Development of an Open-Source Software Package for Watershed Modeling with the Hydrological Simulation Program in Fortran

Lampert, D.J.^{a,*}, Wu, M.^a,

^a*Argonne National Laboratory, 9700 South Cass Avenue, Building 362, Lemont, IL 60439, 1.630.252.2533*

Abstract

The Hydrological Simulation Program in Fortran (HSPF) is used extensively for the assessment of water quantity and water quality issues. Herein, the development of an open-source, cross-platform package for building input files, performing simulations, postprocessing and calibrating HSPF models using the Python Programming Language is presented. The flexible nature of Python opens the door to automated preprocessing and calibration routines, visualization, multiprocessing, and larger-scale model development. The software is applied using Python scripts, which provides a flexible mechanism for learning and applying HSPF. An example application of the software was used to build a calibrated HSPF model the Hunting Creek watershed within the Patuxent River Basin, Maryland, USA, which is the example application distributed with the HSPF calibration software package HSPEXP. A script of a few hundred lines was used to build a calibrated model comparable to HSPEXP in a simulation time of less than two hours.

Keywords:

watershed modelling, Python programming, hydrology, HSPF

1. Introduction

Changes in technology, land use, and rapid population growth place an increasing large burden on both the quantity and quality of water in engi-

*Corresponding author:

Email address: djlampert@gmail.com ()

neered and natural systems. Effective allocation of water resources requires the use of watershed models to simulate the flow of water and the transport of sediment and other associated water quality constituents. The Hydrological Simulation Program in Fortran (HSPF) was developed by the United States Environmental Protection (EPA) in the late 1970s to attempt to fill this need (Singh and Frevert, 2010).

Development of an HSPF model for a watershed requires data on stream flow networks and associated catchment areas, land use, time series of meteorological parameters, and a time series of observed stream flows for calibration. HSPF possesses a flexible, modular design and has been successfully applied to diverse climatic regimes around the world (Singh et al., 1995). HSPF possesses modules for simulating hydrology, temperature, snow, sediment transport, and other quality constituents that can be utilized or ignored as needed. As a result, the model has remained relevant long after its original development (Singh and Frevert, 2010).

HSPF has been jointly maintained by the EPA and the United States Geologic Survey (USGS) up through the current release, Version 12.2 with the EPAs Better Assessment Science Integrating point and Nonpoint Sources (BASINS). The BASINS package provides an integrated Geographic Information Systems (GIS) data system to facilitate the extraction of data, pre-processing of data sets, construction of input files, calibration of model parameters, and visualization of results. Each of these components is essential for the development and application of HSPF models.

While the underlying concept of an integrated preprocessor and modeling platform for BASINS is useful, the BASINS tools are limited. Much of the BASINS graphical user interface (GUI) was written in Visual Basic 6, and thus is not compatible with recent releases of Microsoft Windows based on Visual .NET. Furthermore, the software cannot be used natively on other operating systems (e.g., Unix-like environments). The GUI design of BASINS reduces much of the flexibility inherent to the original modular HSPF design. Attempts to perform simple BASINS tutorials are cumbersome and software crashes are frequent. For these reasons, alternative methodologies are needed for preprocessing, simulating, postprocessing, and calibrating HSPF. This paper discusses the development and application of the software “PyHSPF,” which is a Python package that can be used for preprocessing, postprocessing, and calibrating HSPF models.

The challenges associated with legacy software such as HSPF are a common problem for environmental modelers. Environmental software packages

generally possess unique strengths and weaknesses, and it is often desirable to integrate complementary packages together (Whelan et al., 2014; Voinov and Shugart, 2013). For example, Shrestha et al. (2013) developed an open modeling interface to combine the Soil and Water Assessment Tool (SWAT), a watershed model with sophisticated agricultural runoff and water quality characteristics, and the Storm Water Management Model (SWMM), which possesses a sophisticated representation of river, canal, and sewer hydraulics. Kim et al. (2008) replaced the relatively simplistic groundwater representation in SWAT with the Modular Finite-Difference Flow (MODFLOW) groundwater model to improve SWAT’s baseflow runoff representation.

HSPF is the product of many influential studies that remain as relevant today as when they were originally performed. However, the software in its existing form is difficult to apply to new scenarios, and the format and sources of the relevant input data have changed dramatically since the original code was written. The Python programming language contains many useful data analysis tools that are relevant for hydrologic modeling. The PyHSPF package described in this paper combines the hydrology and water quality capabilities of HSPF with data analysis tools in Python to create a flexible framework for the many tasks needed to develop watershed models.

