Abstract

MultiPoint Thomson Scattering (MPTS) is an established, accurate method of finding the temperature, density, and pressure of a magnetically confined plasma. Two Nd:YAG (1064 nm) lasers are fired into the plasma with a effective frequency of 60 Hz, and the light is Doppler shifted by Thomson scattering. Polychromators on the NSTX-U midplane collect the scattered photons at various radii/scattering angles, and the avalanche photodiode voltages are saved to an MDSplus tree for later analysis. IDL code is then used to determine plasma temperature, pressure, and density from the captured polychromator measurements via Selden formulas.[1]

Previous work[2] converted the single-processor IDL code into Python code, and prepared a new architecture for multiprocessing MPTS in parallel. However, that work was not completed to the generation of output data and curve fits that match with the previous IDL. This project refactored the Python code into a object-oriented architecture, and created a software test suite for the new architecture which allowed identification of the code which generated the difference in output. Another effort currently underway is to display the Thomson data in an intuitive, interactive format.

W. Wallace
Laney College, Oakland, California
wallace.max@gmail.com

Continued Development of Python-Based Thomson Data Analysis And Associated Visualization Tool for NSTX-U

Abstract

MultiPoint Thomson Scattering (MPTS) is an established, accurate method of finding the temperature, density, and pressure of a magnetically confined plasma. Two Nd:YAG (1064 nm) lasers are fired into the plasma with a effective frequency of 60 Hz, and the light is Doppler shifted by Thomson scattering. Polychromators on the NSTX-U midplane collect the scattered photons at various radii/scattering angles, and the avalanche photodiode voltages are saved to an MDSplus tree for later analysis. IDL code is then used to determine plasma temperature, pressure, and density from the captured polychromator measurements via Selden formulas.[1]

Previous work[2] converted the single-processor IDL code into Python code, and prepared a new architecture for multiprocessing MPTS in parallel. However, that work was not completed to the generation of output data and curve fits that match with the previous IDL. This project refactored the Python code into a object-oriented architecture, and created a software test suite for the new architecture which allowed identification of the code which generated the difference in output. Another effort currently underway is to display the Thomson data in an intuitive, interactive format.

Deliverables:
1.finish Python code for MPTS operation on NSTX-U
2.Develop a Visualization Toolkit for Thomson-Scattered Data

