JARE54 で持ち込む超音波風速計の開発レポート

沖田博文

2012/2/3
2012/2/20 改訂

1 Abstract

C_T^2 測定装置の開発を行った。C_T^2 の測定には Applied Technologies, Inc. の Sonic Anemometer/Thermometer Model #SATI-3SX 超音波風速計を用いた。測定データは RS232C 経由で出力される。Linux PC (Ubuntu 10.10, kernel 2.6) で書き出すためにソフト sonic を C 言語で開発した。得られたデータから C_T^2 を求めるため sonic.awk を、自動観測を行うために sonic.sh の開発も行った。

#SATI-3SX の校正を自作の “Zero Air Chamber” を用いて常温で行った。校正後に測定値のばらつきを調べた。測定の結果、風速の測定誤差は ±0.0142[m/s]、温度の測定誤差 ±0.0160[°C] であった。この値はカタログ値の精度 (風速 ±0.03[m/s], 温度 ±0.1[°C]) よりも小さかった。この結果から平均風速 5.8[m/s] のドームふじ基地での装置固有の誤差を見積もると $\Delta C_T^2 \sim \pm 0.0005[\mathrm{K^2 m^{-2/3}}]$ 程度となった。南極での接地境界層における C_T^2 は 0.01∼0.001$[\mathrm{K^2 m^{-2/3}}]$ 程度と予想されるため、この#SATI-3SX は十分な測定精度を有している事がわかった。

なお実験に使用した Linux PC にはシリアルポートが無かったため秋月電子通商 USB シリアル変換ケーブル (グレー色、変換チップは Prolific PL2303) を用いたが問題なく使用することが出来た。また#SATI-3SX の初期設定は Windows PC (Windows 7 Professional) から Tera Term を用いる必要があった。実際にドームふじ基地で無人観測する際は crontab によって自動的に観測を実行し C_T^2 値のみを日本に転送する予定である。

2 イントロダクション

第 54 次日本南極地域観測隊 (2012 年 11 月∼2013 年 3 月に実施、以降 JARE54) では内陸ドームふじ基地への旅行が計画されている。JARE54 での主目的は先の第 53 次隊 (JARE53) において砕氷船「しらせ」によって輸送した赤外線望遠鏡観測システムをドームふじ基地に設営し、無人リモート観測装置を立ち上げることである。これに加えて JARE52 に引き続き JARE54 においても天文学的なサイト調査を実施する必要があると筆者は考えている。

そこでサイト調査として冬期の DIMM 観測及び C_T^2 の測定を提案する。本レポートによって C_T^2 の測定を行う装置の開発とその性能評価について示す。なお DIMM 観測については「DIMM Observation during Mid-winter using rEmote Facility at DOME Fuji (DOME-F) の提案」(2012 年 1 月 8 日、沖田博文) を参照されたい。

3 C_T^2 の定義と超音波風速計での測定原理

C_T^2 は温度構造定数と呼び以下の用に定義される。

$$C_T^2 \equiv \frac{< |T(x+r) - T(x)|^2 >}{r^{2/3}} \tag{1}$$

ここで時刻 t_i に観測した風速を V_i、温度を T_i として Taylor's Hypothesis を仮定すると

$$C_T^2 = \frac{1}{n-1} \sum_{i=2}^{n} \frac{|T_i - T_{i-1}|^2}{\{(t_i - t_{i-1}) \times (V_i + V_{i-1})/2\}^{2/3}} \tag{2}$$

と書きなおすことができる。よって適当な頻度・回数で風速と温度を測定することで C_T^2 が計算できる。Taylor's Hypothesis が良く成り立つためには頻度つまり測定周波数を大きくすればよい。

4 Applied Technologies, Inc. #SATI-3SX

Applied Technologies, Inc. #SATI-3SX は超音波風速計である。原理は送受信機間で超音波の送受信を行い所要時間から風速と音速を計算する装置である。特徴は物理的な動作部分がないため無風状態でも風速を測ることが出来る点、空気の温度そのものを非接触で測定することが出来る点が挙げられる。Operators Manual に記載されているスペックのうち今回の用途に必要なものを抜粋して表 1 に示す。また RS232C の設定値を表 2 に示す。

測定データは#SATI-3SX を電源に接続し RS232C ケーブルを PC につなぐだけでそのまま得られる。#SATI-3SX の重要かつ必要最低限の設定を表 3 に示す。この設定で風速・温度が出力される。またサンプリング周波数は 200Hz、読み出し周波数は 20Hz となる。なおこれらは Windows PC(Windows 7 Professional) からターミナルソフト Tera Term[*1]を用いて設定した。

