Visualizing and Analyzing ADCP and CFD Data Using Python and Other Open Source Tools

M. G. Denno, P.E.[1] and G. S. Lemay, P.E.[2]

[1]Gomez and Sullivan Engineers, 399 Albany Shaker Road, Suite 203, Loudonville, NY 12221; PH (518) 407-0050; email:mdenno@gomezandsullivan.com
[2] Gomez and Sullivan Engineers, PO Box 2179, 41 Liberty Hill Road, Building 1, Henniker, NH 03242; PH (603) 428-4960; email:glemay@gomezandsullivan.com

ABSTRACT

An Acoustic Doppler Current Profiler (ADCP) can be linked with a Global Positioning System (GPS) to collect spatially-referenced bathymetric and three-dimensional velocity profile datasets. These datasets provide a detailed view of natural and engineered systems' hydraulics that can be used to develop and validate three-dimensional computational fluid dynamics (CFD) hydraulic models. ADCPs and CFD models generate large quantities of data. Use of a scripting language such as Python to process and manipulate the datasets results in an efficient, transparent and reproducible method. Having both the observed (ADCP) and simulated (CFD) data in a common open format allows the datasets to be manipulated and compared programmatically and makes for easy side-by-side visualizations.

The study team used ADCP datasets to validate four CFD models on the Connecticut River in Massachusetts, and this paper presents the key role that open source tools (including Python, the Geospatial Data Abstraction Library, NumPy, and ParaView, among others) played in the process. The study process included: 1) designing an appropriate field data collection scheme; 2) collecting the field data in a structured and consistent manner; 3) processing the ADCP and CFD model data and manipulating to/from model coordinate systems; 4) using open source tools to convert the data to a common format; and 5) to analyze the results and inform fish passage management decisions on the Connecticut River.

INTRODUCTION

Recent technology and computing advances have greatly enhanced the data collection and modeling of open-channel hydraulics. Velocity data collection has greatly improved with the increased use of Acoustic Doppler Current Profilers (ADCPs), which when linked with a Real-Time Kinematic Global Positioning System (RTK-GPS) can provide a precise and high-resolution georeferenced point cloud (X, Y, Z) of three-dimensional (3D) velocity data (u, v, w) throughout the water column, as well as allow for quick and simple cross-sectional total flow measurements. Collecting ADCP data, like most hydraulic field data, can be difficult and expensive to collect for more than a small number of flows under steady or near-steady conditions. Computational Fluid Dynamic (CFD) modeling uses numerical methods to calculate hydraulic data (including water velocities) at high-resolution spacing. CFD models allow many flow conditions to be simulated, and with the advent of increasingly fast and relatively inexpensive computers the ability to model larger

systems in greater detail is continually improving. Comparing CFD model results with field observations is critical, however, to evaluate a model's strengths and weaknesses and determine whether a model is appropriate for meeting study objectives. Using georeferenced 3D velocity observations from a combined RTK-GPS/ADCP system to assess CFD model outputs is a sensible, effective, and detailed method to assess CFD model accuracy when combined with other traditional field data (e.g., water surface elevations, flow distributions, etc.).

ADCP data and CFD outputs provide a detailed view of natural (e.g., streams and rivers) and engineered (e.g., canals, powerhouse intakes) open-channel systems' hydraulics. The vast amount of hydraulic data available allows engineers, environmental scientists, fisheries biologists, and others to visualize and study open-channel hydraulics at dams, road crossings, and other primarily-manmade structures to help inform design decisions relative to fish passage, dam safety, bridge design, and numerous other purposes. This wealth of data, however, also brings about many challenges in understanding and using it to make informed decisions.

Fortunately, there are many available open-source tools to help engineers and scientists process, manipulate, compare, and visualize datasets. Open source software is software with source code available for anyone to inspect, modify, enhance, and generally use the software for any purpose (What is open source software, 2017).

The goal of our study was to develop and validate CFD models at four study areas at the Turners Falls Hydroelectric Project (Project) on the Connecticut River in Massachusetts and then conduct CFD model production runs for various flow scenarios to inform upstream and downstream fish passage management decisions. This paper discusses how several open source tools, including Python, the Geospatial Data Abstraction Library (GDAL), NumPy, and ParaView, among others, were used to process, compare, visualize, and assess ADCP and CFD hydraulic data to meet the study goals and objectives.

