

気象観測データの解析

2 高層気象観測データの解析

今回は、AMeDAS データと同様にして、自分で FORTRAN または C でプログラムを作成し、CD-ROM に入った高層気象観測データを読みこんで解析します。今回もデータを読みこむためのサブルーチンや関数はあらかじめ作成してあります。まず、サンプルプログラムを実行してみます。具体的には、

```
/home/snaoki/aero> f77 sample.f aero_sub.f
```

とします。sample.f がプログラムの本体で、その中で使われているデータの読みこみ用のサブルーチンは aero_sub.f に書かれています。C の場合は、

```
/home/snaoki/aero> cc sample.c aero_sub.c -lm -lf2c
```

とします。aero_sub.c は FORTRAN で作成したプログラムを自動で変換して作成したため、-lf2c というオプションをつけて専用のライブラリを参照します。コンパイルが成功すると実行ファイルが生成されているはずです。生成されていることを確認したら実行します。地点番号と日時を聞かれますので、入力してください。館野（つくば）の地点番号は 47646 です。観測は 1 日に 2 回、日本時間の 9 時と 21 時に行なわれていますが、台風が接近している場合などには、3 時と 15 時にも観測を実施することがあります。

```
/home/snaoki/aero> ./a.out
Station No. ?
47646
Year, Month, Day, Hour (3, 9, 15, 21) ?
2009, 5, 13, 21
Reading data ...
Output was written to file 'output.txt'.
```

結果は output.txt というファイルに書き出されるので、cat などのコマンドで中身を確認してください。

```
/home/snaoki/aero> cat output.txt
Station No. = 47646,
Time = 2100, May 13, 2009
Pres.   GPH Temp. Hum. Dir. Speed
  hPa    m      C   % deg.  m/s
1002.7 Surf. 15.3  98 180  0.7
1000     54  16.5  79 162  1.0
 925     714  13.0  52 334  5.0
 900     944  11.9  49 301  8.0
```

<中略>

```
10 31163 -43.7 *** 273 15.0
5 ***** ***** *** *** *****
```

今回は指定気圧面（1000hPa、850hPa、700hPa など）のデータだけを書き出しています。実際の高層気象観測では、指定気圧面だけでなく、特異点といって、温度勾配や風向・風速が急に变化する高度でのデータを必要に応じて作成しています。前線面や逆転層などを詳しく解析するときには、特異点のデータも利用します。

次に、指定した地点と日時における状態曲線を描くために、気温と露点温度を出力するプログラムを作成してみます。サブルーチン AERDT は、IYY、IMM で指定した年、月について、配列 ISTA で指定したすべての観測地点のデータを読みこみます（以下の例では 1 地点だけのデータを読みこんでいます）。観測地点の一覧表は index.txt というファイルで与えられています。サブルーチン AERDT は、aero_sub.f の中に書いてあるので、コンパイルのときに、このファイル名を指定すれば、自分でサブルーチンを作る必要はありません。

露点温度の計算には、飽和水蒸気圧の近似式（テテンの式）

$$e_s = 611 \exp\left(17.27 \frac{T - 273.16}{T - 35.86}\right)$$

を用います。両辺の対数をとると、

$$\frac{T - 273.16}{T - 35.86} = \frac{1}{17.27} \ln\left(\frac{e_s}{611}\right)$$

となるので、

$$1 - \frac{237.3}{T - 35.86} = \frac{1}{17.27} \ln\left(\frac{e_s}{611}\right)$$

と変形できて、

$$T = 35.86 + \frac{237.3}{1 - \frac{1}{17.27} \ln\left(\frac{e_s}{611}\right)}$$

が得られます。ここで、左辺の温度 T を露点温度 T_{dew} に置き換え、 e_s にテテンの式を代入すると、

$$T_{\text{dew}} = 35.86 + \frac{237.3}{1 - \frac{T - 273.16}{T - 35.86} - \frac{1}{17.27} \ln(h)}$$