5 USB シリアル変換ケーブル

#SATI-3SX の PC との接続は RS232C ストレートケーブルによるシリアル通信で行う。ここで実験に用いた Linux PC(Ubuntu 10.10 kernel 2.6) 及び Windows PC には RS232C コネクタを搭載していなかったため USB シリアル変換ケーブルを用いて実験を行った。USB シリアル変換ケールには秋月電子通商取り扱いの USB シリアル変換ケーブル [グレー色] を用いた。入手が容易で 900 円と安価であったのが選定の理由である。なお変換チップは Prolific PL2303 が使用されている。Prolific のウェブサイト[*2] で最新の Windows 用ドライバが入手出来る。また Linux (kernel 2.6 から?) の場合は特に設定をしなくても USB に差すと `/dev/ttyUSB0` もしくは `/dev/ttyUSB1` として認識される。

6 Linux PC からの読み出し

#SATI-3SX から出力されるデータを Linux PC で読み込むためのソフトを C 言語で開発した。引数 1 に

[*1] http://sourceforge.jp/projects/ttssh2/
[*2] http://www.prolific.com.tw/eng/downloads.asp?ID=31

Measurement Range:	
Wind Velocity	±30 m/sec
Temperature	-40 °C to +60 °C
Path Length	15 cm
Accuracy:	
Wind Speed	±0.03 m/sec
Temperature(Absolute)	±2 °
Sonic Temperature	±0.1 °
Resolution:	
Wind Speed	0.01 m/sec
Temperature	0.01 °
Sonic Temperature	±0.1 °
Output:	
Data Rate	10 samples /sec (nominal)
Digital	Serial RS-232C compatible
BAUD Rate	9600 or 115,200
Others:	
Operating Temperature Range	-40 °C to +60 °C
Power Requirements	+12VDC (+9 to +18 VDC@100mA)

表 1　#SATI-3SX スペック一覧

Baud Rate	9600
Parity	EVEN
Number of Data Bits	7
Number of Stop Bits	1
Flow Control	NONE

表 2　RS232C 設定値

Baud Rate	9600
ASCII Output	ON
Temperature	ON
Sampling per Output	1
Ticks per Sampling	0

表 3　#SATI-3SX 設定値。この設定で風速・温度が出力される。またサンプリング周波数 200Hz、読み出し周波数 20Hz となる。

秒単位で測定時間を指定し実行すると標準出力に UNIX time(μsec 単位), U,V,W(風速), T(温度) を出力する。このソースを以下に示す。

```
/* rs232c_sonic.c */

#include <stdio.h>
#include <termios.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/time.h>

int main(int argc, char *argv[]){
    struct termios tio;
    int fd;
    FILE *fpp;
    char callback[256];
    double now=0.0, start=0.0;
    struct timeval t, s;

    fd=open("/dev/ttyUSB0",O_RDWR | O_NOCTTY | O_NDELAY);
    fpp=fopen("/dev/ttyUSB0","r");
    if(fd<0){
        printf("Error: Cannot open the Serial Device\n");
        return -1;
    }

    tio.c_cflag = B9600 | CS7 | PARENB | CREAD | CLOCAL;
    tio.c_iflag = 0;
    tio.c_oflag = 0;
    tio.c_lflag = 0;
    tio.c_cc[VMIN] = 0;
    tio.c_cc[VTIME] = 1;
    tcsetattr(fd, TCSANOW, &tio);

    gettimeofday(&s,NULL);
    start=s.tv_sec+s.tv_usec/1000000.0;
    while(1){
        fgets(callback,sizeof(callback),fpp);
        gettimeofday(&t,NULL);
        now=t.tv_sec+t.tv_usec/1000000.0;
```

```
        printf("%lf  %s",now,callback);
        if((now-start) > atoi(argv[1])) break;
    }

    close(fd);
    fclose(fpp);
    return 0;
}
```

コンパイルと実行 (例) 及び実行結果 (例) を以下に示す。

```
$ gcc -lm -o sonic rs232c_sonic.c
$ ./sonic 10 > hoge
$ less hoge
$ 1329715022.172777  .98  C 334.29
$ 1329715022.220800  U 00.46  V-00.95  W-00.77  T 03.95  C 334.27
$ 1329715022.265824  U-00.05  V-00.59  W-00.84  T 03.95  C 334.27
$ 1329715022.316793  U-00.86  V-00.04  W-01.24  T 04.01  C 334.31
$ ・・・・
```

