1. 背景と目的

急速に進んでいる近代化により私たちの生活は豊かなものになってきている。しかっし、その結果地球にダメージを与えていることも事実である。その中でも近年拡大している地球規模での環境汚染に国際的な関心が高まっている。特に、地球温暖化、酸性雨問題などは行政、研究者をはじめ民間団体、企業においても重要視されており、テレビや新聞でも大々的に取り上げられて人々の関心も高まっている。このため、環境汚染物質のお発生を調べることは、地球環境問題を対処するにあたり大変重要なことである。

環境汚染の原因となっている気体は大気中に微量にしか存在していない。窒素酸化物である二酸化窒素はこの気体のひとつである。二酸化窒素は光化学スモッグや酸性雨の原因となっている汚染物質であり、人への影響については主に呼吸器系統の障害が知られている。二酸化窒素は自然生成によるものもあるが、発生源に集中しいて存在することは稀で、系全体における濃度は相対的に低い。環境の影響を与える高濃度の二酸化窒素は主に人間活動によって生成されたものであり、工場などの固定発生源や自動車などの移動発生源での高密度発生が知られている。

ここまで気体の種類に応じてさまざまな手法によりポイントでの濃度測定が行われてきた。しかし、それらはあくまでもそのポイントにおける濃度しか測定しておらず、広域にわたる濃度分布の代表値として利用することは難しい。現在問題となっている環境汚染物質ではその影響が広域にわたるものが多く、その測定方法が求められている。既存の方法は測定器周辺の気体をサンブリングして測定しているため、広域にわたる濃度を行うためには同時に複数台の設置を必要とし現実的ではない。

本研究では広域にわたって測定可能な DOAS 法に基づき得られたデータを使用し大気中の二酸化窒素の濃度を解析することを目的とする。過去に測定したデータと二酸化窒素の吸収断面積のデータを基に、DSP での解析処理を C 言語、データ読み込みと送受信を LabVIEW でプログラムを作成し解析を行った。

2. 概要

本研究ではDSP(Texus Instruments製 6713DSK)に観測データと濃度を求めたい気体の吸収断面積のデータを入力し、気体のコラム量を算出、出力するシステムを作成した。図 2.1 に構成を示す。

DSP は送信された吸収断面積、観測値のスペクトルデータをウェーブレット変換により周波数成分に分解し、それぞれのデータを最小二乗法を用いて比較することでコラム量を算出する。

また、DSP への入出力には PC を用い、データの送受信を行うプログラムを LabVIEW(National Instruments 製)で作成した。プログラムでは観測データと吸収断面積のデータをそれぞれのテキストファイルから読み込み、読み込んだデータを DSP へ送り、算出された結果を受信し、画面に表示するものを作成した。

