東京大学地震研究所技術研究報告，No. 26，31–36 頁，2020 年.
Technical Research Report, Earthquake Research Institute, the University of Tokyo, No. 26, pp. 31–36, 2020.

報　告

WIN フォーマットデータを ObsPy で読み込む新しいモジュール

中川茂樹 *† ・加藤愛太郎 **

New Module for Reading WIN Format Data in ObsPy

Shigeki NAKAGAWA*† and Aitaro KATO**

は　じ　め　に

地震波形データを解析するツールとして，近年，ObsPy（Beyreuther *et al.*, 2010）がよく用いられている．このツールは，様々なフォーマットで記述された地震波形データを取り扱うことが可能で，その中には我が国で広く一般に用いられている WIN フォーマット（卜部，1994）も含まれている．しかし，Jones *et al.*（2020）が指摘しているように ObsPy に含まれる WIN フォーマットの読み込みモジュールには実装のバグも含めて大きく 3 つの問題が存在する．それは，1）WIN フォーマットの 0.5 バイト及び 3 バイト圧縮されたデータの解凍にバグがある，2）欠測が含まれるデータやサンプリングレートが変化するデータを正しく取り扱えない，3）動作が遅い，である．特に 3 番目の問題は，大量のデータを取り扱う際に影響が大きく，例えば，6800 チャネル 1 分長の WIN ファイルを読み込むのに約 150 秒を要している．そこで，これらの問題に対応した新たなモジュールを製作することにした．

バグ修正の試み

圧縮データに関する問題を解決するために ObsPy 付属の WIN 読み込みモジュールの修正を試みた．

WIN フォーマットは秒単位でデータをパケット化している．例えば，1 秒分のデータについてそれぞれの直前サンプルとの差分が −8 から 7 の間に収まっている場合，0.5 バイト圧縮が選択される．0.5 バイトの差分データは上位 4 ビットと下位 4 ビットに順番に格納している．この圧縮データの取扱いについて 0.5 バイトと 3 バイトの場合に ObsPy 付属の WIN 読み込みモジュールにバグが存在した．

0.5 バイト圧縮されたデータは，通常 1 バイトずつメモリに読み込まれており，それを上位 4 ビットと下位 4 ビットに分割する．この下位 4 ビットを取り出す操作にバグが存在した．WIN システムのソースコード（卜部・東田，1992）では，

```
abuf[i + 1] = abuf[i] + (((int8_w)(*(dp++) << 4)) >> 4);
```

と書かれており，差分の 1 バイトデータを左 4 ビットシフトして符号付き 1 バイト整数にキャストし，その後右 4 ビットシフトしている．しかし，ObsPy 付属の WIN 読み込みモジュールのソースコードでは，

```
idata2 = idata2 +¥
        (from_buffer(sdata[i : i + 1],
                     np.int8)[0] << 4) >> 4
```

となっていて，WIN システムのソースコードにある 1 バイト整数にキャストする部分に相当する動作が抜けている．結果として，ビット演算はしているものの，左 4 ビットシフトして右 4 ビットシフトすると元の値に戻ることになる．我々は，修正案として，ビットシフト演算に代わり論理積を導入することにした．

```
s4(from_buffer(sdata[i : i + 1],
               np.int8)[0] & 0b00001111)
```

これにより下位 4 ビットのデータを取得できるようになった．なお，符号無し整数を符号あり整数に変換する関数 s4 を新たに定義した．

0.5 バイト圧縮されたデータのサンプリングレートが偶数の時，差分データの個数は奇数であり，WIN フォーマットのデータブロックの最後の 1 バイトの下位 4 ビットに未使用の空きができる．データを読み取る際にこの空きは飛ばす必要があるが，ObsPy では考慮されていなかった．

