Software Development for Wind Profiler Signal Processing Using Python with NumPy and SciPy

Noor Hafizah Binti Abdul Aziz[1,2], Masayuki K. Yamamoto[1], Toshiyuki Fujita[1],
Hiroyuki Hashiguchi[1], Mamoru Yamamoto[1]
[1]Research Institute for Sustainable Humanosphere (RISH), Kyoto University, Japan.
[2]Faculty of Electrical Engineering, Universiti Teknologi MARA (UiTM), Malaysia.

1. Introduction

Wind profiler is a useful means to measure altitude profiles of vertical and horizontal wind velocities with high time and vertical resolutions (Hocking, 2011). Range imaging (RIM) is a technique that improves range resolution down to several ten meters by using frequency diversity and adaptive signal processing. RIM is useful for resolving fine-scale structure of atmospheric instability such as Kelvin-Helmholtz billows. Therefore RIM can be used for early detection of small-scale turbulence. In order to develop an algorithm that detects the small-scale turbulence automatically, a software with high portability (i.e. work under multiple platform) and generality (i.e. written by popular software language) is required to be developed. In developing software, we are using Python with SciPy and NumPy libraries.

2. USRP2 as a Radar Digital Receiver

USRP2 (Universal Software Radio Peripheral 2) is a software defined radio receiver. USRP2 is controlled by UHD ('Universal Software Radio Peripheral' Hardware Driver). UHD is available from C++ and Python. Software defined radar receiver is able to be developed using free and popular software languange. Because USRP2 does not have trigger terminal to start and stop data sampling, the leakaged of transmission signals are used for ranging. Oversampling facility is easy to be implemented because of high-time resolution data sampling up to 25 MHz is available. Data were collected by USRP2 and 1.3-GHz LQ7 transmission system (Imai et. al., 2007).

3. Python with NumPy and SciPy

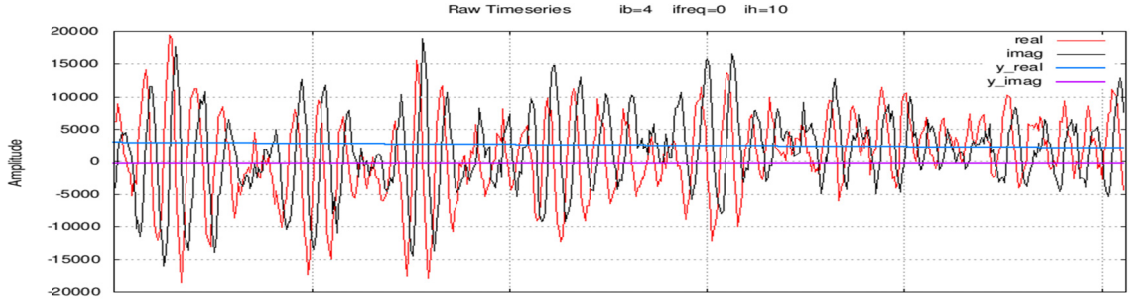
Python runs on Windows, Linux/Unix, Mac OS X, and has been ported to the Java and .NET virtual machines which is free to use, even for commercial products, because of its OSI-approved open source license. SciPy library is built to work with NumPy arrays which is the fundamental package needed for scientific computing with Python. NumPy and SciPy contain a powerful N-dimensional array object, sophisticated functions such as linear algebra, Fourier transform, and random number capabilities. Indexing like operations can be used in this data signal processing (example 'numpy.where' which returns array indices which satisfy given conditions). Other useful mathematical operators such as numpy.mean, numpy.max, numpy.empty, numpy.sum, numpy.array, numpy.copy and numpy.dot is used to increase the calculation and coding efficiency. Using Python, radar signal processing software has been developed in order to estimate spectral parameters (i.e. echo power, Doppler velocity and Doppler velocity variance).