DATA COLLECTION

Gomez and Sullivan was tasked with developing and validating CFD models at four locations around the Project (Figure 1). The four study areas included models at the canal/forebay/intake for each of the Project's two powerhouses (Station No. 1 and Cabot Station) and models of the Connecticut River near the Project's two fishways (the Spillway fishway, located at the Turners Falls Dam spillway, and Cabot fishway, located adjacent to Cabot Station). Since this paper is focused on the process of using open-source tools to visualize and analyze the results, and the process was similar for each study area, we have chosen to primarily focus on the process for one study area (Station No. 1) in detail.

Figure 2 shows the field data that was collected at Station No. 1, including the ADCP transects which are shown in light blue. The ADCP data was collected using a Sontek RiverSurveyor M9 ADCP unit linked to a Leica RTK-GPS unit. The collected ADCP data is natively saved as proprietary binary files (*.riv). Fortunately, Sontek's

RiverSurveyor software can export the raw location (x, y, z) and velocity vector (u, v, w) data as a series of text (*.txt) and MATLAB (*.mat) files. The ADCP data were then processed using python to convert the data files into Visualization Toolkit (VTK) files that can be viewed in ParaView (Figure 3).

MODEL RESULTS AND COMPARISONS

The CFD model for Station No. 1 was developed using Flow-3D, which has built-in capabilities to review model results. This viewer is good for quickly visualizing the results and generating some model outputs, however it lacks the capabilities of a robust visualization software. Flow-3D can accept a text file (called a neutral file) containing coordinates (x, y, z) as input and it will output a text file that contains the requested model output (u, v, w, fluid fraction, etc.). Leveraging this functionality, and using Python, we created an outfile that contains the same coordinate values (x, y, z) as the ADCP data and extracted the CFD model results (u, v, w) at the same locations within the model domain as where the field data was collected (Figure 3).

After CFD development, the model was validated against real world observations to assess performance. The validation was done at varying levels of detail, ranging from comparing water surface elevations at a few points to comparing depths and velocities throughout the domain. We converted the simulated CFD model results (which have the same coordinate positions within the model domain) to the same common VTK format as the ADCP data. This allowed us to easily compare the observed and simulated velocities within the domain and verify the model's accuracy. Having the data in a common format also allowed us to analyze the data and create plots to visualize the differences between the observed and simulated values (Figure 4).

Figure 5 and Figure 6 are examples of how detailed hydraulic data for the CFD models can be processed and visualized using open source technology to help make better informed decisions. We made extensive use of the Python's NumPy (van der Walt et al. 2011), Matplotlib (Hunter 2007), and Pandas (McKinney 2010) packages.

REFERENCES

John D. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

ParaView and VTK (Kitware Inc.). Retrieved from http://www.paraview.org/

Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org.

RiverSurveyor S5/M9 System Manual. Sontek. 2016.

Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

What is open source software? (2017, March 29). Retrieved from: https://opensource.com/resources/what-open-source