となって（ h は相対湿度）、露点温度 T_{dew} の計算式として、

$$T_{\text{dew}} = 35.86 + \frac{T - 35.86}{1 - \frac{T - 35.86}{237.3} \frac{\ln h}{17.27}}$$

が得られます。

プログラム：

```

C 配列の大きさ: IHXは1日の観測回数(=4)、IDXは1月の日数(=31)、
C              ILXは気圧面の数(=26)、NXは地点数(=1)である。
C              PARAMETER (IHX=4, IDX=31, ILX=26, NX=1)

C 配列(入力データ)の宣言:
C  ISTA (NX): 観測地点番号。館野(つくば)は47464など。
C  ICHECK (IHX, IDX, NX): データが存在する場合は1、
C                          存在しない場合は0。
C  IZ (IHX, IDX, ILX, , NX): ジオポテンシャル高度 [m]。
C  IU (IHX, IDX, ILX, , NX): 相対湿度 [%]。
C  ID (IHX, IDX, ILX, , NX): 風向 [°、北=0、東=90]。
C  P (IHX, IDX, ILX, , NX): 気圧 [hPa]。
C  T (IHX, IDX, ILX, , NX): 気温 [°C]。
C  S (IHX, IDX, ILX, , NX): 風速 [m/s]。
C              INTEGER ISTA(NX), ICHECK(IHX, IDX, NX),
C              +      IZ(IHX, IDX, ILX, NX), IU(IHX, IDX, ILX, NX),
C              +      ID(IHX, IDX, ILX, NX)
C              REAL P(IHX, IDX, ILX, NX), T(IHX, IDX, ILX, NX),
C              +      S(IHX, IDX, ILX, NX)

C 配列(出力データ)の宣言:
C  TD (IHX, IDX, ILX, NX): 露点温度 [°C]。
C              REAL TD(IHX, IDX, ILX, NX)

C 欠損値: データが無効のとき、IMISS、RMISSで指定された値が格納される。
C          整数型の場合はIMISS、実数型の場合はRMISSである。
C          PARAMETER (IMISS=999999, RMISS=1.0E9)

C 地点番号を指定する。
C 入力を求めるメッセージを標準出力に書き出す。
C          WRITE(6,*) 'Station No. ?'

C READ文で入力値を読みこむ。
C          READ (5,*) ISTA(1)

C 日時を指定する。
C 入力を求めるメッセージを標準出力に書き出す。
C          WRITE(6,*) 'Year, Month, Day, Hour (3, 9, 15, 21) ?'

```

C READ文で入力値を読みこむ。

```
READ (5,*) IYY, IMM, IDD, IHH
```

```
IHH = (IHH + 3) / 6
```

C サブルーチンAERDTによって、

C 指定された年、月の観測データを読むこむ。

C ※サブルーチンAERDTは、IYY、IMMで指定された年、月について

C データを読みこむ。

```
CALL AERDT
```

```
+ (IHX, IDX, ILX, NX, ISTA, IYY, IMM, IMISS, RMISS,
```

```
+ ICHECK, P, IZ, T, IU, ID, S)
```

C ここからDOループ11~14が始まる。

C 各時刻の各気圧面について、露点温度を計算する。

```
DO 11 N=1, NX
```

```
DO 12 IL=1, ILX
```

```
DO 13 IDDD=1, IDX
```

```
DO 14 IHHH=1, IHX
```

C 出力データを格納する配列に、あらかじめ欠損値を入れておく。

```
TD(IHHH, IDDD, IL, N) = RMISS
```

C データが存在する場合だけ、

C つまり、T(IHH, IDD, IL, N)、IU(IHH, IDD, IL, N)が欠損値でない場合だけ、

C 計算を実行する。

```
IF ((T(IHHH, IDDD, IL, N) .NE. RMISS)
```

```
+ .AND. (IU(IHHH, IDDD, IL, N) .NE. IMISS)) THEN
```

C 関数TDEWを用いて、露点温度を計算する。

C 結果をTDに代入する。

```
TD(IHHH, IDDD, IL, N) = TDEW(T(IHHH, IDDD, IL, N)+273.15,
```

```
+ 0.01*REAL(IU(IHHH, IDDD, IL, N)))
```

```
+ -273.15
```

```
ENDIF
```