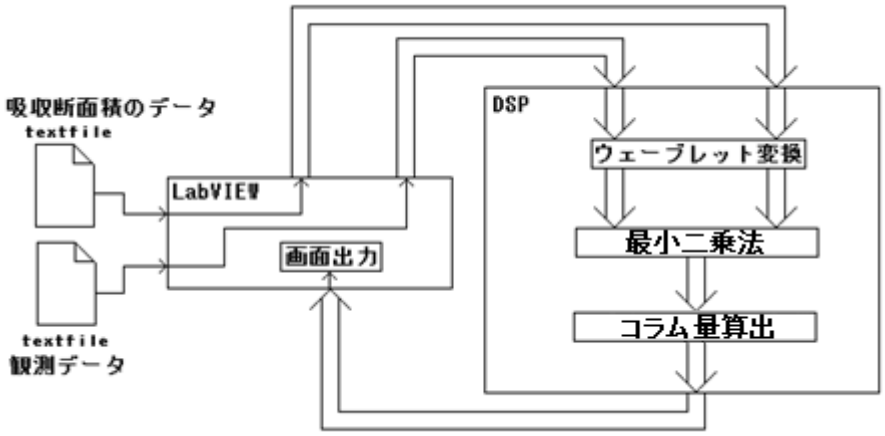


図 2.1　PC と DSP の役割

PC を分光器に替え、画面出力をデジタル表示器に変更することによって PC を使用せずにリアルタイムで濃度の観測が可能になる。

DSP のプログラムは Code Composer Studio™ IDE v3.1(Texus Instruments 製)で作成。

3. 二酸化窒素 NO2

本研究では二酸化窒素 NO_2 の濃度を解析する。ここでは、NO_2 について詳しく述べることにする。

二酸化窒素は物質が高温で燃焼する際、窒素が酸化されて生成される。しかし二酸化硫黄と違い、窒素の供給源は空気中の大気構成の中で最も多く含まれている窒素である。また、土壌での微生物による窒素化合物の分解によって発生するものや、山火事等の自然現象で生成される窒素酸化物も多く、自然発生した分だけでも大気中では 0.002～0.003ppm 程度の濃度になる。人為的な発生源は固定発生源として工業用･家庭用ボイラーや各種工業用炉(焼結炉、焼成炉、乾燥炉等)を設置した工場、そして家庭の厨房など広範囲に及ぶ。そしてさらに多いのが自動車、航空機、船舶等の移動発生源である。窒素を含む燃料の燃焼及び窒素化合物の製造のみでなく、空気を用いる燃焼のすべてが発生源となり、数ある大気汚染物質の中でも特徴的なものである。特に高圧で燃料を燃焼させる自動エンジン(特にディーゼルエンジン)は高濃度の排出源であり、モータリゼイションによる自動車台数の増加により二酸化硫黄に比べ二酸化窒素の総量は日本では減らない傾向にある。ただ、ディーゼル車を中心に排出濃度を低減した低公害車の開発も進んでいる。

二酸化窒素は他の大気汚染物資と同様に目、気管支、肺胸部の呼吸器系を刺激し障害を起こす。光科学オキシダントの一次生成物質とされており、夏季の昼間に窒素酸化物濃度の上昇が懸念される。また、硫黄酸化物と同じように大気中で酸化され硝酸塩となり、降水に溶けて酸性雨の原因となるとされている。ばい煙発生施設及び自動車エンジンの燃焼効率向上(技術革命)、発生したばい煙中の窒素酸化物の除去が二酸化窒素汚染防止の課題である。大気汚染防止法により、二酸化硫黄の場合は煙突高等より排出量が規制されており、二酸化窒素の場合は施設の種類及び規模、並びに使用燃料の種類などによって濃度による排出基準が定められている。以下に NO_2 の製法、性質について示す。

二酸化窒素は一酸化窒素が酸化されて生じる。

$$2NO + O_2 \rightarrow 2NO_2 \tag{3.1}$$

NO_2 は赤褐色、刺激臭の強い気体で有害である。自動車の排気による公害は、参加窒素 NO_2 によるものが多い。これは、エンジン中で空気が高温に熱されるため、

$N_2 + O_2 \rightarrow 2NO_2$ により生じるもので、ハイオクタンを用いるエンジンは、エンジン内の濃度も高く、それだけ酸化窒素が多くなる。これらが空気中に放出され、温度が下がると、自然に有害な NO_2 が変化する。これが公害の原因である。

NO_2 は水に溶けやすく、その水溶液が硝酸である。

$$2NO_2 + H_2O \rightleftarrows HNO_3 + HNO_2 \tag{3.2}$$

NO_2 は高温で次のように分解するため、酸化作用を示す。

$$2NO_2(\text{赤褐色}) \underset{\text{冷却}}{\overset{\text{加熱}}{\rightleftarrows}} 2NO(\text{無色}) + O_2 \tag{3.3}$$

上の反応は可逆反応で、NO を冷却すると次第に赤褐色が濃くなり、やがて完全にNO_2になる。また、NO_2をさらに冷却すると、次第に赤褐色が薄くなり、やがて無色の四酸化窒素N_2O_4になる。N_2O_4を加熱していくと、これと逆の反応をする。