4. Signal Processing Flow

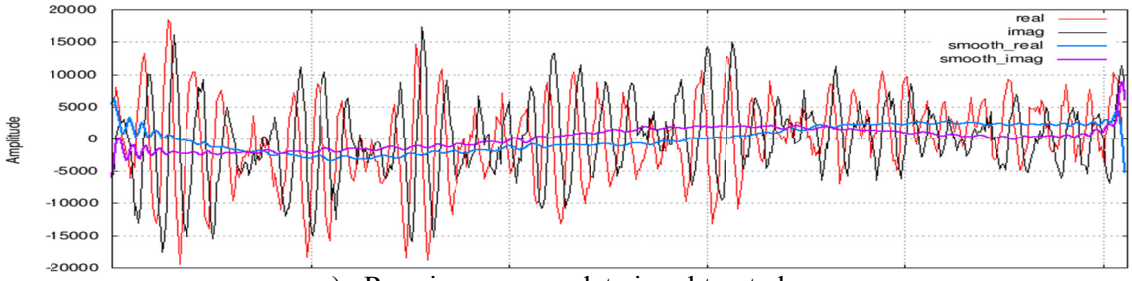
The signal processing starts from IQ signal detection and analog to digital (A/D) conversion which had been processed in USRP2. In the PC, the IQ signals are ranged, decoded, and averaged in time (i.e. coherent integration). Subsequently, in offline signal processing which developed using Python with NumPy and SciPy, it is divided into two parts: time series signal processing focused on clutter rejection and spectral parameter estimation. In the time series signal processing, DC component had been removed and then high-pass filtering is done by running mean method. Ranging, decoding and coherent integration are processed by online or offline. In the second section, there are three calculations: Doppler spectrum calculation by FFT, noise level estimation and spectral parameter (i.e. echo-power, Doppler velocity and Doppler spectrum) estimation by moment method.

Figure 1 shows a result of time series signal processing. Figure 1a shows a raw IQ time series data, and red and black curves show real and imaginary component, respectively. Figure 1b shows a time series after DC component is removed (black and red curves). Low frequency IQ signals computed by 100 points running average are shown by blue (real component) and purple (imaginary component) curves. Figure 1c shows the time series after the running-averaged data are subtracted (i.e., highpass filtered data). The time series data shown in Figure 1c are used for spectral moment estimation.

a) Raw time series



b) DC had been removed and low-frequency signals computed by 100 points running average



c) Running average data is subtracted

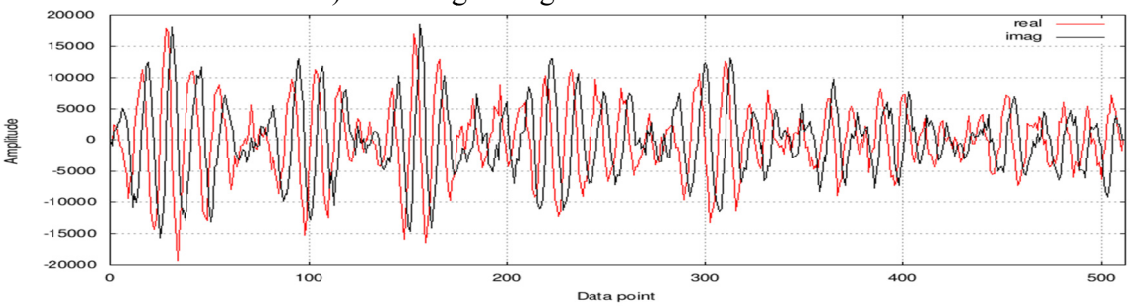


Fig. 1 : Example of received time series.

Figure 2 shows a result of spectral data using the time series shown in Figure 1. Red curve is a Doppler spectrum computed from raw time series (see Figure 1a) and black curve is a corrected Doppler spectrum from high-pass filtered time series (see Figure 1c). Clear-air echo exist in the range between 2 to 3 ms^{-1} and noise level is 92 dB. For executing FFT, scipy.fftshift and scipy.fftpack are used.