C ここでDOループ11~14が終了する。

```
14 CONTINUE
```

```
13 CONTINUE
```

```
12 CONTINUE
```

11 CONTINUE

C 出力ファイルを開く。

C 機番は10以降の番号を指定する。

C STATUSは書き出しの場合は' UNKNOWN' を指定する。

C FORMはテキストファイルの場合は' FORMATTED' を指定する。

```
OPEN(10, FILE=' output. txt' , STATUS=' UNKNOWN' ,  
+     FORM=' FORMATTED' )
```

C ここからDOループ21が始まる。

C 指定気圧面について、結果を出力する。

C 地上(IL=1)の結果は出力しない。

```
DO 21 IL=2, ILX
```

C データが存在する場合だけ、つまり、P(IHH, IDD, IL, 1), T(IHH, IDD, IL, 1),

C TD(IHH, IDD, IL, 1)が欠損値でない場合だけ、

C 結果を出力する。

```
IF ((P(IHH, IDD, IL, 1). NE. RMISS)  
+   . AND. (T(IHH, IDD, IL, 1). NE. RMISS)  
+   . AND. (TD(IHH, IDD, IL, 1). NE. RMISS)) THEN
```

C 気圧、気温、露点温度を出力する。

C 関数INTで気圧の値を整数型に変換する。INTは切り捨て、NINTは四捨五入。

```
IP = INT(P(IHH, IDD, IL, 1))  
WRITE(10, ' (1X, I4, 1X, F5. 1, 1X, F5. 1)')  
+     IP, T(IHH, IDD, IL, 1), TD(IHH, IDD, IL, 1)
```

```
ENDIF
```

C ここでDOループ21が終了する。

21 CONTINUE

C ファイルを閉じる。

```
CLOSE(10)
```

```
STOP
```

```
END
```

C 主プログラムの後に、関数やサブルーチンのような副プログラムを書く。

- C 露点温度を計算するための関数。
- C 関数の種類は実数 (REAL)、関数名はTDEWとする。
- C 引数はTとHである。

```
REAL FUNCTION TDEW ( T, H)
```

- C 露点温度を計算する。
- C ALOGは対数関数である。

```
TDEW = 35.86
+      + (T-35.86) / (1. - (T-35.86) / 237.3 * ALOG (H) / 17.27)
```

- C 関数はRETURN文とEND文で終了する。
- C RETURN文が実行された時点でのTDEWの値が関数の値として返される。

```
RETURN
END
```

(参考) C

```
#include <stdio.h>
#include <math.h>

/*===== 関数プロトタイプ =====*/

/* プログラムの中で別に作成した関数を参照するときは、
   関数プロトタイプを作成する。
   関数プロトタイプは、include文と同様に、
   ソースファイルの最初にまとめて記述する。 */
int aerdt( int ihx, int idx, int ilx, int nx, int *ista, int iyy, int imm,
           int imiss, float rmiss, int *icheck,
           float *p, int *iz, float *t, int *iu, int *id, float *s );
float f_tdew( float t, float h );

/*===== 露点温度を計算するための関数 =====*/

/* 関数の種類は浮動小数点(float)型、関数名は f_tdew とする。
   引数は t、h(float型)である。 */
float f_tdew( float t, float h )
{

/* 変数を宣言する。 */
float tdew;
```

```

/* 露点温度を計算する。
   log は対数関数である。 */
tdew = 35.86
      + (t-35.86) / (1. - (t-35.86) / 237.3 * log (h) / 17.27);

/* 関数は return 文で終了する。
   変数 tdew の値が関数 f_tdew の値として返される。 */
return tdew;

}

/*===== main 関数 =====*/

/* プログラムは main 関数から実行される。 */
int main( void )
{

/* 配列の大きさ: ihmax は 1 日の観測回数(=4)、idmax は 1 月の日数(=31)、
   ilmax は気圧面の数(=26)、nmax は地点数(=1)である。 */
int ihmax=4, idmax=31, ilmax=26, nmax=1;

/* 配列(入力データ)の宣言:
   ista [nmax]: 観測地点番号。東京(大手町)は 44132 など。
   icode [ihmax, idmax, nmax]: データが存在する場合は 1、
   存在しない場合は 0。
   iz [ihmax, idmax, ilmax, nmax]: ジオポテンシャル高度 [m]。
   iu [ihmax, idmax, ilmax, nmax]: 相対湿度 [%]。
   id [ihmax, idmax, ilmax, nmax]: 風向 [°、北=0、東=90]。
   p [ihmax, idmax, ilmax, nmax]: 気圧 [hPa]。
   t [ihmax, idmax, ilmax, nmax]: 気温 [°C]。
   s [ihmax, idmax, ilmax, nmax]: 風速 [m/s]。
   配列のサイズが大きいので、静的変数(static)として宣言する。
   static では、関数が呼ばれるごとに新たにメモリを確保するのではなく、
   プログラムが終了するまで固定されたメモリを用いる。 */
static int ista[1], icode[4*31*1],
        iz[4*31*26*1], iu[4*31*26*1], id[4*31*26*1];
static float p[4*31*26*1], t[4*31*26*1], s[4*31*26*1];

/* 配列(出力データ)の宣言:
   td [ihmax, idmax, ilmax, nmax]: 露点温度 [°C]。 */
static float td[4*31*26*1];