Running an HSPF simulation requires only two input files (though it is possible to use more). The first is the User Control Input (UCI) file, a plain text file that contains instructions for a simulation. The second is the Watershed Data Management (WDM) file, which is an unformatted, direct-access file that contains time series of input data. HSPF Version 12.2 allows multiple WDM files, each of which can store hundreds of time series (Bicknell et al., 2001). Building WDM files requires access to many of the subroutines in the HSPF library. Traditionally this is accomplished with other software such as USGS ANNIE program, or the WDMUtil program that is distributed with BASINS. HSPF uses a message file for runtime error reporting that can be built using the WDIMEX program distributed with the HSPF source code. Herein we describe a new approach to construct the HSPF input files and retrieve simulation results from HSPF simulations.

Calibration of the hydrology process parameters is an important step in developing an HSPF model. To assist in HSPF hydrology calibration, the USGS developed a system called HSPEXP (Lumb et al., 1994). HSPEXP uses comparisons of observed and simulated flow statistics including total runoff, highest 10% flows, lowest 50% flows, baseflow recession rate, storm flows, and seasonal flows to assist in developing a calibrated HSPF model. Follow-

ing a simulation, HSPEXP provides guidance on what parameters should be increased or decreased and the rationale behind the changes based on the differences in these key statistics. By matching these statistics, the model is expected to demonstrate good consistency with flow-duration curves. HSP-Exp examines the errors in the simulation results and provides guidance on parameter adjustment until errors are less than the predefined criteria. Herein the performance of the new software tools is compared to HSPEXP.

2. Materials and Methods

2.1. HSPF Library Compilation

HSPF Version 12.2 is open source and available online (AQUA TERRA Consultants, 2014). Much of the legacy Fortran 77 code still exists in the current version, including many extraneous utility subroutines designed for pre- and post-processing data. The previous release of HSPF (Version 11) was designed for statically compiling the libraries on Unix-like systems, and the current release contains many artifacts from this approach. The latest release, however, provides no formal make file or detailed instructions to compile the code. The compiled HSPF libraries distributed with BASINS were built with Lahey Fortran, a commercial compiler. Because Fortran 77 does not utilize dynamic memory allocation, the ability to re-compile the program is essential (i.e., for development of larger models).

For these reasons, a thorough characterization of the HSPF 11 and 12.2 source codes was performed to determine the essential files needed to build a single HSPF library. A few minor changes were made to the HSPF source code to remove dependencies introduced by extraneous subroutines and to modify some of the file input and output subroutines. No changes were made to the underlying hydrology and water quality process code. The changes to the source code are bundled into an “HSPF13.zip” archive available with the PyHSPF source code in the “misc” directory. The essential Fortran and parameter files from the various LIB3.0 subdirectories were copied into a single directory named “hspf13” along with the source code changes. Collectively the files in this directory can be used to compile 32-bit or 64-bit path-independent shared libraries of essential HSPF subroutines for both for Windows (dynamic-link library, DLL) and Linux (shared object, SO) operating systems with the open source GCC compiler collection (Version 4.6.2) (Stallman, 2003). The resulting libraries from the compilations were used to run the test files distributed with the HSPF Version 12.2 source code. The

simulation output was consistent with output files in the source distribution. A batch file (for Microsoft Windows) and a shell script (for Unix-like systems) were developed to facilitate the compilation process. The files require only three calls to the GNU compilers using the files in the hspf13 directory. One call is needed to compile the underlying C code into objects, one to compile the underlying Fortran code into objects, and one to link the object files into a library. The batch file and shell script are distributed with PyHSPF in the “misc” directory along with a detailed summary of the HSPF Version 12.2 source code and the changes needed to compile the library.

2.2. Python Extensions to the HSPF Library

The Python Programming Language is a general-purpose, interpreted, high-level language that emphasizes readability (Van Rossum and Drake, 2011). High-level interpreted languages such as Python can be used to integrate together the many tasks involved in building and applying a watershed model. Numeric Python (NumPy) provides a fundamental package for scientific computing with Python including access to multidimensional containers for data. NumPy possesses a Fortran to Python interface generator (F2Py) that can be used to build connections between Python and Fortran (Peterson, 2009).

Python extensions to the HSPF library were developed using a series of Fortran subroutines that act as pointers to the essential HSPF input and output subroutines. The subroutines were placed into a file “hspf_f2py_module.f” with the HSPF source code in the “src/hspf13” directory and include F2PY intent statements that indicate the variables to be used as Python input arguments and the output return expressions. The Python extensions to HSPF were built via an operating system-independent NumPy setup script. The extensions provide access to subroutines that are required to build the WDM files for HSPF. An additional Python script “make_messagefile.py” was written to replace the WDMEX program to build the HSPF message file. This file is distributed with the PyHSPF source code in the “misc” directory.