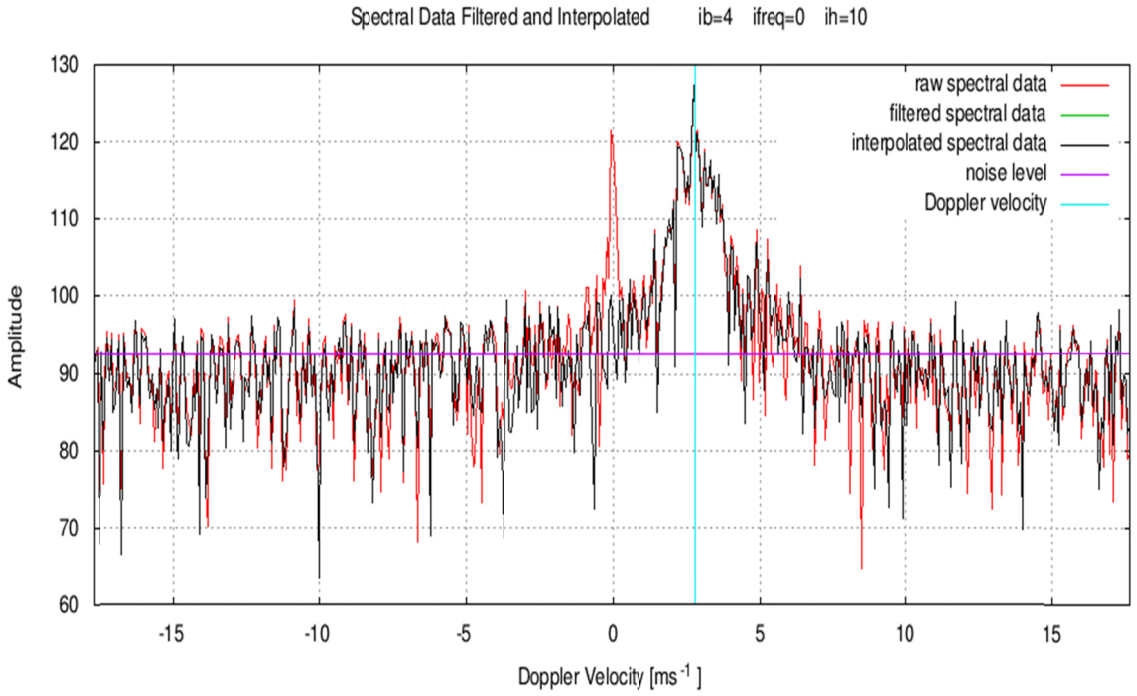


Fig. 2 : Example of spectral data.

In spectral parameter estimation, the following procedures are taken. (i) Noise level calculation using Hildebrand (1974), (ii) 5 points running mean (smoothing) to the Doppler spectra, (iii) peak search, (iv) determination of continuous Doppler velocity range where received power is greater than threshold (noise level + 3×(noise standard deviation)), and (v) spectral parameter estimation using the moment method. In the all procedures, numpy.where and numpy.max are used. Figure 3 shows a result of spectral parameter estimation. The black curve is raw Doppler spectrum and green curve is smoothed Doppler spectrum. The Doppler velocity range selected by smoothed Doppler spectrum and the threshold (blue dotted line) is used for spectral parameter estimation such as echo power, standard deviation and Doppler spectra. These calculations use numpy.mean, numpy.where and numpy.max.