この実行結果 (例) のように得られたデータ (raw データ) には文字が含まれていたり 1 列目が欠落していたりするので sed コマンドで U,V,W,T,C の文字を消し、1~3 行目を削除する処理を行う。

```
$ sed -e "s/U//g" hoge | sed -e "s/V//g" | sed -e "s/W//g" | \
  sed -e "s/T//g" | sed -e "s/C//g" | sed '1,3d' > hogehoge
```

なお Ubuntu のバグ (?) なのか /dev/ttyUSB0 の設定がうまく Linux PC に伝わらない事がある。そこで上記ソフトを実行する前に

```
$stty -F /dev/ttyUSB0 clocal
```

と入力して予め /dev/ttyUSB0 の設定を変更しておくとうまく実行できる。以降本レポートでは Linux PC から#SATI-3SX からデータを受信するソフトを sonic と呼ぶ。

7 C_T^2 を計算するソフト

sonic で得られたデータから C_T^2 を計算するソフトを c 言語で開発した。引数 1 に前章で示した sed コマンドの実行結果ファイル、引数 2 に測定の回数、引数 3 に reject する測定した温度と median 温度の差の絶対値を指定すると標準出力に測定開始時の UNIX time(μsec)、風速の median (m/s)、温度の median(°C)、C_T^2 の計算に用いた測定回数、C_T^2 を出力する。ソースを以下に示す。

```
/* ct2.c */

#include <stdio.h>
```

```
#include <math.h>

#define N 600 /* 観測回数、20Hz x 30sec = 600（暫定値）*/
#define X 1.0 /* reject する|測定した温度 - median 温度|、1 度（暫定値）*/


int main(int argc, char *argv[]){

    FILE *fi;
    int i, j, m=0, n=0;
    double t[N], u[N], v[N], w[N], T[N], c[N], V[N], Tm[N], Vm[N];
    double Tmed=0.0, Vmed=0.0, hoge=0.0, CT=0.0;

fi=fopen(argv[1], "r");
    for(i=0; i<N; i++){
        fscanf(fi, "%lf%lf%lf%lf%lf%lf\n",&t[i],&u[i],&v[i],&w[i],&T[i],&c[i]);
        V[i] = sqrt(u[i]*u[i] + v[i]*v[i] + w[i]*w[i]);
        Tm[i]=T[i];
        Vm[i]=V[i];
    }
    fclose(fi);

    for(j=1; j<N; j++){
        for(i=0; i<N-j; i++){
            if(Tm[i] < Tm[i+1]){
                hoge = Tm[i];
                Tm[i] = Tm[i+1];
                Tm[i+1] = hoge;
            }
            if(Vm[i] < Vm[i+1]){
                hoge = Vm[i];
                Vm[i] = Vm[i+1];
                Vm[i+1] = hoge;
            }
        }
    }
    if(N%2==1){
        Tmed = Tm[(N-1)/2];
        Vmed = Vm[(N-1)/2];
    }
```

```
    else{
        Tmed = (Tm[(N-1)/2]+Tm[(N-1)/2+1])/2.0;
        Vmed = (Vm[(N-1)/2]+Vm[(N-1)/2+1])/2.0;
    }

for(i=0;i<N;i++){
        if( pow((T[i]-Tmed),2.0) > pow(X,2.0) ){
            m=0;
        }
        else{
            if(m==0){
                m=1;
            }
            else{
          CT=CT+pow((T[i]-T[i-1]),2.0)/pow(((t[i]-t[i-1])*(V[i-1]+V[i])/2.0), 2/3.0);
                n=n+1;
            }
        }
    }
    CT=CT/n;
    printf("%lf\t%0.2lf\t%0.2lf\t%d\t%.10lf\n",t[0],Vmed,Tmed,n,CT);
    return 0;
}
```

コンパイルと実行 (例) 及び実行結果 (例) を以下に示す。

```
$ gcc -lm -o CT2 ct2.c
$ ./CT2 hogehoge
$ 1329710641.616789       2.16    2.55    599     0.0210390602
```