```

```

int ihh, idd, imm, iyy, ista1, i, il, ip, status;
FILE *fp;

/* 欠損値: データが無効のとき、imiss、rmiss で指定された値が格納される。
   整数型の場合は imiss、浮動小数点型の場合は rmiss である。 */
int imiss = 999999;
float rmiss = 1.0e9;

/* 地点番号を指定する。
   入力を求めるメッセージを標準出力に書き出す。 */
printf( "%s\n", "Station No. ?" );

/* scanf 文で入力値を読みこむ。 */
scanf( "%d", &ista1 );
ista[0] = ista1;

/* 日時を指定する。
   入力を求めるメッセージを標準出力に書き出す。 */
printf( "%s\n", "Year, Month, Day, Hour (3,9,15,21) ?" );

/* scanf 文で入力値を読みこむ。 */
scanf( "%d,%d,%d,%d", &iyy, &imm, &idd, &ihh );
ihh = (ihh + 3) / 6;

/* 関数 aerdt によって、指定された年、月の観測データを読むこむ。
   ※関数 aerdt は、iyy、imm で指定された年、月について
   データを読みこむ。 */
status = aerdt (ihmax, idmax, ilmax, nmax, ista, iyy, imm, imiss, rmiss,
               icode, p, iz, t, iu, id, s);

/* ここから for ループが始まる。
   各時刻の各気圧面について、露点温度を計算する。 */
for (i=1; i<=ihmax*idmax*ilmax*nmax; i++)
{

/* 出力データを格納する配列に、あらかじめ欠損値を入れておく。 */
td[i-1] = rmiss;

/* データが存在する場合だけ、

```



```

つまり、t[i-1]、iu[i-1]がゼロでない場合だけ、
計算を実行する。*/
if (t[i-1] != rmiss && iu[i-1] != imiss) {

    td[i-1] = f_tdew( t[i-1]+273.15, 0.01*iu[i-1] ) - 273.15;

}

/* ここで for ループが終了する。 */
}

/* 出力ファイルを開く。
   モードは"w"(テキストファイルへの書き出し)を指定する。 */
fp = fopen( "output.txt", "w" );

/* ここから for ループが始まる。
   指定気圧面について、結果を出力する。
   地上(il=1)の結果は出力しない。 */
for (il=2; il<=ilmax; il++)
{

/* データが存在する場合だけ、
   つまり、p[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax]、
   t[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax]、
   td[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax]が欠損値でない場合だけ、
   結果を出力する。*/
if (p[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax] != rmiss
    && t[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax] != rmiss
    && td[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax] != rmiss)
{

/* 気圧、気温、露点温度を出力する。 */
ip = p[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax];
fprintf( fp, " %4d %5.1f %5.1f¥n",
        ip, t[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax],
        td[(ihh-1)+(idd-1)*ihmax+(il-1)*idmax*ihmax] );

}

/* ここで for ループが終了する。 */
}