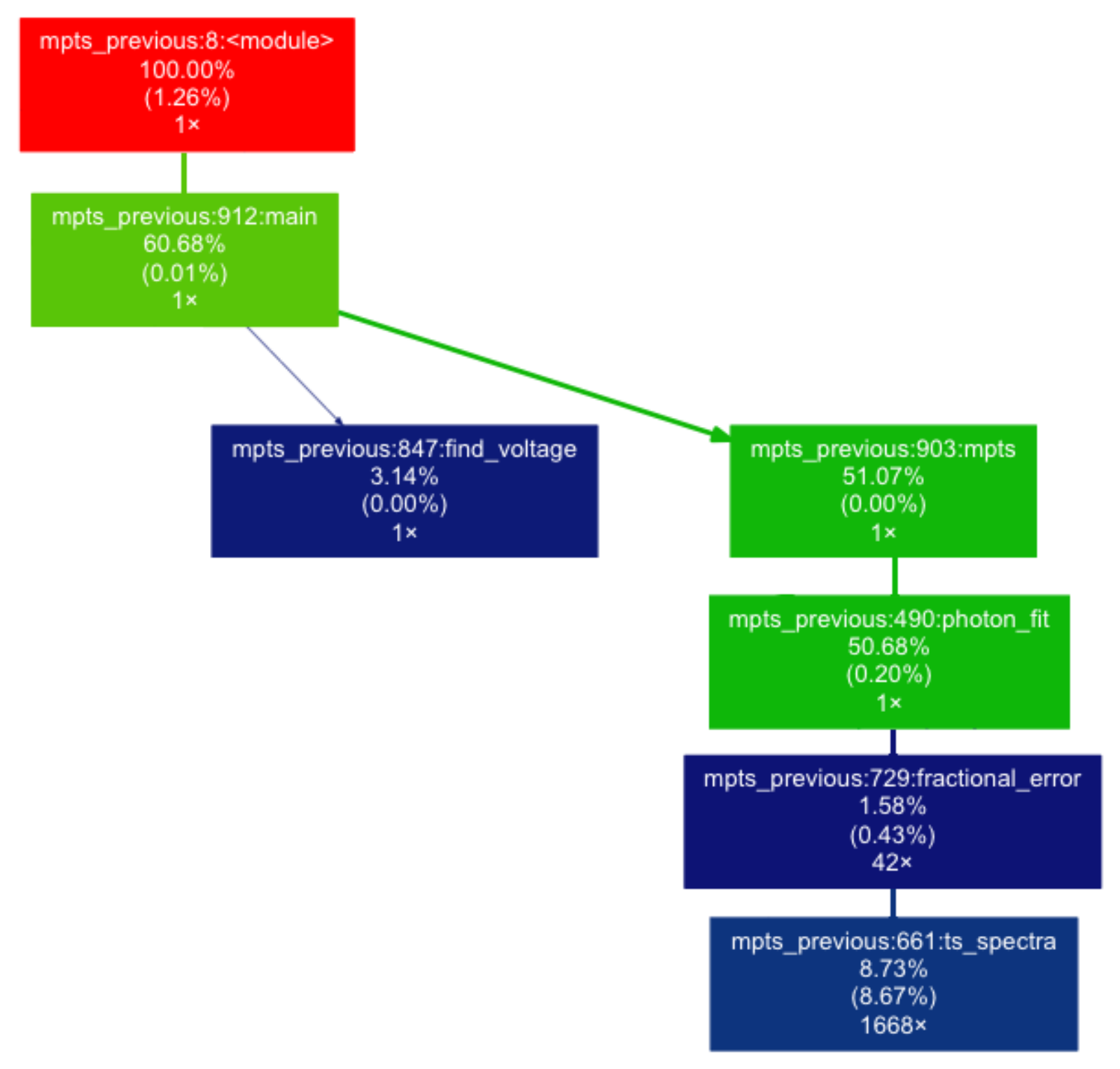


Figure 1. Software profiler output of the original MPTS code before the refactor. The percentage of runtime and total function calls are displayed under each method name. Finding an error in a complex software package such as this is difficult, if possible; and frequently requires a line-by-line execution and a deep understanding of the code flow.