2020 年 11 月 4 日受付，2021 年 1 月 4 日受理.
† nakagawa@eri.u-tokyo.ac.jp
* 東京大学地震研究所地震火山情報センター
** 東京大学地震研究所地震予知研究センター
* Earthquake and Volcano Information Center, Earthquake Research Institute, the University of Tokyo.
** Earthquake Prediction Research Center, Earthquake Research Institute, the University of Tokyo.

```
--- core.py.ORG 2019-03-04 10:23:24.000000000 +0900
+++ core.py    2019-03-12 20:30:28.050444596 +0900
@@ -52,6 +52,10 @@
        return False
    return True

+def s4(v):
+   if(v & 0b1000):
+       v = -((v - 1) ^ 0xf)
+   return(v)

 def _read_win(filename, century="20", **kwargs):  # @UnusedVariable
    """
@@ -137,11 +141,14 @@
             if datawide == 0.5:
                 for i in range(xlen):
                     idata2 = output[chanum][-1] + ¥
-                        from_buffer(sdata[i:i + 1], np.int8)[0] >> 4
+                        s4((from_buffer(sdata[i:i + 1],
+                                np.int8)[0] & 0b11110000) >> 4)
                     output[chanum].append(idata2)
+                    if i == (xlen -1):
+                        break
                     idata2 = idata2 +¥
-                        (from_buffer(sdata[i:i + 1],
-                                np.int8)[0] << 4) >> 4
+                        s4(from_buffer(sdata[i:i + 1],
+                                np.int8)[0] & 0b00001111)
                     output[chanum].append(idata2)
             elif datawide == 1:
                 for i in range((xlen // datawide)):
@@ -157,8 +164,8 @@
             elif datawide == 3:
                 for i in range((xlen // datawide)):
                     idata2 = output[chanum][-1] +¥
-                        from_buffer(sdata[3 * i:3 * (i + 1)] + b' ',
-                                native_str('>i'))[0] >> 8
+                        (from_buffer(sdata[3 * i:3 * (i + 1)] + b' ',
+                                native_str('>i'))[0] >> 8)
                     output[chanum].append(idata2)
             elif datawide == 4:
                 for i in range((xlen // datawide)):
```

図 1. ObsPy の WIN I/O モジュールに対する差分．unified 形式で表示してある．unified 形式の差分は，@@ がファイルの行番号，+ が挿入された行，− が削除された行を示す．unified 形式の詳細については，diff(1)のマニュアルを参照していただきたい．

我々はその点も修正した．

また，3 バイト圧縮された差分データを直前サンプルの値に加算する際，ビット演算の優先度は加算よりも低いので，括弧が必要であった．ObsPy 付属の WIN 読み込みモジュールのソースコードでは抜けていたので，追加した．

これらの修正をソースコードに施した．ソースコードの差分を図 1 に示す．修正により，1 番目の問題点であった圧縮データの解凍に関するバグは修正された．しかし，2 番目と 3 番目の問題点は依然残ったままであった．特に，動作速度の改善には，ソースコードの全面的な修正が必要と思われた．

新しいモジュールの導入

ObsPy 付属の WIN 読み込みモジュールのソースコードを部分的に修正するだけでは，全ての問題を解決するには効率が悪いと考えられたので，新しいモジュールを作成することにした．新しいモジュールに求められることは，WIN フォーマットデータを読み込み，ObsPy の Stream オブジェクトにデータを格納して返すモジュールで，冒頭に挙げた 3 つの問題点を解決すること，である．加えて，バグの低減と保守性の向上も重要である．