以降本レポートでは C_T^2 を計算するソフトを CT2 と呼ぶ。

8 自動観測シェルスクリプト

実際の C_T^2 観測は bach シェルスクリプト及び crontab によって自動で実行する。以下にシェルスクリプトのソースを示す。ここでは観測時間を 32 秒としている。

```
#!/bin/sh
#sonic2.sh

stty -F /dev/ttyUSB0 clocal
```

```
#観測時間（sec）
TIME=32
DIR=‘date -u +%Y%m%d_%H%M%S‘

echo "SONIC OBSERVATION START AT ‘date -u +%Y/%m/%d_%H:%M:%S‘ (UTC)" >> \
/home/okita/2012_02_20/sonic.log
/home/okita/2012_02_20/sonic $TIME > /home/okita/2012_02_20/RAWdata/$DIR.sonic
echo "SONIC OBSERVATION FINISH AT ‘date -u +%Y/%m/%d_%H:%M:%S‘ (UTC)" >> \
/home/okita/2012_02_20/sonic.log

echo "    START CT2 CALCULATE AT ‘date -u +%Y/%m/%d_%H:%M:%S‘ (UTC)" >> \
/home/okita/2012_02_20/sonic.log
sed -e "s/U//g" /home/okita/2012_02_20/RAWdata/$DIR.sonic | sed -e "s/V//g" | \
sed -e "s/W//g" | sed -e "s/T//g" | sed -e "s/C//g" | sed ’1,3d’ > \
/home/okita/2012_02_20/hoge
/home/okita/2012_02_20/CT2 /home/okita/2012_02_20/hoge >> \
/home/okita/2012_02_20/result.ct2
rm -f /home/okita/2012_02_20/hoge
echo "    FINISH CT2 CALCULATE AT ‘date -u +%Y/%m/%d_%H:%M:%S‘ (UTC)" >> \
/home/okita/2012_02_20/sonic.log

exit 0
```

シェルを実行可能にするため

```
$ chmod +x sonic2.sh
```

を実行する必要がある。また crontub を以下に設定し 1 分ごとに sonic2.sh を実行するよう設定した。

```
$ crontab -e
0-59 * * * * /home/okita/2012_02_20/sonic2.sh > /dev/null 2>&1
```

これで RAWdata ディレクトリに raw データ、result.ct2 ファイルに計算した C_T^2 が自動的に保存されるよう設定できたことになる。

9 #SATI-3SX の校正

マニュアルの手順に従い#SATI-3SX の校正を行った。今回の校正ではマニュアルに示されていた付属の “Zero Air Chamber” を使用せず、ダンボールで自作した “Zero Air Chamber” を使用した。理由は付属の “Zero Air Chamber” では#SATI-3SX 全体を覆うことが出来ず、次節で述べる測定値の検証のときに温度の測定誤差を求める事が出来ないからである。

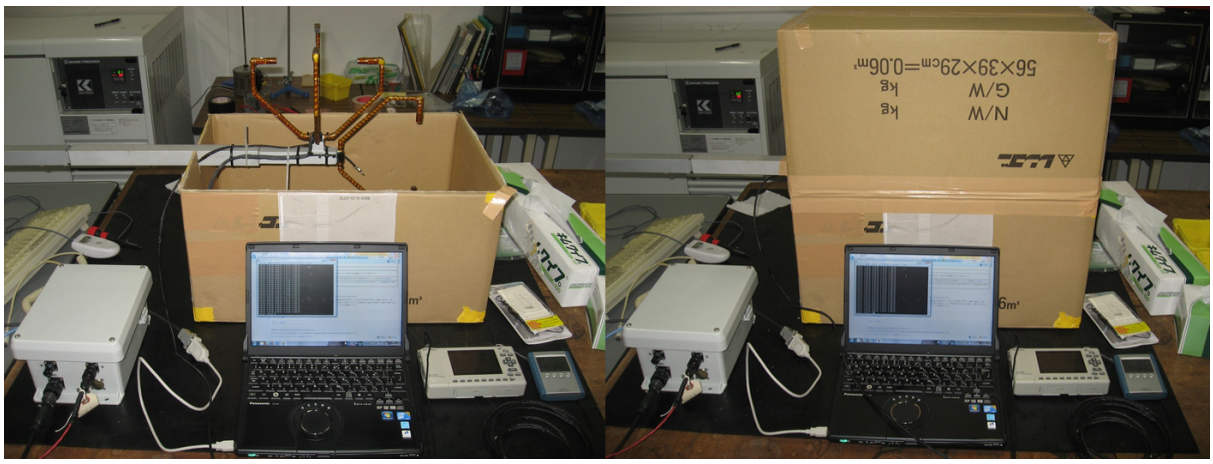

図 1　ダンボールで自作した “Zero Air Chamber”