2.3. PyHSPF Software Description

Building an HSPF model requires physical hydrography data on catchment areas and stream reaches, land use category data, time series of meteorological forcing data, and hydrology process parameters for model land segments. The hydrology process parameters are typically calibrated to a

particular location, which necessitates time series data of stream flows. All of these data must be gathered together and translated into the HSPF input UCI and WDM files. A series of Python classes were developed to interact with the extensions to HSPF to gather input data, build input and output files, run simulations, calibrate hydrology process parameters, postprocess simulation results, and forecast changes in hydrology under different scenarios. The PyHSPF package is divided into four modules containing a variety of classes and methods for these tasks: the core, preprocessing, calibration, and forecasting modules.

PyHSPF utilizes several third party software packages to gather and process data from publicly-available databases and to postprocess HSPF simulation results. Figure 1 shows a summary of the PyHSPF dependencies, code development, and the publicly-available datasets for which data collection tools have been developed for HSPF model construction.

Figure 1: Summary of PyHSPF dependencies, new contributions, and external dataset interaction.

2.4. PyHSPF Core Module

Figure 2 shows a class usage diagram for the PyHSPF core module. A base “HSPFModel” class was developed to collect the information needed for an HSPF simulation, generate HSPF input and output files and call the main HSPF subroutine after input file creation. HSPF simulates hydrology and water quality using the concept of “operations” to represent the physical world. HSPF has three primary operation types: pervious land segments (PERLNDs), impervious land segments (IMPLNDs), and reaches and mixed

reservoirs (RCHRESs). As illustrated in Figure 2, Python classes named “Perlnd,” “Implnd,” and “Rchres” were created to act as containers for the HSPF-specific hydrology and water quality parameters for each of these three operation types.

A “Watershed” class was developed to act as a container for the physical hydrography data for an HSPF model. A watershed in this context is assumed to consist of a series of interconnected subbasins composed of various homogeneous land types, a stream reach, an impervious area. A “Subbasin” class was developed to act as a container for these physical characteristics including flowplane length and slope, stream reach length and slope, land use categories and corresponding areas, and the location (latitude, longitude, and elevation). The stage-discharge relationship for a subbasin reach is required by HSPF and can either be specified directly to the Subbasin class with a table or estimated from the average flow and velocity. The Subbasin instances are created and then supplied to the Watershed class through a Python Dictionary that has subbasin names as keys and instances of the Subbasin class as values. The connections between the stream reaches are specified as a Python Dictionary with keys that are the subbasin names and values that indicate the downstream subbasin reach. After an instance of the Watershed class is constructed, the HSPFModel class can be used to translate the information into Perlnd, Implnd, and Rchres class instances through the “build_watershed” method. It is also possible to use an existing instance of the HSPFModel class to construct a new HSPFModel using the “build_from_existing” method (which is useful for calibration and forecasting). These class concepts are illustrated in Figure 2.

The impact of anthropogenic activities on hydrology and water quality can be simulated in HSPF using “special actions” that are described in the UCI file in the SPEC-ACTIONS block. In HSPF, special actions are used to adjust state variables and parameter values to represent activities such as changes to infiltration parameters or production of detached soil due to row crop cultivation during the course of a simulation. The “SpecialAction” class in Figure 2 was developed to act as a container for special actions and to facilitate the addition of special actions to HSPFModel instances. An appendable list of special actions are distributed with the software.

Time series data are essential to HSPF simulations. Every land segment in HSPF requires time series for precipitation and potential evapotranspiration. To simulate snowmelt and water quality parameters, a number of additional time series must be supplied. The HSPFModel class utilizes Python

Figure 2: PyHSPF Class Usage Diagram for the PyHSPF Core Module. The diagram illustrates how the various classes interact to build a baseline HSPF model, construct the HSPF input and output files, and run a simulation to populate the output WDM file with time series results.

Dictionaries to store the time series data. The time series dictionaries assume unique identifiers are provided as keys to identify time series that are composed of a start date, time step in minutes, and a Python List of data. Additional functions were created to assign time series to either all the land segments in the watershed, specific subbasins, or specific land use categories as shown in Figure 2.