さて，Python には共有ライブラリ内の関数を呼び出す ctypes と呼ばれるライブラリが標準で用意されている．Python スクリプトから共有ライブラリを利用することは，C 言語等のコンパイラ言語と同等の処理速度が期待でき，動作速度の改善などのために使われる典型的なテクニックの一つである．また，既存のライブラリを利用することから，C 言語のソースコードを Python 言語に翻訳する必要がなく，従って翻訳時にバグが混入するといった心配も無くなる．さらに，機能拡張や仕様変更等が行われた際もそれら変更は共有ライブラリ側で行われることが多いので，対応にかかる工数を削減することが期待される．一方で，共有ライブラリにバグがあれば当然にそれを継承するし，ソースコードが煩雑になる恐れはある．ctypes の利用にはメリット・デメリットを勘案する必要があるが，ObsPy でも miniSEED フォーマットデータの取扱いにおいて，IRIS が配布する共有ライブラリ libmseed を ctypes から利用している．そこで，我々もこれに倣い，WIN システムが備える共有ライブラリを利用することにした．具体的には，植平ほか（2010）が WIN システムに導入した libwinsystem という共有ライブラリを用いることにした．libwinsystem は WIN システムをコンパイルする時に同時に作成される共有ライブラリの一つで，WIN フォーマットデータを読み書きするための関数が含まれている．本研究では，この共有ライブラリに含まれる関数のうち，WIN フォーマットデータを 4 バイト整数に変換する win2fix() と秒ヘッダーをデコードする bcd_dec() を用いた．

本研究で製作したモジュールのソースコードを図 2 に示す．ObsPy 付属の WIN 読み込みモジュールと取り替えて使用できるように，引数はファイル名だけとした．

ObsPy 付属の WIN 読み込みモジュールが抱える 3 つの問題点への対応は次の通りである．まず，1）WIN フォーマットの 0.5 バイト及び 3 バイト圧縮されたデータの解凍にバグがある点については，libwinsystem に含まれる WIN フォーマットデータを 4 バイト整数に変換する win2fix() を導入したことから自動的に解決された．副産物として，現在の WIN フォーマットにはデータ圧縮しないオプションも実装されているが，それにも自動的に対応することができた．次に，2）欠測が含まれるデータやサンプリングレートが変化するデータを正しく取り扱えない原因は，ObsPy 付属の WIN 読み込みモジュールでは，チャネル番号毎に ObsPy の Stream オブジェクトを構成して

```
#!/usr/bin/env python3
'''
Version 0.85    By S.N.
DONE:
 + missing data (gap): code ha dasai
 + variable sampling rate (gap)
 + time zone
TODO:
 + misisng data (zero-pad, hold-prev-data)
 + WIN32
'''
import sys
import ctypes as C
import numpy as np
import struct
from obspy import Stream, Trace, UTCDateTime
import time

TZ = '+0900'
#TZ = ''

HEADER_5B = 1048576
MAX_SR = HEADER_5B

__clibwinsystem = C.CDLL("/usr/local/win/lib/libwinsystem.so")

__clibwinsystem.bcd_dec.argtypes = (C.c_int * 6, C.c_char_p)
__clibwinsystem.bcd_dec.restype = None

__clibwinsystem.win2fix.argtypes = (C.c_char_p,
                                    np.ctypeslib.ndpointer(dtype=np.int32,                ndim=1,
flags='C_CONTIGUOUS'),
                                    C.POINTER(C.c_uint), C.POINTER(C.c_long))
__clibwinsystem.win2fix.restype = C.c_ulong

__clibwinsystem.win_chheader_info.argtypes      =      (C.c_char_p,      C.POINTER(C.c_uint),
C.POINTER(C.c_long),
                                                    C.POINTER(C.c_int))
__clibwinsystem.win_chheader_info.restype = C.c_ulong

class windata:
    def __init__(self, chid, begintime, endtime, sr, d):
        self.chid = chid
        self.begintime = begintime
        self.endtime = endtime
```