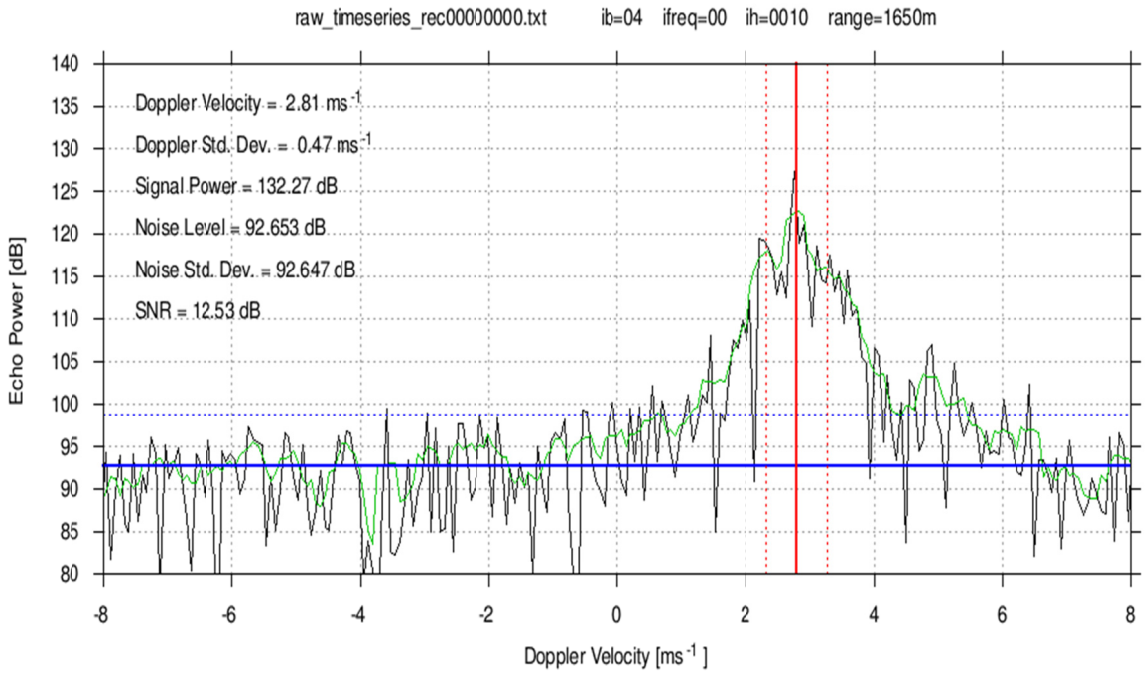


Fig. 3 : An example of spectral parameter estimation result.

5. Oversampling Result

Figure 4 shows a spectral parameter estimation result using original range resolution determined by 1-μs transmitted pulse width (150m), and Figure 5 shows a result of 10 times oversampling. The result demonstrates that even the simple oversampling can reveal the fine-scale vertical changes of received signals. The vertical variability of Doppler spectra is much clearer than no-oversampling scale. Therefore, oversampling is useful for high accurate reproduction of turbulence structure by further using RIM and for developing an algorithm to reveal fine-scale turbulence structure by decorrelating signals in range.

6. Conclusion

A radar signal processing software necessary for building detection algorithm of small-scale turbulence using RIM is being developed using Python with NumPy and SciPy package. Our results demonstrate that Python with NumPy and SciPy is a powerful tool for radar signal processing. The software includes the following functions remove clutter signals by high pass filtering, calculate Doppler spectrum using FFT, estimate noise level using method by Hildebrand (1974), find the Doppler velocity range where radar echo exists, and estimate spectral parameters with good accuracy. We showed the example that the software we developed is useful for mitigating clutter signals. Further we showed a capability of oversampling to reveal fine-scale structure of radar echo and wind velocity. Further development will be show in subsequent studies.