The time series data are provided to HSPF and generated from a simulation using WDM files. Because of the unformatted, direct-access nature of the WDM file structure, many subroutine calls are needed to actually interact with WDM files. To simplify the interaction with WDM files, a “WDMUtil” class based very loosely on both the BASINS WDMUtil program another package (Cera, 2013) was developed. Time series in the WDM files are identified with a unique dataset number and possess a series of at-

tribute identifiers that are specified in the “msg” subdirectory of LIB3.0 in the HSPF Version 12.2 source code. Some of the important attributes are the time step unit (minutes, hours, or days), the number of time step units per time step, the time series type, and the constituent identifier. A full listing of the attributes were read into a Python dictionary that is stored with the core module in the “attributes” file that is used by the WDMUtil class to facilitate addition of these attributes to WDM time series datasets. The WDMUtil class contains Python methods to open and close WDM files, get unique dataset numbers for a time series in the file, create and delete datasets, add data to datasets, and to create and get attributes of datasets in the WDM file. Based on the time series provided to the HSPFModel class and the associated assignments made to it, the HSPFModel class works with the WDMUtil class to automatically generate an input WDM file through a “build_wdminfile” method.

The final pieces of information required to run an HSPF simulation are the desired model output and the run period. The HSPFModel class possesses a “build_ucifile” method to build the UCI file that takes a Python List of “targets” and the start and end date as inputs. The targets are translated into the EXT TARGETS (external targets) block in the UCI file. To use this block in HSPF, the datasets must be created and the assignments made in a WDM file before the simulation. This functionality is performed automatically with the HSPFModel class for a number of different desired output variables including runoff components, reach flow rates, groundwater recharge, simulated evapotranspiration, and state variables for the various operations. The WDM output file is built automatically with the UCI file to accommodate the desired simulation output. After the input files have been generated, the HSPFModel class can call the main HSPF subroutine through the Python library extensions using the “run” method. The HSPFModel class can then be stored for additional simulation information later using the built-in Python “pickle” module.

While there are still many steps needed to utilize the HSPFModel class, the class structure illustrated in Figure 2 enables direct integration of source data from different databases into HSPF models. The Python class structure simplifies modification of model parameter values and simulation assumptions needed for calibration and scenario development. Existing software typically utilizes cumbersome direct modification of UCI files. The PyHSPF source code contains a number of scripts to highlight usage in the “examples” directory. The first example illustrates how to supply the data needed to run

HSPF as shown in the class usage diagram in Figure 2.

2.5. *PyHSPF Preprocessing Module*

The “preprocessing” module contains a number of classes that interact with the online databases shown in Figure 1 through Python’s built-in “urllib” module and other essential utilities. There are examples provided with the software to illustrate the functionality of these tools. The input data supplied by these classes can then be integrated directly into applications with the HSPFModel class to facilitate rapid model construction in new watersheds. The preprocessing classes rely on several third party Python modules including the Open Source Geospatial Data Abstraction Library (GDAL) (Warmerdam, 2008), the Python Shapefile Library, PyShp (Lawhead, 2014), and the Python Imaging Library, Pillow (Clark, 2014).

Watersheds in the United States are classified using a Hydrologic Unit Code (HUC) that varies from two to fourteen digits (HUC2 to HUC14) depending on the size of the catchment basins. The National Hydrography Dataset Plus (NHDPlus) provides high resolution hydrography data for the United States based on the HUC system. Streams are divided into cataloging units with both a unique 14-digit HUC and a unique common identifier (comid) feature attribute. The source data are publically available on the World Wide Web and are distributed for various 2-digit HUCs (also called Vector Processing Units or VPUs) with NHDPlus Version 2 (McKay et al., 2012). The source data include the National Elevation Dataset (NED) which is a 30-meter raster elevation grid, flowlines and catchment shapefiles, and databases of feature attributes. The class usage diagram shown in Figure 3 illustrates how data from the NHDPlus Version 2 dataset can be communicated into an instance of the PyHSPF Watershed class.

The NHDPlusExtractor class is designed to extract relevant HSPF model data for a HUC8 from the online NHDPlus database. The NHDPlusExtractor automatically extracts and decompresses VPU-level source datasets (which can exceed several gigabytes in size) from the World Wide Web using the Python’s urllib module and the 7zip program (Pavlov, 2007). Once the source files exist locally, the NHDPlusExtractor uses the comids and HUC14 feature attributes to extract data for a HUC8. Each VPU has on the order of one hundred HUC8s. The NHDPlus data extracted include flowline and catchment shapefiles, the NED raster and several other databases. These databases contain hydrography data including the elevation and slope of each reach (elevslope), annual average flow and velocity estimates from

the Enhanced Unit Runoff Method (EROM) (McKay et al., 2012), and a number of flowline value added attributes (VAAs) including the hydrological sequence (hydroseq) feature attribute that can be used to identify the upstream and downstream reach sequence. A “Flowline” class was developed to act as a data container for the important flowline attributes for HSPF model development in the various NHDPlus files as shown in Figure 3.