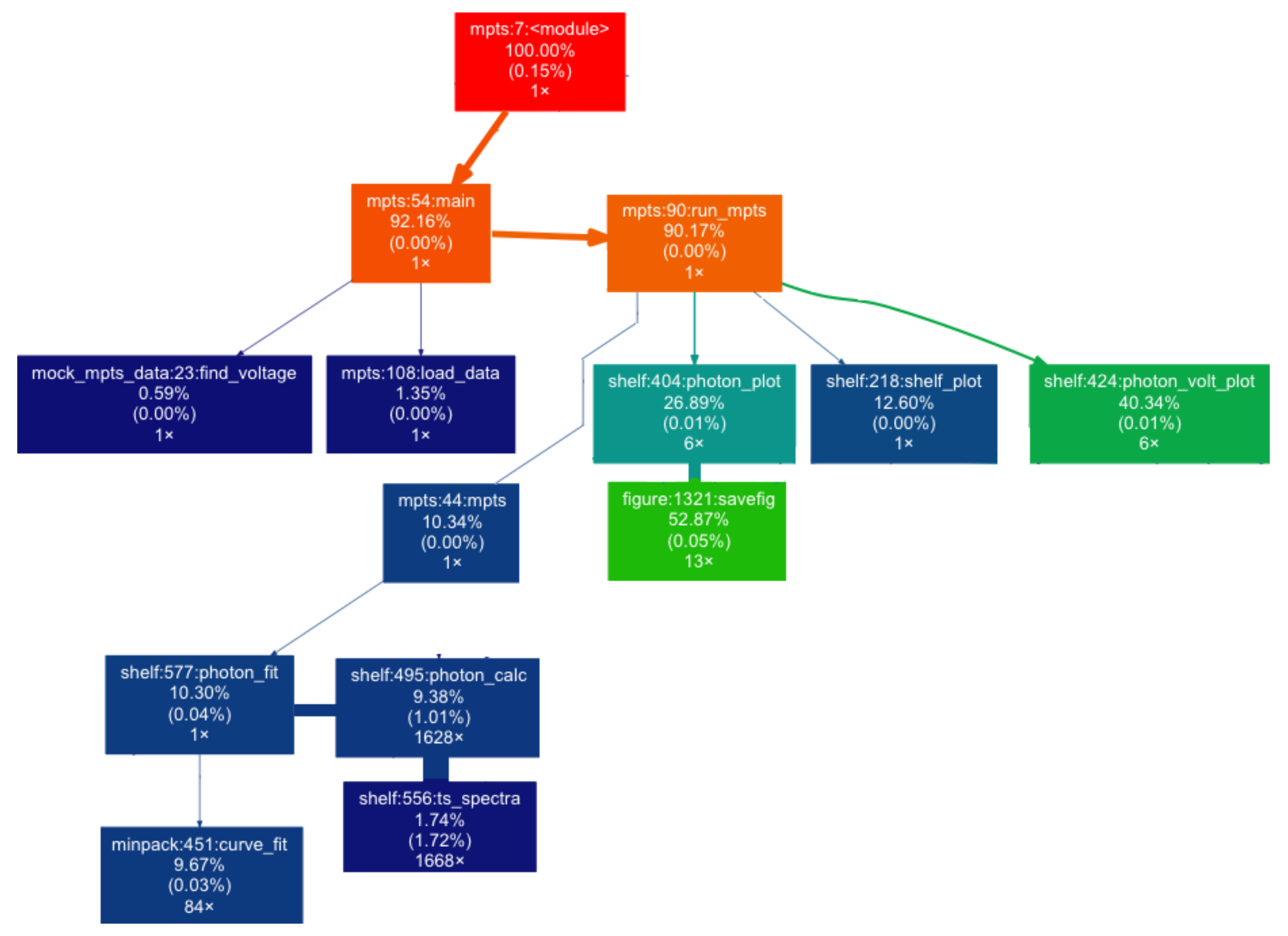


Figure 2. Software profiler output of the refactored MPTS code. While there are more functions being called, less total time is spent in each function. Smaller, more compact code is also easier to test and verify for correctness. The error in this code has been targeted and identified in the curve_fit routine.

Refactoring As Debugging

The bulk of the refactoring effort was to separate the Python code into distinct, atomic code sections. By isolating each section into individually verifiable blocks with known inputs and outputs, previously identified bugs and errors can be located, and new software errors prevented.

Test-driven development was used to validate that the new code behaved exactly the same at every point during the refactor. In addition, errors which would normally propagate throughout the software and only surface during runtime are prevented, as the software is continually verified for correctness during the development effort.

Deliverables:

1. Improved documentation and source code under version control (https://github.com/wallacemax/thompy_wallace)

2.Easier to understand, performant, scalable Thomson processing code ready for multi-processor operation

3.Include Thomson functionality into TV application on cluster to reduce lag time between experimental shot and determination of plasma characteristics.