FIGURES

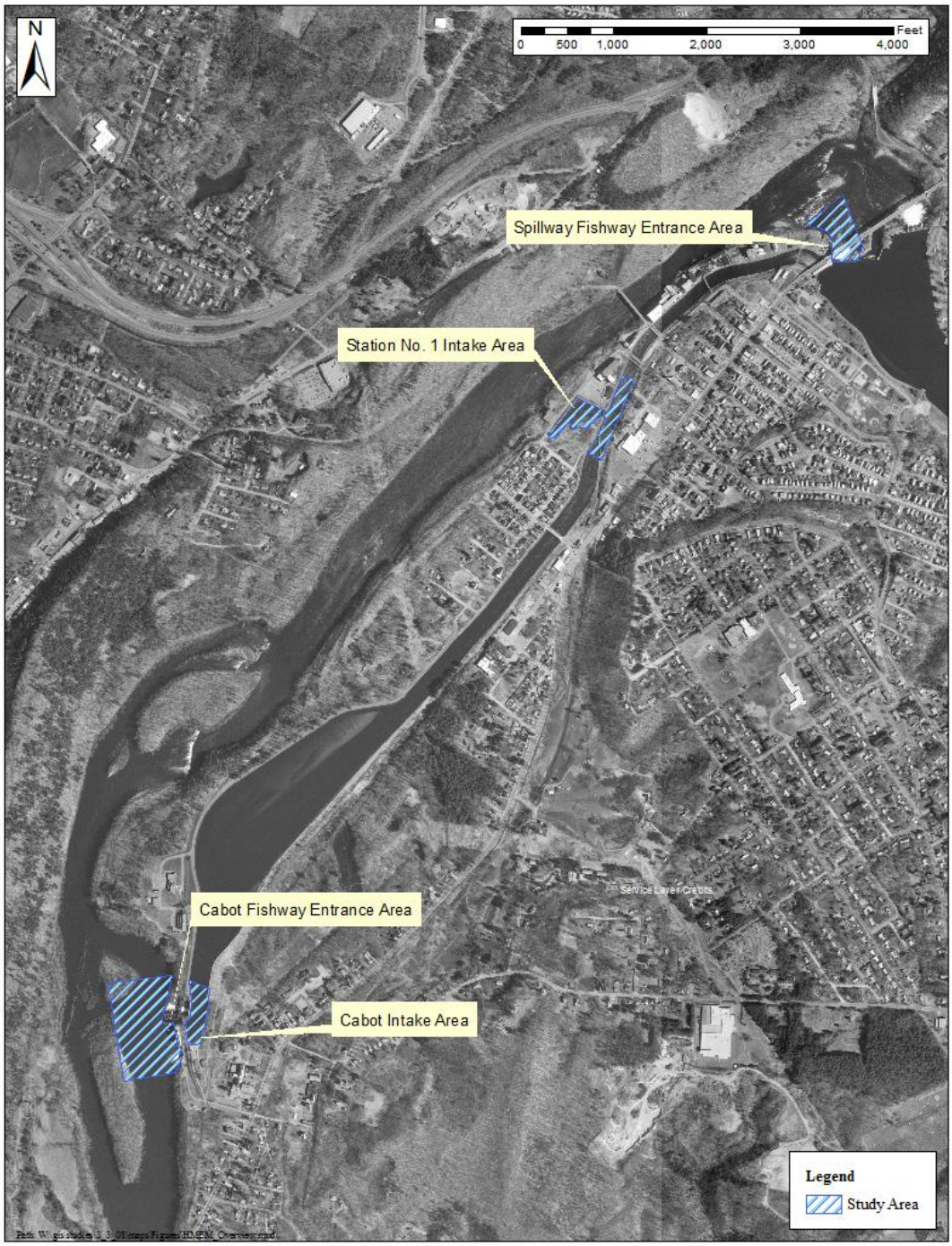

Figure 1: Overview of the four CFD model areas

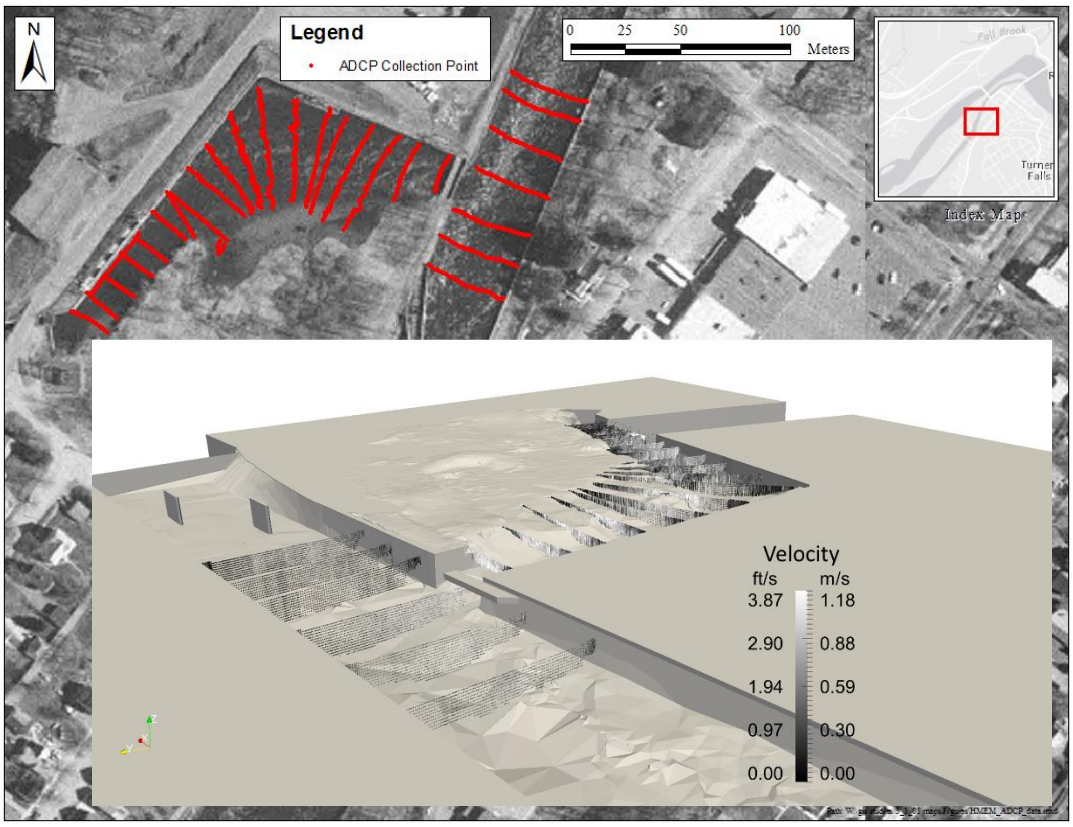

Figure 2: Station No. 1 field data collection. Inset: ADCP data converted to VTK files and visualized in ParaView.