```

```

}

/* ファイルを閉じる。 */
fclose( fp );

return 0;

}

```

実行例：

```

/home/snaoki/aero> f77 prog02_1.f aero_sub.f
/home/snaoki/aero> ./a.out
Station No. ?
47778
Year, Month, Day, Hour(3, 9, 15, 21) ?
2009, 5, 15, 21
/home/snaoki/aero> cat output.txt
1000 17.2 13.5
925 12.3 10.4
900 10.6 9.4
<中略>
300 -32.2 -34.5

```

※C の場合は、コンパイル時に `-lf2c` オプションと `-lm` オプションを指定する必要があります。

課題 1：福岡(47807)における 2009 年 5 月 15 日 21 時、16 日 21 時、17 日 21 時の状態曲線（気温と露点温度のグラフ）を作成し、印刷して提出せよ（ひとつの時刻につき 1 枚、合計 3 枚）。1 枚の図に気温と露点温度の両方を作図するので、線の種類を変え、凡例をつけること。状態曲線の縦軸は気圧、横軸は温度とする。縦軸は対数軸とし、上下を反転させる。縦軸の範囲は 1000hPa から 300hPa とする。比較のため、横軸の範囲は 3 枚の図の間で統一すること。

ヒント 1：gnuplot で作図する場合、第 1 要素が横軸、第 2 要素が縦軸になるので、温度を第 1 要素、気圧を第 2 要素とするようなデータファイルを作成するとよい。縦軸の範囲は、

```
gnuplot> set yrange [1000:300]
```

のように設定すれば、上下を反転させることができる。対数軸は、

```
gnuplot> set logscale y
```

のように指定する。このままだと、目盛りが 10 のべき乗の値に対してだけしか表示されないので、

```
gnuplot> set ytics (300, 500, 700, 850, 925, 1000)
```

のように目盛をつける値を明示的に指定するとよい。

ヒント 2 : gnuplot で作図する場合、1 枚のグラフに 2 つ以上のデータ（今回の場合、気温と露点温度）を作図するためには、2 つのデータファイルを同時に指定すればよい。たとえば、

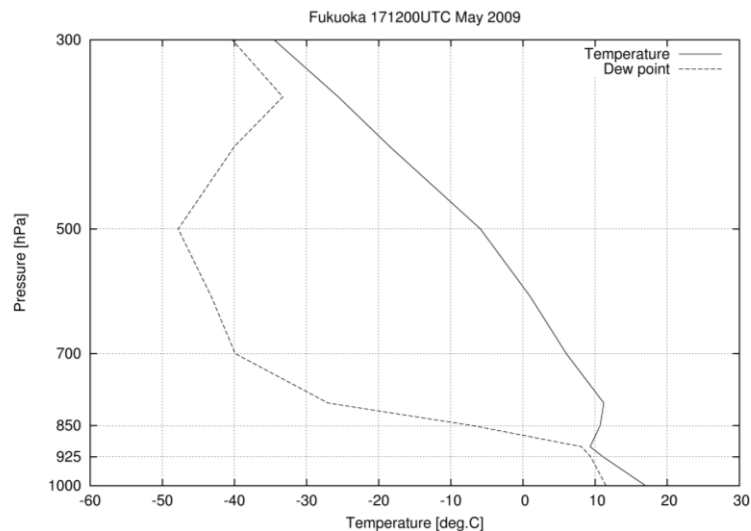
```
gnuplot> plot 'data1.txt', 'data2.txt'
```

のようにする。凡例を指定したいときは、

```
gnuplot> plot 'data1.txt' title 'Temperature', 'data2.txt' title 'Dew point'
```