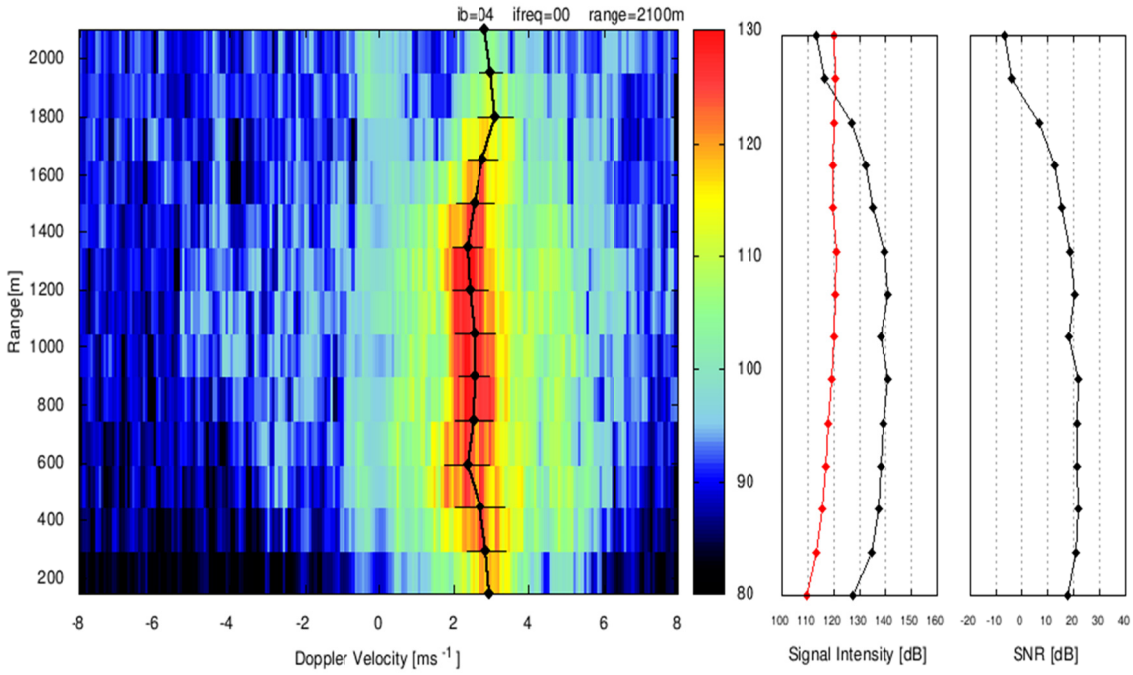


Fig. 4 : Result using 150-m resolution (i.e., without oversampling).

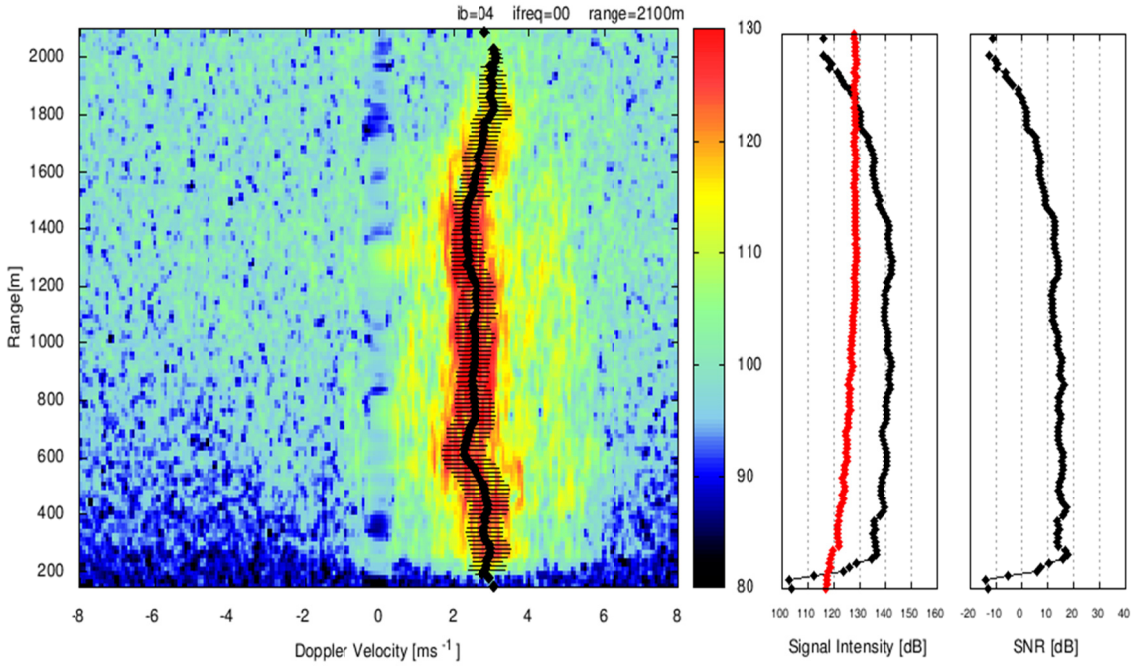


Fig. 5 : Result using 10-times (15-m) oversampling.

Acknowledgement

This research was supported by Adaptable and Seamless Technology Transfer Program through Target-Driven R&D (A-STEP) Exploratory Research (Research No. AS232Z00186A).

References

1. Hildebrand, P.H., and Sekhon, R.S., Objective Determination of the Noise Level in Doppler Spectra, Appl. Meterol., 13, 808-811, 1974.
2. Hocking, W.K., A review of Mesosphere-Stratosphere-Troposphere (MST) Radar Developments and Studies, circa 1997-2008, Journal Atmosphere Solar Terr. Phys., 73, 848–882, 2011.
3. Imai, K., Nakagawa, T., and Hashiguchi, H., Development of Tropospheric Wind Profiler Radar with Luneberg Lens Antenna (WPR LQ-7), Sumitomo Electric Technical Review, 38-42, 2007.