters within VELMA do you want to allow to vary (and within what ranges)?

2. What observed and simulated data do you want to include? What data do you want to match with what simulated variables?

3. What objective functions do you want to use to compare simulated and observed data?

4. Where are the locations of all the relevant files?

Accordingly, each of the underlined items are specified within the calibratorConfiguration.csv file.

**An example of the CalibrationConfiguration.csv file can be found at** CalibrationConfigurationExample\_Revised\_10.xlsx

Other examples:

Example\_CalibratorConfiguration\_runoff.csv

Example\_CalibrationConfiguration\_latestComplex\_2017-06-06.csv

**A description of all aspects of the CalibrationConfiguration.csv file are given in a series of pptx files, and they can be found at**

CalibratorConfiguration\_Records\_QuickRef\_7.pptx

This includes more information than is necessary to simply run a MOEA-VELMA calibrator run, but it describes the full API used within Java to link the MOEAFramework with VELMA.

**The VELMA KeyName Glossary can also be very helpful for determining which variables to match to your observed data.**

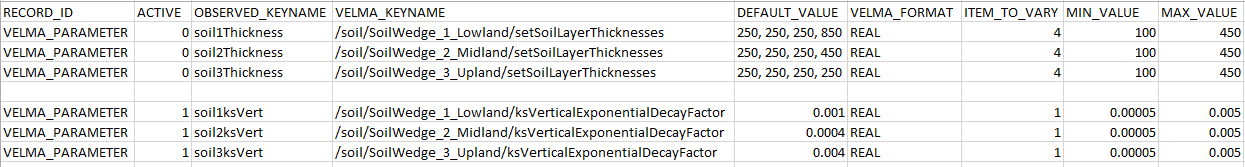
VelmaResult\_KeyName\_Glossary\_4.xlsx

Specifying Parameters

There are three types of parameters that can be sent from the calibrator to VELMA:

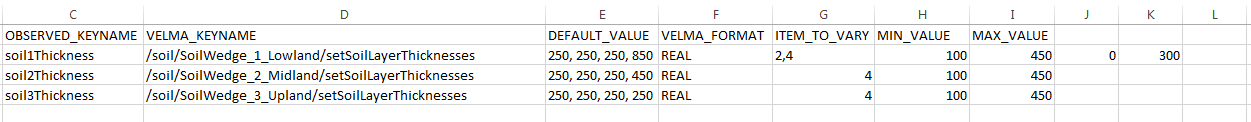
VELMA\_PARAMETER: This is a parameter that is adjusted or ‘calibrated’ by the genetic algorithm used within the MOEA framework. A user specifies the VELMA\_KEYNAME of the parameter and the range between which the genetic algorithm can search for an optimal parameter choice.

Here is an example of VELMA\_PARAMETERs specified within a CalibratorConfiguration.csv file.



For a detailed description of each of the required fields, see the CalibratorConfiguration\_Records\_QuickRef\_.pptx document. There are two things to note here though. The ACTIVE column can be either ‘0’ or ‘1’ and determines whether the parameter is used by the calibrator. Also, the ITEM\_TO\_VARY column is especially important when a particular VELMA\_KEYNAME has more than one VALUE. For example, for the VELMA\_KEYNAME /soil/SoilWedge\_1\_Lowland/setSoilLayerThicknesses, there are 4 values which pertain to the four layers of the soil in this particular VELMA setup. To change only the 4th value (layer), place a 4 in the ITEM\_TO\_VARY column.

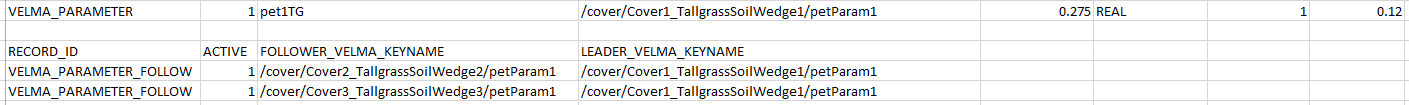
If you want to vary more than one value, then use the following formatting:



Note that this is only a portion of the spreadsheet. Here the 2nd and 4th values (layers) are changed, and the appropriate formatting for including more than one MIN and MAX value ranges.

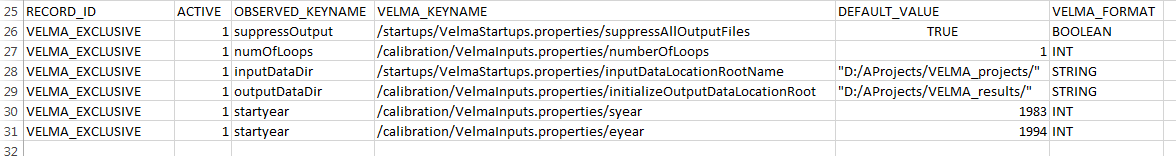
Again, for more details about specifying VELMA\_PARAMETER values in the CalibratorConfiguration.csv file, see examples and the QuickRef document.