9.1　校正方法

中ダンボール 2 個で#SATI-3SX を完全に被い、隙間をガムテープで埋め、完全な無風状態を作る。図 1 は本校正のためにダンボールで自作した “Zero Air Chamber” である。マニュアルによると校正には温度と湿度の情報が必要なため、温度はキーエンス (株)NR-1000 データロガーに Pt1000、湿度はエンペック気象計 (株) デジコンフォ II を用いて測定し、校正に使用した。測定精度はそれぞれ ±0.1% of rdg + 0.5°、±7%RH である。無風かつ温度一定を実現するため、本校正は実験室において深夜に行った。温度むらが懸念されるため実験装置は十分に室温に均しておいた。また騒音によってダンボールが振動すると校正がうまく行かないと考えられるので、可能な限り音の発生を抑えて校正を行った。

9.2　校正

2012 年 1 月 30 日 3:45JST に光学暗室にて校正を行った。校正時の温度は 13.6 ±0.5 °C、湿度は 27 ±7 % であった。

10　#SATI-3SX の測定値の検証

前節の校正を終えた後、そのまま測定値の検証実験を行った。

10.1　実験原理

無風かつ温度が一定であれば、装置由来のノイズがなければ風速と温度の標準偏差は 0 となることが期待される。よって無風かつ温度一定の環境で#SATI-3SX による測定を行って得られた標準偏差が 0 でなければこれは装置由来のノイズと言うことが出来る。

ΔU[m/s]	ΔV[m/s]	ΔW[m/s]	ΔT[°C]
1.11E-2	1.28E-2	8.52E-3	1.33E-2
7.54E-3	1.00E-2	1.01E-2	1.58E-2
3.63E-3	6.90E-3	9.38E-3	1.43E-2
2.44E-3	4.39E-3	7.04E-3	1.07E-2
4.31E-3	1.87E-3	7.37E-3	1.13E-2
8.78E-3	8.00E-4	9.51E-3	1.43E-2
1.10E-2	0.00E+0	9.91E-3	1.49E-2
1.42E-2	5.65E-4	1.03E-2	1.54E-2
1.40E-2	1.13E-3	1.03E-2	1.57E-2
1.24E-2	1.69E-3	1.05E-2	1.60E-2

表 4　無風・温度一定環境で得られた#SATI-3SX の測定値の標準偏差

10.2　実験方法

前節で用いた自作の “Zero Air Chamber” で#SATI-3SX を完全に被って無風・温度一定の環境を作り、その中で#SATI-3SX による風速及び温度の測定を行う。60 秒間の測定を 10 回行い測定後に風速・温度の標準偏差を計算する。

10.3　実験

2012 年 1 月 31 日 13:20JST から光学暗室にてデータ取得を行った。

10.4　結果

得られた風速 U,V,W[m/s]、温度 T[°C] の標準偏差を表 4 に示す

10.5　考察

表 4 から#SATI-3SX の測定値の標準偏差は 0 ではなく装置由来のノイズが存在する事がわかる。10 回の測定結果には大きな違いが見られないことから測定の結果は妥当である。ここで得られた標準偏差の最大値を装置由来のノイズ、つまり測定誤差と定義すると風速で ±0.0142[m/s]、温度で ±0.0160[°C] の誤差が測定に含まれていると言える。この値はカタログ値の精度 (風速 ±0.03[m/s], 温度 ±0.1[°C]) よりも小さい。ここから平均風速 5.8[m/s] のドームふじ基地での装置固有の誤差を見積もると $\Delta C_T^2 \sim \pm 0.0005[\mathrm{K^2 m^{-2/3}}]$ 程度となった。南極での接地境界層における C_T^2 は 0.01∼0.001$[\mathrm{K^2 m^{-2/3}}]$ 程度と予想されるため、この#SATI-3SX は十分な測定精度を有していると言える。

11 まとめ

Applied Technologies, Inc. の Sonic Anemometer/Thermometer Model #SATI-3SX 超音波風速計を用いた C_T^2 測定装置の開発を行った。Linux から制御するための各種ソフトを作成した。#SATI-3SX の校正を自作の “Zero Air Chamber” を用いて常温で行った。装置固有の測定誤差は風速 ±0.0142[m/s]、温度 ±0.0160[°C] であり観測に必要十分な精度であることがわかった。