のようにすればよい。線の種類は自動的に設定される。

作図例（5 月 17 日 21 時の状態曲線）：



(参考) 逆転層

温暖前線の前面では、寒気の上に暖気が乗っているため、前線面では逆転層（上層にいくほど気温が高くなっている層）が見られることがある（指定気圧面のデータのみからでは分解能が不足し、実際には生じていても見えないことがある）。これを前線逆転層というが、逆転層の上で湿潤になっているのが特徴である。一方、寒冷前線の後面や移動性高気圧付近では、上層から下降してくる空気が断熱圧縮で加熱されることによって、逆転層が生じる。これを沈降逆転層というが、逆転層の上で乾燥しているのが特徴である。

課題 2 : 札幌(47412)、松江(47741)、南大東島(47945)における 2010 年 7 月 13 日 9 時における温位[K]と相当温位[K]のグラフを作成し、印刷して提出せよ（温位のグラフを 1 枚、相当温位のグラフを 1 枚の合計 2 枚）。相当温位 θ_e は近似式

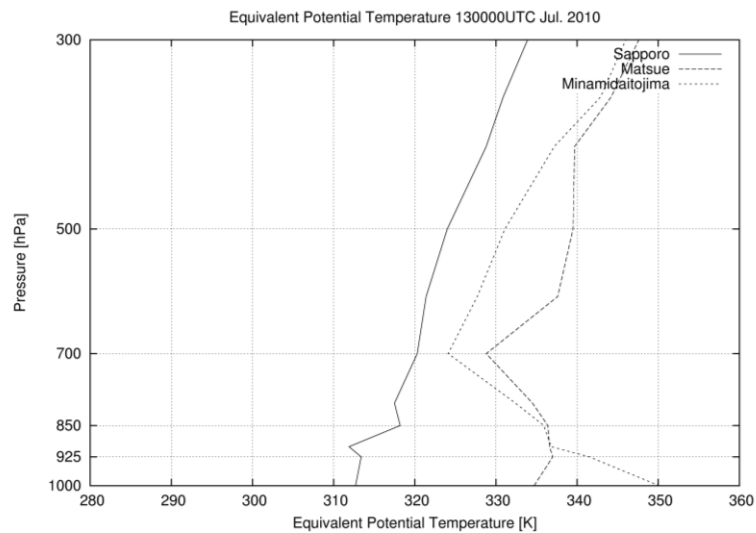
$$\theta_e = \theta + 2.6 \times 10^3 q$$

を用いて計算せよ（定義式ではないので注意すること）。ただし、 θ は温位[K]、 q は比湿[kg/kg]である。比湿は、空気に含まれる水蒸気の密度と空気全体の密度との比であり、水蒸気の分子量と乾燥空気の平均分子量との比が 0.622 であることを考慮すると、

$$q = \frac{0.622e}{(p-e)+0.622e} = \frac{0.622e}{p-0.378e}$$

と書ける。ただし、 p は気圧、 e は水蒸気圧（飽和水蒸気圧に相対湿度をかけたもの）である。1 枚の図に 3 地点のデータを作図するので、線の種類を変え、凡例をつけること。グラフの縦軸は気圧、横軸は温位または相当温位とする。縦軸は対数軸とし、上下を反転させる。縦軸の範囲は 1000hPa から 300hPa とする。比較のため、横軸の範囲は 2 枚の図の間で統一すること。

作図例（7月13日9時の相当温位）：



（参考）梅雨前線付近の成層状態

梅雨前線の南側の対流圏下層では、温暖で湿潤な空気が流入する（湿舌という）ので、相当温位が高くなっている。このため、上層に行くほど相当温位は低くなっている。このような成層状態を対流不安定というが、飽和に達しなければ不安定は顕在化しない。一方、梅雨前線の北側では、水蒸気の量が多くないので、温位と同様に上層に行くほど相当温位は高くなっている。梅雨前線付近では、水蒸気の凝結に伴って不安定が顕在化し、活発な対流活動が生じるので、空気が鉛直方向によく混合され、湿潤過程での保存量である相当温位は鉛直方向にほぼ一樣になる。このような成層状態を湿潤中立成層とよぶ。