VELMA\_PARAMETER\_FOLLOWER: This type of parameter is used to give a parameter used in Velma the same value as given by a VELMA\_PARAMETER by the genetic algorithm using the MOEA calibrator. For example, for a Velma setup with 3 cover types, you may want to calibrate the PET parameter but have that calibrated parameter be the same for all 3 cover types.



Here we want to calibrate petParam1 as a VELMA\_PARAMETER for /cover/Cover1\_TallgrassSoilWedge1/petParam1, but we want to have two other cover type parameters /cover/Cover2\_TallgrassSoilWedge2/petParam1 and /cover/Cover3\_TallgrassSoilWedge2/petParam1 give the same value as that for Cover1. In order to do this, simply give specify the FOLLOWER\_VELMA\_KEYNAME and the LEADER\_VELMA\_KEYNAME as shown in the example.

VELMA\_EXCLUSIVE: This parameter is directly sent to VELMA and is not calibrated—instead, its value remains constant. Here’s an example of a number of different VELMA\_EXCLUSIVE specifications.



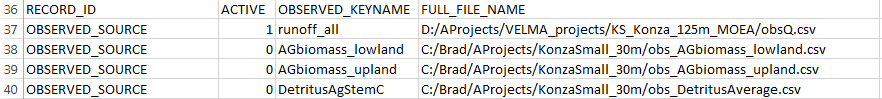
Observed Data Input Requirements

Observed input data files must be specified in a particular format to be read by the MOEA-VELMA calibrator.   
Each observed input data file must exist separately. It must follow a CSV format that contains the Year, Julian Day, and Value. An example is shown below:

|  |  |  |
| --- | --- | --- |
| 1996 | 298 | 0.000105 |
| 1996 | 299 | 0.000138 |
| 1996 | 300 | 0.000162 |
| 1996 | 301 | 0.000147 |
| 1996 | 302 | 0.000195 |
| 1996 | 303 | 0.000509 |
| 1996 | 304 | 0.00021 |

There are no headers in the file, and the file should be saved as a CSV file. Note that observed data are not required to exist for the entire duration of the VELMA run. For example, if you had a VELMA run that went from 1980-2000, then the above file example would be perfectly adequate, and would only compare the simulated values for those time periods.

Here’s an example of specifying observed data for use within the CalibratorConfiguration.csv file.



Explanation of MOEA OUTPUT file

The location of the MOEA output file is specified in the CalibrationConfiguration.csv file. The output file is a CSV file that gives details regarding the objectives used, objective values obtained, and variables used in simulation. The MOEA output file currently outputs all tested individuals. The user is required to use software to determine the best trade-off individual based upon their preference for different objective metrics.

You can find an example of a MOEA output file in the documentation directory of MOEA-VELMA.

The first column specifies that the following columns are objectives. The subsequent columns give the objective names and their resulting values for this particular individual. Each individual is given as a separate row. The MOEA-VELMA algorithm appends to this file as more individuals are evaluated. After the "variables" column, the variables for the given individual are presented. After the variables (moving left to right) if there are VELMA\_PARAMETER\_FOLLOWER or VELMA\_EXCLUSIVE parameters in the run, they will be shown as followers and exclusives, respectively. Finally, a timestamp is printed to show when each individual VELMA run completed.

NOTE THAT MOEA CURRENTLY ONLY MINIMIZES FUNCTIONS. Therefore, if you are using NSE as an objective, remember that the output in the moeaoutput.csv file is the negative of your actual values. For example, if the moeaoutput.csv shows NSE results of -0.7, -0.68, -0.65, etc, these refer to actual NSE values of 0.7, 0.68, 0.65. We are currently working to fix this ‘bug’ but for now, you simply need to be aware of this.

Hard-coded elements of MOEA-VELMA calibration

There are a number of hard-coded portions of the calibration code that links the MOEAFramework to VELMA. In the future, these may be able to be specified by the users; however, as of now they are fixed elements of the source code. If you choose to alter the source code, just remember you will need to re-compile the MOEA-VELMA.jar file in order for the changes to take place.

Hard-coded elements include the following:

Choice of Algorithm: Currently the MOEA-VELMA.jar is set to use the genetic algorithm NSGA-II (Deb et al. 2002). The MOEAFramwork2.6 allows for many other algorithms to be used; however, the current integrated code is set up to use only NSGA-II.

Max Evaluations: 1,000,000. This is essentially infinite, so the algorithm is set to run forever. The user must terminate the calibration algorithm by pressing "Ctrl + C" at the Command Window.

Crossover and Mutation: The Simple Binary Crossover (SBX) rate is 1.0, meaning that 100% of all individuals partake in reproduction. The SBX distribution index is 10.0, which defined how "adventurous" the genetic algorithm is. The Probability of Mutation (PM) rate is set to 1/numberOfVariables and its distribution index is 15.0.

Parallel Implementation: The MOEA-VELMA framework is set to run in parallel and to utilize all available processors of a given machine. Multiple-machine set up is not currently integrated; however, you could start multiple MOEA-VELMA runs on different machines with different RandomSeeds.