4. データの流れ

本研究では DSP に観測データと濃度を求めたい気体の吸収断面積のデータを入力し、気体の濃度を算出、出力する図 4.1 のようなシステムを作成した。

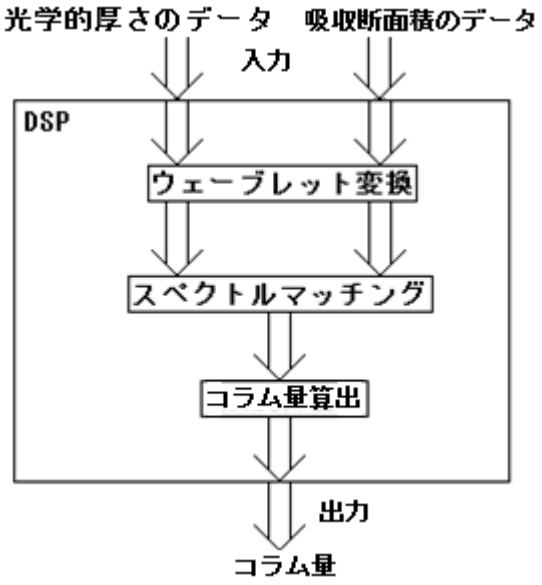


図 4.1　DSP の処理

また、このシステムを実現するために、コンピュータを用い、データの送受信を LabVIEW で再現した。LabVIEW で行うのは観測データと吸収断面積のデータをそれぞれのテキストファイルから読み込み、読み込んだデータを DSP へ送る。そして、算出された結果を受信し、画面に表示するものを作成した。かいつまんで言えば、図 4.2 のように LabVIEW はデータのパイプの役割しか行っていないということである。

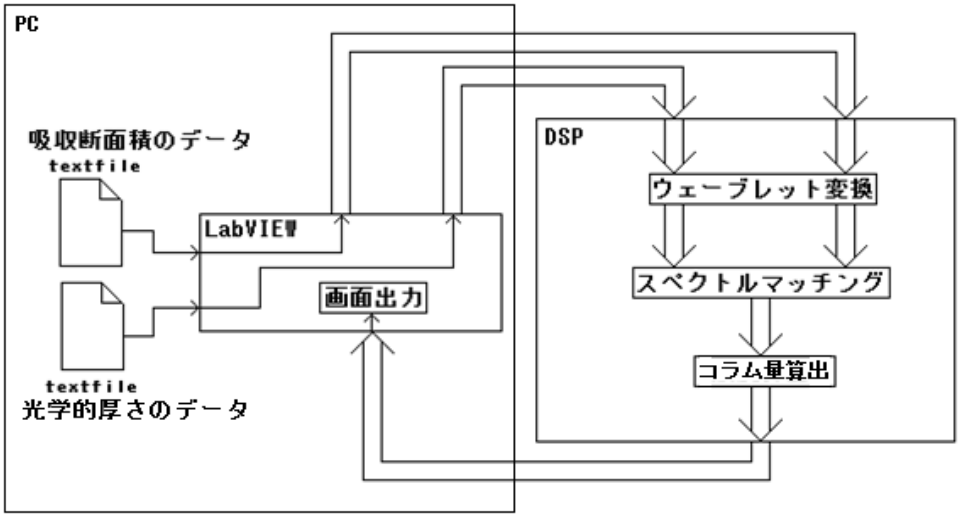


図 4.2　データの流れ

LabVIEW の部分を観測機に変え、出力をデジタル表示するものに変更することによってコンピュータを使用せずにリアルタイムで濃度の観測が可能になる。

4.1　通信

コンピュータと DSP を通信するために LabVIEW を使用した。LabVIEW ではデータの読み込み、送受信を行っている。DSP ではデータを受信して処理し、データを返している。実際に作成した vi ファイルのデータと DSP に組み込んだプログラムを以下に示す。