図 2.　開発したモジュールのソースコード.

```
            self.sr = sr
            self.d = d

def _read_onesec(data, start, output, ch2tr, maxtr):
    tt = (C.c_int * 6)(0,0,0,0,0,0)
    abuf = np.empty(MAX_SR, dtype=np.int32)
    g_size = C.c_ulong()
    sys_ch = C.c_uint()
    s_rate = C.c_long()

    blksizetmp = data[start:start+4]
    blksize = struct.unpack(">I", blksizetmp)[0]
    buff = data[start+4:start+10]
    __clibwinsystem.bcd_dec(tt, buff)

    if tt[0] < 70:
        tt[0] += 2000
    else:
        tt[0] += 1900
    begdate = UTCDateTime(tt[0], tt[1], tt[2], tt[3], tt[4], tt[5])
    prevdate = begdate - 1

    blksize -= 10
    buff = data[start+10:start+10+blksize]

    offset = 0
    while offset < blksize:
        g_size = __clibwinsystem.win2fix(buff[offset:], abuf, sys_ch, s_rate)
        chid = sys_ch.value
        if chid in ch2tr and output[ch2tr[chid]].endtime == prevdate and output[ch2tr[chid]].sr ==
s_rate.value:
            ii = ch2tr[chid]
            output[ii].d = np.append(output[ii].d, abuf[0:s_rate.value])
            output[ii].endtime = begdate.timestamp
        else:
            ii = maxtr
            maxtr = maxtr + 1
            ch2tr[chid] = ii
            output.append(windata(chid, begdate.timestamp, begdate.timestamp, s_rate.value,
np.copy(abuf[0:s_rate.value])))

        offset += g_size

    del(abuf)
    return((blksize + 10, maxtr))
```

図 2. 開発したモジュールのソースコード（続）.

```
def read_winfile(fname):
    output = []
    ch2tr = {}
    maxtr = 0

    with open(fname, "rb") as fpin:
        data = fpin.read()
        datasize = len(data)
        offset = 0
        while offset < datasize:
            (blksize, maxtr) = _read_onesec(data, offset, output, ch2tr, maxtr)
            offset += blksize
    fpin.close()

    traces = []
    for i in range(maxtr):
        t = Trace(data=np.array(output[i].d).flatten())
        t.stats.channel = format(output[i].chid, '04X')
        t.stats.sampling_rate = float(output[i].sr)
        if not TZ:
            t.stats.starttime = UTCDateTime(output[i].begintime)
        else:
            stime0 = str(UTCDateTime(output[i].begintime))
            stime1 = stime0.replace('Z', TZ)
            t.stats.starttime = UTCDateTime(stime1)
        traces.append(t)
            t.stats.starttime = UTCDateTime(stime1)
        traces.append(t)

    del(output)
    del(ch2tr)
    return Stream(traces=traces)

if __name__ == '__main__':
    winfname="991109.064607"
    time_sta = time.perf_counter()
    st = read_winfile(winfname)
    time_end = time.perf_counter()
    ela_time = time_end - time_sta
    print('Elapsed time = %f sec' % ela_time)

    st.plot()

    sys.exit(0)
```

図 2. 開発したモジュールのソースコード（続）.

いるTraceオブジェクトを生成し，時刻等の不連続をチェックすること無く，新しい時刻のデータを順番に追加していたために生じていた．そこで，本研究の新しいモジュールでは，チャネル番号毎に生成したTraceオブジェクトにデータを追加する際に，時刻の不連続やサンプリングレートの変化を確認し，不連続が検知された場合には新しいTraceオブジェクトを生成して，Traceオブジェクトを分割することにより対応した．

最後に，3）動作が遅い点について，C言語で書かれたWINシステム付属のライブラリを用いたため大幅に向上していることが期待されるが，確認のために簡単な比較テストを行った．比較テストには，ObsPy付属のWIN読み込みモジュールに圧縮解凍のバグ修正を適用したモジュールと本研究で新たに作成したモジュールを用いた．テストに用いたWINフォーマットデータは，6800チャネルの1分長データ（サイズは約50 MB）と18チャネル62秒長データ（サイズは約70 KB）である．テスト環境は，CPUはCore i7 4770（3.4 GHz），メモリは16 GBで，OSはCentOS 7.7（1908），ソフトウェアはPython 3.6.8，ObsPy 1.1.1，WINシステム3.0.6を用いた．比較テストはそれぞれのデータとモジュールについて5回ずつ読み込みに要した時間を計測した．それぞれの計測前にページキャッシュを解放して，メモリ上にキャッシュされたデータが用いられないように注意した．比較テストの結果を表1に示す．本研究で作成したモジュールはObsPy付属のモジュールに比べて，約5〜7倍程度速くWINフォーマットデータを読み込めることがわかった．特に，6800チャネル1分長のデータを読み込むのにObsPy付属のWIN読み込みモジュールでは2.5分程度要していたのが，本研究で作成したモジュールを使うと30秒以下で読み込むことが可能で，データ解析においても現実的な読み込み時間になったと考えられる．