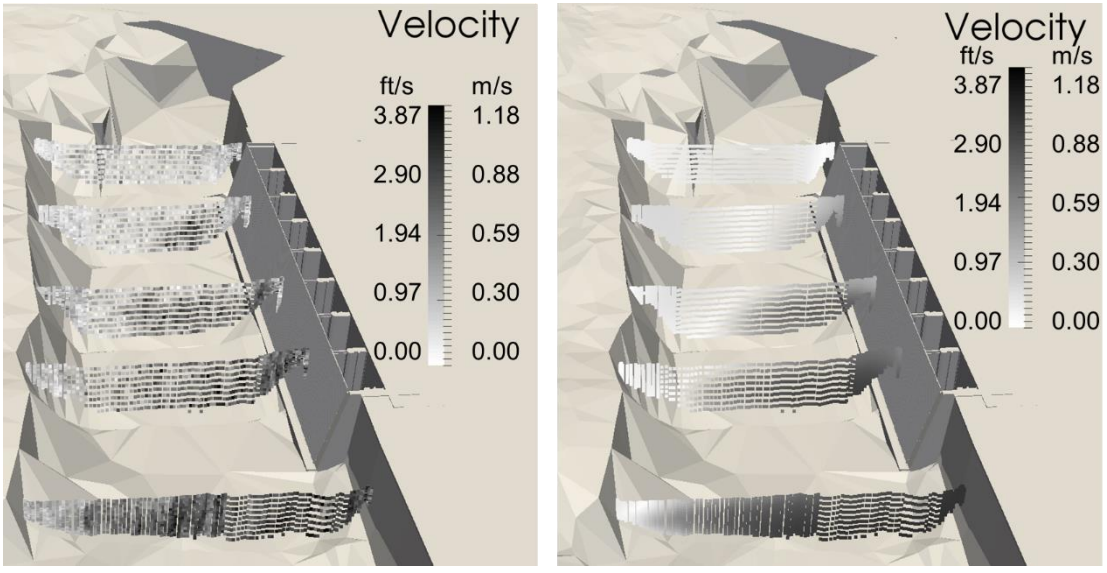

Figure 3: Fifty Shades of Grey – comparing ADCP and CFD outputs in the Station No. 1 Forebay. ADCP (A) and Flow 3D data (B) converted to VTK files and visualized in ParaView.