4.1.1　vi ファイル

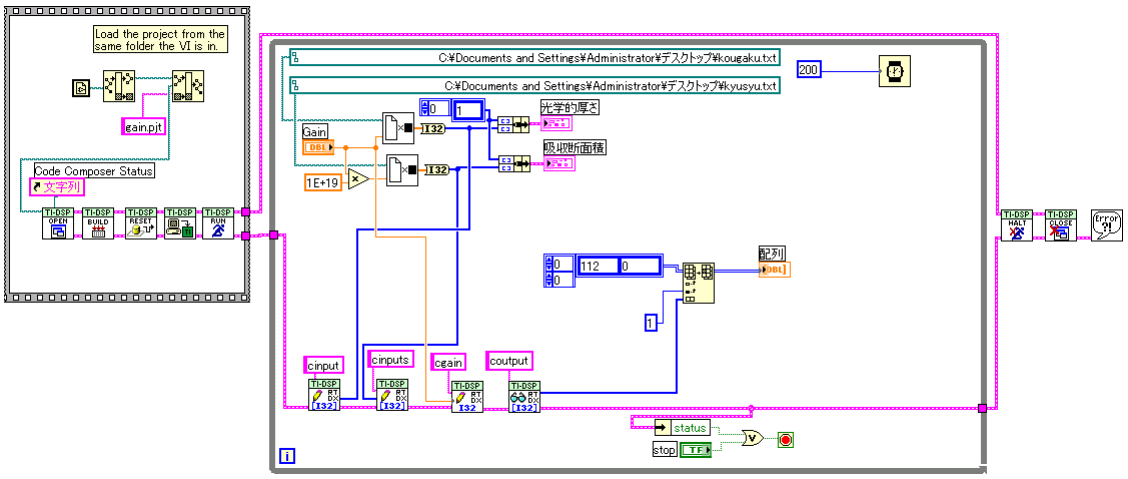


図 4.3　ブロックダイヤグラム

図 4.3 のようにブロックダイヤグラムを作成した。左のボックスでプログラムのプロジェクトを読み込み、プログラムを開き、ビルド、初期化を行い、実行している。右の while 文のボックスでは、吸収断面積、光学的厚さの txt ファイルのアドレスより読み出し、Gain を乗算している。Gain は DSP へ整数のみでしか送れないという条件より、小数であるデータを整数に変換している。そして、配列データに変換し、DSP へ転送する。この図では、光学的厚さを cinput、吸収断面積を cinputs として転送している。処理されたデータは coutput という名前で受信され、受信されたデータは配列データへ代入され、画面表示される。