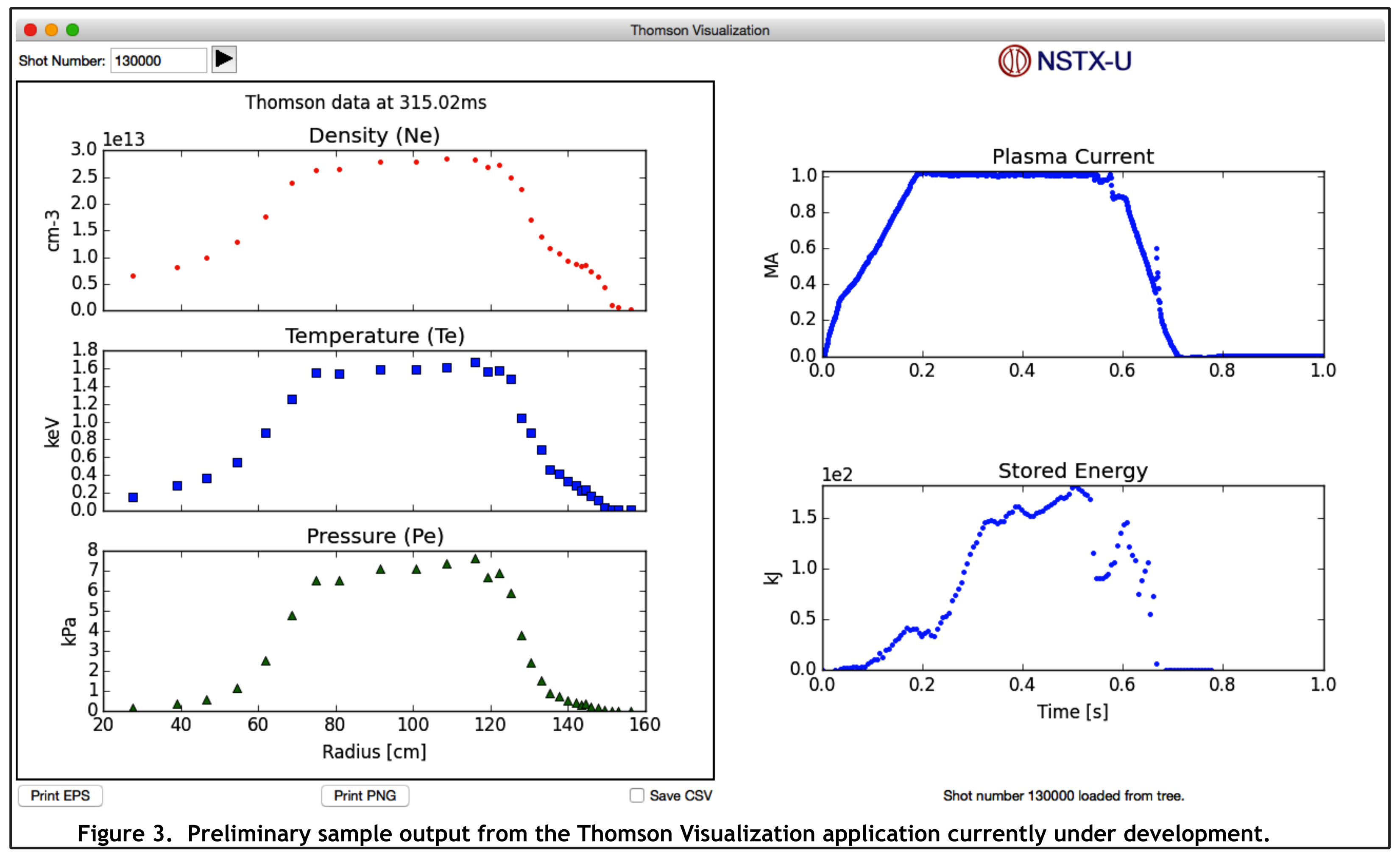
Visualization for Thomson Plot Generation

Another aspect of the project is a Thomson Visualization software to quickly analyze Thomson temperature, pressure, and density data along with other time-based signals simultaneously for correlation and intuition.

Using a Python-library interface to the MDSplus data tree, nodes are loaded and displayed intuitively to a data consumer. An interactive cursor can be used with time-dependent data to display the nearest associated MPTS-sampled data.

At any time point in a specified shot number, a customer may export the plots displayed to either EPS or PNG format. A further option is to also export the displayed data sets in CSV format. The exported plots are saved at 1000DPI for printing fidelity.

The software is still under heavy development, but will ultimately be made available via the computing cluster to interested scientists.



Shot number 130000 loaded from tree.

Figure 3. Preliminary sample output from the Thomson Visualization application currently under development.

References

[1] Selden, A. (n.d.). Simple analytic form of the relativistic Thomson scattering spectrum. Physics Letters A, 405-406.
[2] Miller, J. Optimizing MultiPoint Thomson Scattering diagnostics in NSTXU. APS DPP Annual Meeting, 2014, New Orleans, Louisiana.

Acknowledgements

This work was made possible by funding from the Department of Energy for the Summer Undergraduate Laboratory Internship (SULI) program. This work is supported by the US DOE Contract No. DE-AC02-09CH11466. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Community College Internship (CCI) program.

Future Work

- Replace integration in MPTS Analysis with fitting routine from Bevington in Data Reduction and Error Analysis, 2003
- Write test suite for command-line parallelization for MPTS
- Save author preferences for units, scale, and preferred data series
- Add additional interactive features, i.e., zoom, formatting
- Display and store customer-specified MDSplus data
- Include Thomson data analysis into Visualization tool, removing delay time between NSTX-U experiment shot and plasma characteristics