Figure 3: NHDPlusExtractor and NHDPlusDelineator class usage diagram. The NHDPlusExtractor class downloads raw NHDPlus files and extracts HSPF model data for an 8-digit HUC. The NHDPlusDelineator class then translates the extracted data files into an instance of the PyHSPF Watershed class.

The USGS maintains a network of flow gauging stations that are categorized by a unique site identification number. An “NHDPlusDelineator” class was created to delineate the watershed for a gauging station based on the NHDPlus data as shown in Figure 3. The class takes the location of

the HUC8-extracted NHDPlus files, the site identifier of gauging station, and land use data as inputs and translates the raw NHDPlus data to an instance of the PyHSPF Watershed class. The NHDPlusDelineator generates an additional shapefile for the watershed boundary and several figures illustrating the watershed subbasins and flowlines and the mass linkages between the subbasins. The generated Watershed class instance can then be used to construct an HSPFModel.

2.6. *PyHSPF Calibration Module*

Several classes were developed to assist in the calibration process based on different assumptions about the datasets used to build the baseline HSPF model and the end goal for the model (sediment transport, water quality, etc.). An “AutoCalibrator” class was developed to perform a calibration by adjusting the HSPF PERLND hydrology module (PWATER) parameters. A class usage diagram is shown for the AutoCalibrator class in Figure 4. The AutoCalibrator requires a baseline HSPFModel class, a list of input PWATER calibration parameters, and the output parameter to be optimized. Once this information has been supplied, the AutoCalibrator possesses an “autocalibrate” method that attempts to optimize the output parameter by perturbing the input parameters until an optimization is achieved.

To retain heterogeneity in land segment hydrology, the parameters are adjusted globally relative to the default values. For example, the nominal lower zone storage (LZSN) is a PWATER parameter related to the rooting depth of the vegetation. To retain differences between between land segments with different vegetation, the LZSN parameters for the model are all adjusted by the same fraction relative to this default. The adjustments relative to the defaults are saved and then perturbed iteratively until the calibration parameter is optimized.

The adjustments and the impact on the simulation parameters are performed by creating a copy of the baseline HSPFModel, adjusting all the calibration parameters, performing a simulation, and then postprocessing the simulation results to determine the impact on the calibration parameter. The PyHSPF core module contains a “Postprocessor” class to analyze model performance and create charts of simulation output. The Postprocessor class utilizes the Scientific Python Library (SciPy) that contains mathematical algorithms and convenience functions for Python (Jones et al., 2001) and Matplotlib, an extensive mathematics plotting library for Python (Hunter, 2007). The Postprocessor class utilizes the underlying WDMUtil class to

interact with the input and output WDM files as well watershed information from the HSPFModel class to create model output as shown in Figure 4. The Postprocessor class creates graphs, calculates mass balances on the various hydrology components, analyzes various components of runoff (surface runoff, interflow, and baseflow), and computes statistics including the correlation coefficients for daily and monthly flows and logarithm of flows.

Figure 4: Calibration Class Usage Diagram. The baseline HSPFModel is copied to a new instance, then the AutoCalibrator adjusts the calibration parameters to maximize the optimization parameter. The Postprocessor aggregates simulation results using the WDMUtil class for the baseline and adjusted cases and provides the simulation results to the AutoCalibrator.

Figure 5 shows the logic diagram associated with the autocalibrate method. The method is based on a path of steepest ascent approach to achieve a global maximum on the optimization parameter. The baseline HSPFModel and PWATER parameters are supplied to the AutoCalibrator before the method is called. Before the optimization commences, the initial values of the state variables are adjusted using a 1-month warm up simulation. The beginning state of the model is set to the state at the end of this simulation. A baseline simulation is then performed to assess the current value of the optimization parameter. Each calibration parameter is then perturbed slightly followed by a simulation and postprocessing to assess the impact on the calibration parameter. If the perturbation improves model performance

[illegible]

14

3. Model Evaluation

An example script was written to evaluate the classes and algorithms described in the preceding text. The script illustrates how to build an HSPF model consistent with the example provided with HSPExp for Hunting Creek in the Patuxent, Maryland, USA Watershed. The example uses hydrography data from NHDPlus as diagrammed in Figure 3 with land use and time series data from the HSPExp example (for consistency). The hydrology process parameters were calibrated using the PyHSPF AutoCalibrator as shown in the usage diagram in Figure 4. The example script is provided with the software in the “examples” directory in the “hunting.py” file.