表 1. WINフォーマットデータの読み込みに要した時間．

Benchmark 1 (continuous data; 1 minute, 6800 channels; 50MB)			
	ObsPy (patched) [sec]	Our new module [sec]	ratio
	150.009464	25.246116	5.9
	150.883847	26.673333	5.7
	156.344153	25.536667	6.1
	153.570893	26.897626	5.7
	147.446116	25.729688	5.7
average	151 ± 3.06	26.0 ± 0.65	5.8±0.2

Benchmark 2 (event data; 62 seconds, 18 channels; 70KB)			
	ObsPy (patched) [sec]	Our new module [sec]	ratio
	0.545416	0.099736	5.5
	0.551342	0.072489	7.6
	0.620417	0.085906	7.2
	0.612111	0.09762	6.3
	0.629641	0.071972	8.7
average	0.59 ± 0.036	0.086 ± 0.012	6.9±1.1

まとめ

ObsPy付属のWIN読み込みモジュールには，WINフォーマットデータの圧縮解凍や欠測等の取扱いにバグがあり，しかも動作が遅いという問題点が存在した．我々は，WINシステムの共有ライブラリを用いた新たなモジュールを作成し，これらの問題点を解決した．データの読み込み速度は約5〜7倍に大幅に向上した．今後はHi-net（National Research Institute for Earth Science and Disaster Resilience，2019）等で採用されているWIN32フォーマットへの対応や欠測値のゼロ詰め又は前値保持オプションの実装，さらにWINフォーマットへのデータ書き出しや植平ほか（2020）で検討が進められているチャネル番号の拡張等への対応が必要と考えている．

なお，開発したモジュールは著者のホームページ（http://www.eri.u-tokyo.ac.jp/people/nakagawa/win/）において公開しているが，配布方法についても検討していきたい．

謝　辞：馬場聖至准教授と西田究准教授の査読意見は，本稿の内容を改善する上で大変有益でした．深く感謝します．なお，本研究の一部は，文部科学省による「災害の軽減に貢献するための地震火山観測研究計画（第2次）」とJST，CREST，JPMJCR1763の支援を受けました．

文　献

Beyreuther, M., R. Barsch, L. Krischer, T. Megies, Y. Behr and J. Wassermann, 2010, ObsPy: A Python toolbox for seismology, *Seismol. Res. Lett.*, **81**, 530–533, doi: 10.1785/gssrl.81.3.530.

Jones, J.P., K. Okubo, T. Clements and M.A. Denolle, 2020, SeisIO: A fast, efficient geophysical data architecture for the Julia Language, *Seismol. Res. Lett.*, **91**, 2368–2377, doi: 10.1785/0220190295.

National Research Institute for Earth Science and Disaster Resilience, 2019, Research Data set, NIED Hi-net, doi:10.17598/nied.0003.

植平賢司・卜部卓・鶴岡弘・中川茂樹，2010，WINシステムの64bit環境への対応，日本地震学会2010年秋季大会予稿集，C11–08.

植平賢司・中川茂樹・鶴岡弘・卜部卓，2020，WINフォーマットにおけるチャンネル番号の拡張，日本地震学会2020年秋季大会予稿集，S02–03.

卜部卓・束田進也，1992，win－微小地震観測網波形験測支援のためのワークステーション・プログラム（強化版），日本地震学会1992年秋季大会予稿集，P41.

卜部卓，1994，多チャンネル地震波形データのための共通フォーマットの提案，日本地震学会1994年秋季大会予稿集，P24.