Modular Python-Based Code for Thomson Scattering System on NSTX-U

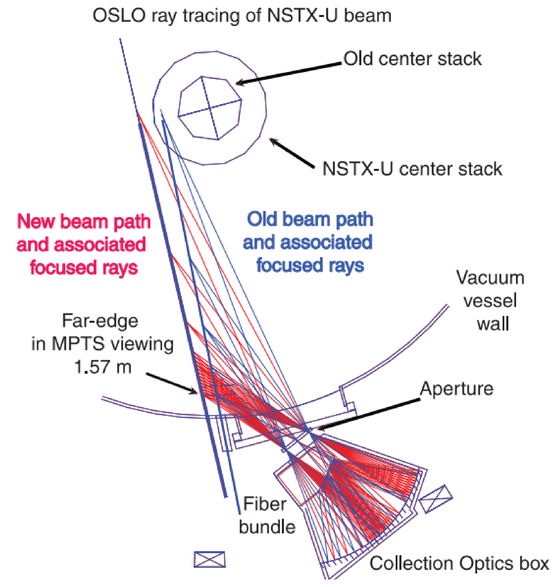
B. Horowitz[1], A. Diallo[2], E. Feibush[2], B.P. LeBlanc[2]

1: Yale University, 2: Princeton Plasma Physics Laboratory

Abstract

Fast, accurate and reliable measurements of electron temperature and density profiles within magnetically confined plasmas are essential for full operation of fusion devices. We detail the design and implementation of a modular Python-based code for the Thomson Scattering diagnostic system of NSTX-U which offers improvements in speed and flexibility by making full use of the Python's architecture and open-source module packages.

Thomson Scattering System



Incoherent Thompson scattering diagnostic setup of NSTX/NSTX-U.

1. A high power laser shone through plasma.
2. Laser light scatters and its intensity in given wavelengths is measured.
3. Total spectrum is recreated by an analysis code.

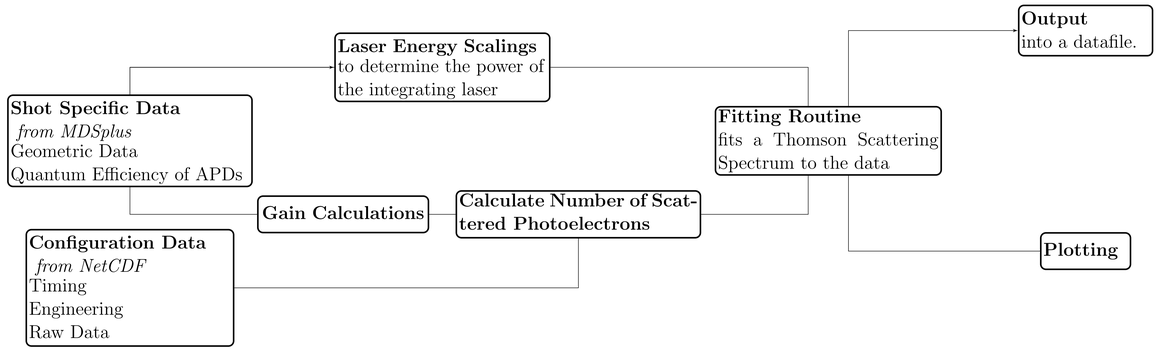
References

[1] A. Diallo, et. al. Prospects for the Thomson scattering system on NSTX-Upgrade In *Rev. Sci. Instrum. 83* (2012)

[2] B.P. LeBlanc, et. al. Radial resolution enhancement of the NSTX Thomson diagnostic In *Rev. Sci. Instrum. 83* (2012)