An independent calibration was done on the UCI file provided with the HSPExp software for comparison. The HSPExp calibration was accomplished by starting with the default values for the hydrology parameters in the Hunting Creek example. The HSPExp guidance was then followed until meeting the default error criteria. The UCI file for the HSPExp-calibrated model is also available with the software in the “examples” subdirectory along with a Python script “hspexp.py” that runs a simulation and plots the results. The hydrology process parameters used for HSPExp calibration are the nominal lower zone storage (LZSN), the lower-zone evapotranspiration parameter (LZETP), the upper zone nominal storage capacity (UZSN), the interflow inflow parameter (INTFW), the infiltration capacity (INFILT), the active groundwater recession rate (AGWRC), and the interflow recession constant (IRC).

3.1. Watershed Description

The HSPExp example simulates three river reaches and associated sub-basins. The example utilizes Imperial units (HSPF simulations can be performed in either Imperial or metric units). The first two reaches (left and right branches) flow together into the third (main) reach where hourly flow data were monitored. The source of the hydrography or land use data were not provided. The time series data for precipitation, evaporation, and stream flows were taken from stations at Mechanicsville, MD, Beltsville, MD, and from USGS gage site number 01594670 (Hunting Creek, MD), respectively. A pan evaporation coefficient of 0.75 was provided in the Hunting Creek UCI file. To account for differences in subbasins, the land use data from the example were aggregated together and then distributed equally using catchment areas from the NHDPlus data. The land use categories and associated areas

for the watershed were 3.14 km² pasture/grass, 2.60 km² agriculture, 17.92 km² forest, and 0.48 km² impervious. The NHDPlus data were extracted for the Patuxent River Basin, which is HUC 02060006. The supporting information shows a plot of the NHDPlus data for the Patuxent River.

3.2. *HSPEXP Calibration*

The HSPEXP software utilizes a terminal interface to interact with HSPF model simulations. The software was installed on a Windows XP Virtual Machine as it is not compatible with newer operating systems. HSPEXP provides an interface to the main HSPF program and can be used to modified the PWATER hydrology parameters for the various land segments. The parameters were adjusted from default values for a simulation, and then the advice provided by the HSPEXP software was followed until the calibration criteria achieved. More details are provided in the supporting information. The results were postprocessed to compute the additional statistics for comparison with PyHSPF.

3.3. *PyHSPF Calibration*

The Nash-Sutcliffe model efficiency (NS) (Nash and Sutcliffe, 1970) for daily flows provides an indication of the predictive power of a watershed model relative to the mean value of the observations assuming an equal absolute error between observed and simulated flows. The NS can also be computed for the logarithm of daily flows. This approach assumes an equal relative error between simulated and observed values. For estimating stream flow the NS for the logarithm of flows is more indicative of seasonal trends while the absolute flow is more indicative of storm flows. Because of the continuous time nature of HSPF, it is important to try to capture both of these hydrologic regimes. To achieve this objective, the calibration was designed to maximum the product of the NS for log and linear flow values to provide an approximately equal weighting to both the seasonal variability and storm event components of the hydrograph. The PyHSPF Postprocessor class can compute both of these statistics and the AutoCalibrator can use this value as the optimization parameter as diagrammed in Figure 4. The same hydrology parameters used by HSPEXP (LZSN, LZETP, UZSN, INTFW, INFILT, AGWRC, and IRC) were used in the PyHSPF autocalibration.

3.4. Results

The preprocessing steps described by the diagram in Figure 3 produce an image of the extracted NHDPlus data used to build a watershed for the gage station at Hunting Creek. The results are shown in Figure 6. The widths of the streams are proportional to estimated cross sectional area based on the EROM model estimates in NHDPlus. The extraction and delineation steps take on the order of an hour to run depending on the internet connection speed, primarily to download the large data files from the World Wide Web.

Figure 6: Hunting Creek, MD watershed catchments, flowlines and NED elevation raster from NHDPlus dataset.