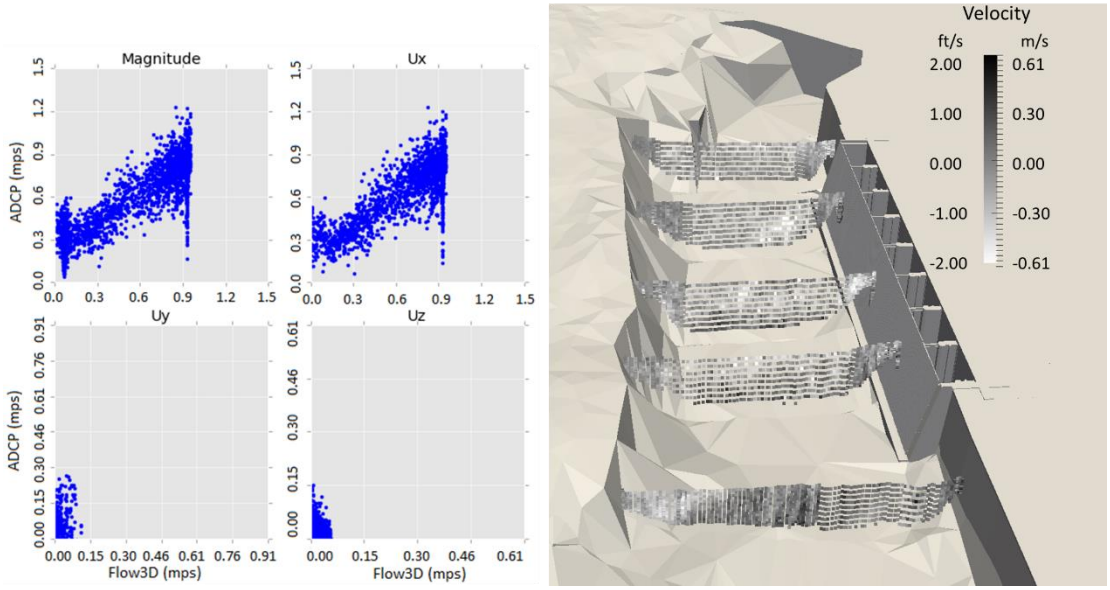

Figure 4: Observed vs simulated velocities: A) component velocity comparison, B) Velocity magnitude difference. Visualized using Matplotlib and ParaView, respectively.

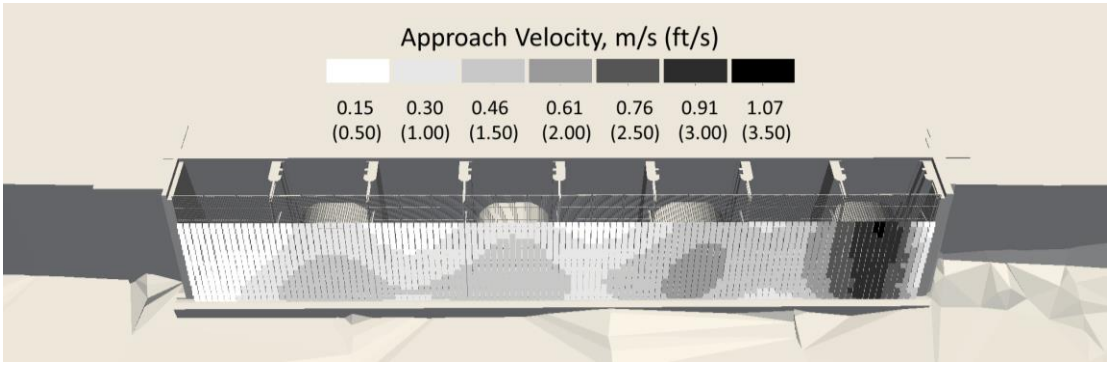

Figure 5: Example of approach velocity plot in front of intake racks, calculated using Python and NumPy and visualized in ParaView

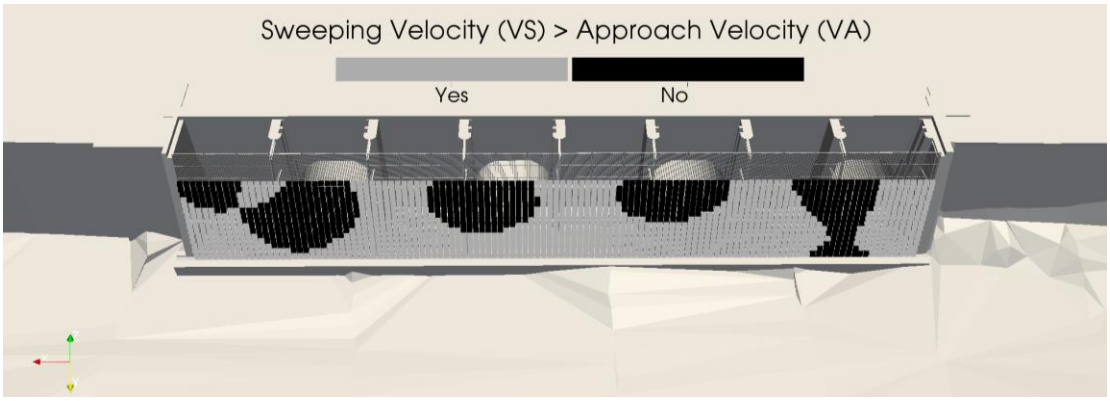

Figure 6: Example of sweeping velocity vs approach velocity plot, visualized in ParaView