[3] E. Feibush, et. al. ElVis: A Portal for Scientific Graphics

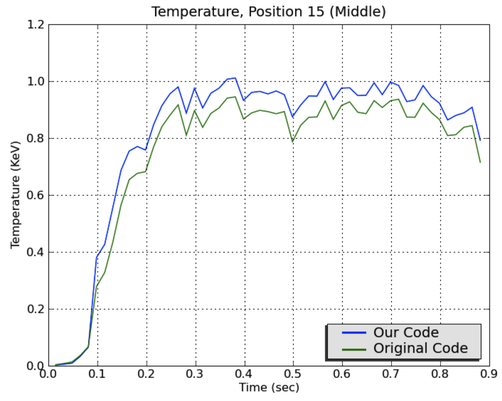
Program Structure Diagram

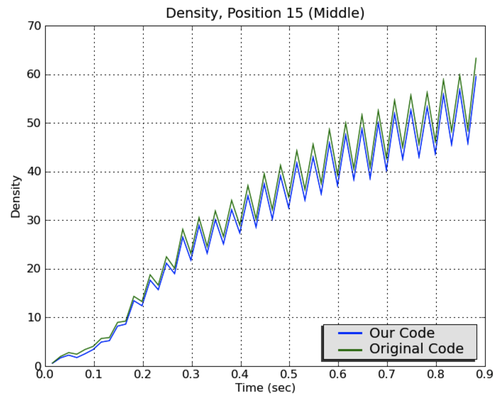


The program includes:

- Thorough in-line documentation.
- Inputs in MDSplus, NetCDF, or CSV.
- Python routines for navigation of MDSplus.
- Utilities for generation of NetCDF files.
- Plotting in MatplotLib, ELVIS, or Gnuplot.
- Python code equivalents for all C++ code.
- Ability to run in Python interactively.

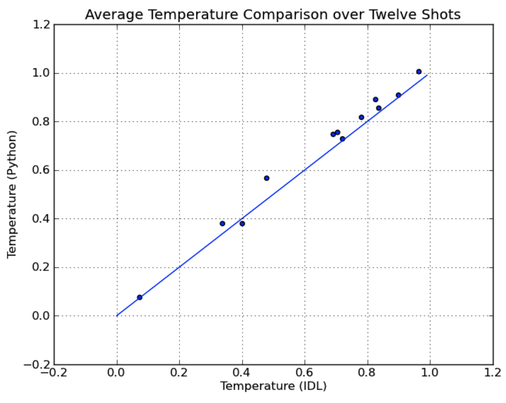
Results

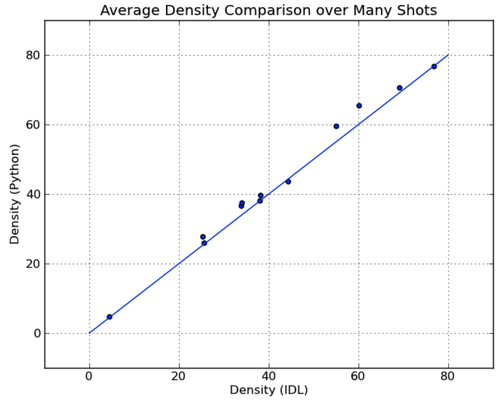




Above: Comparison of temperature and density of two codes (shot number 139047). Note that density computations were uncorrected for differences in laser intensity.

Below: Comparison of average temperature and average density for twelve different shots.





Advantages

There are three main advantages of our program over the existing IDL package.

1. Parallization of fitting routines over multiple processors.
2. Integration of C++ code to clear bottlenecks and make a faster program.
3. Modular components to increase flexibility, user readability, and development speed.

Optimization of Code

Profiling

- Analyze the code using cProfile
- Speed up sections using Numpy, parallelization, or integration of C code

Parallization

- Use pprocess python module to parallelize fitting over each polychromators
- With eight processors, reduced speed by 80%

Integration of C++ Code

- C-based codes are very efficient at performing iterative tasks
- Integrate C/C++ code into Python using weave package
- Could reduce computation time by 2x
- Currently not fully integrated into the program

Future Work

- Adjust error calculations to properly include laser energy and weighting factors.
- Integrate C/C++ code into fitting routine.
- Include weighting factors into fitting routine.
- Make compatible with NSTX-U and other Thomson scattering devices.
- Make parallelization efficient use all available processors.