Figure 7 shows the calibration results and highlights some of the features of the new software. The optimization routine took less than 30 minutes to complete for this modeling exercise on a personal computer with four IntelTM Core i5 processors and 32 megabytes of memory. The storm event from June 6, 1989 through June 10, 1989 was not captured by either model likely due

to inconsistencies between the input precipitation and observed flow time series. Several low flow periods during winter months were overestimated likely due to an absence of snow and ice simulation. HSPF possesses modules for snow but they were not used for consistency with the HSPEXP example. Despite these two issues, the autocalibration method produced a good NS value for both daily flows (0.728) and log-daily flows (0.721). Singh et al. (2005) reported a NS for daily flows of 0.69 for a 15-year HSPF simulation, while Kim et al. (2007) reported a daily flow-NS of 0.35 using an another autocalibration method. The value achieved with the method used here is higher than these other studies. The PyHSPF software facilitates extraction and postprocessing of the time series for the various runoff components and visualization of the runoff data. These features may be helpful in future HSPF modeling efforts.

Because HSPF is a continuous, long-term model, it is important that the model capture the cumulative flow distribution or flow-duration curve. With the exception of the near-zero flows, the simulations were close to the observed flow-duration curve as shown in Figure 7. The software enables HSPF users to perform a simulation and immediately visualize the simulated flow distribution versus the observed distribution. The supporting information shows the division of flows amongst the three runoff components for each of the storm events during the simulation. These graphs can be generated immediately following a simulation and may be useful to HSPF modelers interested in capturing particular storm flow events.

A comparison of the HSPEXP calibration statistics between the HSPEXP and PyHSPF calibrations is shown in Table 1, while a comparison of some other important statistics is shown in Table 2. The PyHSPF result was consistently low for the various statistics whereas HSPEXP over and under predicts the observations. The PyHSPF model explains the variability in the data better than the HSPEXP approach as evidenced by consistently higher statistics. The discrepancy in the calibration between the models appears to be related to a cancellation of over prediction and under prediction errors in HSPEXP. Some flows were greatly over predicted while some were greatly under predicted. It is also possible that the adjusted evaporation data were consistently higher than actual potential evapotranspiration.

HSPF has remained relevant long after its inception because of its flexible, modular design and history of successful applications in different regions throughout the world. The learning curve associated with HSPF is high in part because of the Fortran 77 punch-card nature of the UCI file and

Figure 7: Hunting Creek HSPF model calibrated hydrograph (top), parity plot (lower left), and flow-duration curves (lower right).

unformatted, direct access format of the WDM file. While enveloping the Fortran with a GUI is useful to experienced HSPF users, a more transparent approach such as the one herein could provide a useful learning tool for new

Table 1: Errors in HSPEXP Parameter Results

Parameter	PyHSPF	HSPEXP	HSPEXP Criteria
Total runoff	-9.8%	8.4%	10%
Highest 10% flows	-11.3%	-2.8%	15%
Lowest 50% flows	-6.2%	2.0%	10%
Seasonal	-12.6%	14.7%	20%
Baseflow recession rate	0.002	0.023	0.03
Total storm volume	-16.3%	22.0%	20%
Storm peak average	-28.9%	1.7%	50%

Table 2: PyHSPF Calibration Statistics

Parameter	PyHSPF	HSPEXP
Daily flow r^2	0.738	0.701
Daily log-flow r^2	0.726	0.654
Monthly flow r^2	0.866	0.870
Monthly log-flow r^2	0.871	0.794
Daily flow NS	0.728	0.616
Daily log-flow NS	0.721	0.611
Monthly flow NS	0.814	0.825
Monthly log-flow NS	0.863	0.772

HSPF users. PyHSPF could be applied in University settings as a way to familiarize students with both Python and HSPF. Furthermore, a GUI-based approach for HSPF is difficult to adjust for applications outside of the original intent of the developers and distorts the model concept of the real world. For these reasons an open-source platform provides a useful alternative to existing GUI-based HSPF platforms.

4. Summary

A new software package has been developed that integrates a number of open-source tools together with a set of modules and classes to perform HSPF simulations. The software allows hydrologic modelers to write short scripts in the Python Programming Language to integrate data into HSPF

models, perform simulations, and postprocess simulation results. Existing HSPF software packages are not completely open source and have limited postprocessing and batch computation tools. The new software contains a number of preprocessing classes to automate the extraction of data directly from several publicly-available datasets into HSPF models. This feature shortens the process of model construction and may be applied to other watersheds in the United States. Alternatively, the new software could be coupled with existing Python data extraction tools (Mitasova et al., 1995; Jennings, 2011) to facilitate HSPF model development with new datasets in other parts of the world. The software includes a class that automatically calibrates the hydrology process parameters from a base model that achieved good performance relative to existing HSPF software.

5. Acknowledgement

The authors wish to thank the United States Department of Energy (DOE) Energy Efficiency & Renewable Energy (EERE) Postdoctoral Research Awards under the EERE Bioenergy Technologies Office administered by the Oak Ridge Institute for Science and Education (ORISE) for funding this research. ORISE is managed by Oak Ridge Associated Universities (ORAU) under DOE contract number DE-AC05-06OR23100.

6. Software Availability

PyHSPF is available on the World Wide Web at:
<https://github.com/djlapert/PyHSPF>

References

- AQUA TERRA Consultants, 2014. Hydrological simulation program in fortran source code. <http://hspf.com/ftp/hspf>. Accessed: 2014-08-26.
- Bicknell, B.R., Imhoff, J.C., Kittle Jr, J.L., Jobes, T.H., Donigan Jr, A.S., Johanson, R., 2001. Hydrological simulation program—fortran: Hspf version 12 user’s manual. AQUA TERRA Consultants .
- Cera, T., 2013. Wdmtoolbox: Read and write watershed data management (wdm) files. URL: <https://pypi.python.org/pypi/wdmtoolbox>.

- Clark, A., 2014. Pillow. URL: <http://pillow.readthedocs.org/en/latest>.
- Hunter, J.D., 2007. Matplotlib: A 2d graphics environment. Computing in Science & Engineering , 90–95.
- Jennings, N., 2011. A Python Primer for ArcGIS®. CreateSpace.
- Jones, E., Oliphant, T., Peterson, P., 2001. Scipy: Open source scientific tools for python. <http://www.scipy.org/> .
- Kim, N.W., Chung, I.M., Won, Y.S., Arnold, J.G., 2008. Development and application of the integrated swat–modflow model. Journal of hydrology 356, 1–16.
- Kim, S.M., Benham, B.L., Brannan, K.M., Zeckoski, R.W., Doherty, J., 2007. Comparison of hydrologic calibration of hspf using automatic and manual methods. Water resources research 43.
- Lawhead, J., 2014. Python shapefile library. URL: <https://pypi.python.org/pypi/pyshp>.
- Lumb, A.M., McCammon, R.B., Kittle, J.L., 1994. Users Manual for an Expert System (HSPEXP) for Calibration of the Hydrological Simulation Program–Fortran. US Geological Survey Reston, VA.
- McKay, L., Bondelid, T., Rea, A., Johnston, C., Moore, R., Dewald, T., 2012. Nhdplus version 2: User guide. URL: <http://www.horizon-systems.com/NHDPlus/index.php>.
- Mitasova, H., Mitas, L., Brown, W.M., Gerdes, D.P., Kosinovsky, I., Baker, T., 1995. Modelling spatially and temporally distributed phenomena: new methods and tools for grass gis. International Journal of Geographical Information Systems 9, 433–446.
- Nash, J., Sutcliffe, J., 1970. River flow forecasting through conceptual models part ia discussion of principles. Journal of hydrology 10, 282–290.
- Pavlov, I., 2007. 7zip. URL <http://www> .
- Peterson, P., 2009. F2py: a tool for connecting fortran and python programs. International Journal of Computational Science and Engineering 4, 296–305.

- Shrestha, N.K., Leta, O.T., De Fraine, B., Van Griensven, A., Bauwens, W., 2013. Openmi-based integrated sediment transport modelling of the river zenne, belgium. *Environmental Modelling & Software* 47, 193–206.
- Singh, J., Knapp, H.V., Arnold, J., Demissie, M., 2005. Hydrological modeling of the iroquois river watershed using hspf and swat1.
- Singh, V.P., Frevert, D.K., 2010. Watershed models. CRC Press.
- Singh, V.P., et al., 1995. Computer models of watershed hydrology. Water Resources Publications.
- Stallman, R.M., 2003. Using gcc: The gnu compiler collection reference manual for gcc 3.3. 1. Free Software Foundation .
- Van Rossum, G., Drake, F.L., 2011. An introduction to Python. Network Theory Ltd.
- Voinov, A., Shugart, H.H., 2013. integronsters, integral and integrated modeling. *Environmental Modelling & Software* 39, 149–158.
- Warmerdam, F., 2008. The geospatial data abstraction library, in: Open Source Approaches in Spatial Data Handling. Springer, pp. 87–104.
- Whelan, G., Kim, K., Pelton, M.A., Castleton, K.J., Laniak, G.F., Wolfe, K., Parmar, R., Babendreier, J., Galvin, M., 2014. Design of a component-based integrated environmental modeling framework. *Environmental Modelling & Software* 55, 1–24.