4.1.2 プログラム

```
#include <rtdx.h>                       /* RTDX                         */
#include <math.h>
#include "target.h"                     /* TARGET_INITIALIZE()          */
#define SIZE 128
#define SIZE2 128
RTDX_CreateInputChannel(cinput);        //入力宣言(配列)  光学的厚さ
RTDX_CreateInputChannel(cinputs);       //入力宣言(配列)  吸収断面積
RTDX_CreateInputChannel(cgain);         //入力宣言(整数)  Gain
RTDX_CreateOutputChannel(coutput);      //出力宣言(配列)  変換後データ
//最小二乗法
float j(float a[SIZE],float b[SIZE],int rate){
        int i;
        float c=0.0;
        for(i=0;i<rate;i++){
                c+=a[i]*b[i];
        }
        return c;
}
float saij(float a[SIZE],float b[SIZE],int rate){
        float ans,x,y,xy,xx;
        xy=j(a,b,rate);
        xx=j(b,b,rate);
        ans=xy/xx;
        return ans;
  }
//最小二乗法終わり
//ウェーブレット変換(演算量削減 Ver)
float ma(float indata[SIZE],int start,int end){
        int i,k;
        float ans=0.0;
        k=end-start+1;
        for(i=0;i<k;i++){
                ans += indata[start+i];
        }
        return ans;
```

```
}
int P(int N){
		int i,ans=1;
		for(i=0;i<N;i++){
		 ans *= 2;
		}
		return ans;
}
void wave(float indata[SIZE],float outdata[SIZE]){
		int N,i,a,twin;
		float H[14];
		H[1]=(1/sqrt(2));
		H[0]=-H[1];
		for(i=2;i<8;i++){
			H[i*2-1]=H[i*2-3]*(1/sqrt(2));
			H[i*2-2]=-H[i*2-1];
		}
			a=0;
			for(N=1;N<=7;N++){
				twin=P(N);
				for(i=0;i<(128/twin);i++){
					outdata[a]=ma(indata,twin*i,twin*i+P(N-1)-1)*H[2*N-2]

+ma(indata,twin*i+P(N-1),twin*(i+1)-1)*H[2*N-1];
					a++;
				}
			}
			outdata[127]=ma(indata,0,127)*H[13];
}
//ウェーブレット変換(演算量削減 Ver)終わり



//メイン関数
void main()
{
```

```
int kou[SIZE],dan[SIZE];            //入力配列(整数)
int output[16];                     //出力配列(整数)
int i,gain = 1000;
float Fkou[SIZE],Fdan[SIZE];        //入力配列(実数)
float Wkou[SIZE],Wdan[SIZE];        //変換後の配列(実数)
float GAIN=1000.0;
// Target initialization for RTDX
TARGET_INITIALIZE();
//チャンネル宣言
RTDX_enableInput(&cgain);
RTDX_enableInput(&cinput);
RTDX_enableInput(&cinputs);
RTDX_enableOutput(&coutput);
for (;;)                    //処理内容(ループ)
{
        if (!RTDX_channelBusy(&cgain))                  //入力がある場合
                RTDX_readNB(&cgain, &gain, sizeof(gain));   //変数 gain へ代入
        while(!RTDX_read(&cinput, kou, sizeof(kou)));       //光学的厚さのデータ取得
        while(!RTDX_read(&cinputs, dan, sizeof(dan)));      //吸収断面積のデータ取得
        GAIN=gain;                                          //整数を実数に変換
        for(i=0;i<SIZE;i++){
                Fkou[i] =kou[i];                            //整数を実数に変換
                Fdan[i] =dan[i];                            //整数を実数に変換
        }
        wave(Fkou,Wkou);                        //光学的厚さに対してウェーブレット変換
        wave(Fdan,Wdan);                        //吸収断面積に対してウェーブレット変換
        for(i=0;i<16;i++){
                output[i]=0;                    //出力初期化
        }
        for(i=0;i<16;i++){
            output[i]=saij(Wkou,Wdan,112+i)*GAIN;   //最小二乗法の結果(コラム量)を出力へ代入
        }
        RTDX_write(&coutput, &output, sizeof(output));      //コラム量(配列)を外部へ出力
}
}
```

データを受け取り、ウェーブレット変換、最小二乗法によりコラム量を算出し、LabVIEW へ転送している。出力結果は最小二乗法で比較するデータ数を変えたもので、始めのデータから 112 個目までから 128 個目までの合計 16 個のコラム量を算出している。これは、最終スケール以外に波長が長い部分を比較対象にしなかった場合、コラム量にどんな影響が出てくるか確認するためである。

また、プログラムを組んで実行する際に多くのエラーが出た。その原因として、DSP のメモリ制限のせいか、出力するチャンネルは 2 チャンネル以上使用できなく、1 チャンネルで使用した場合、約 250 個までの配列データを転送できることが分かった。

5. LabVIEW でのウェーブレット変換

ウェーブレット変換のアルゴリズムを DSP へ組み込むためにアルゴリズムを十分理解する必要がある。そのため、DSP へ組み込む前に LabVIEW にてウェーブレット変換を行った。そこで、演算量を削減したアルゴリズムを考案し、考えをまとめるために上記のような C 言語で記述し、関数をパーツ化などした。アルゴリズムについては後に記してある。

作成したのは以下の仕様のものである。吸収断面積、光学的厚さ、変換マトリクスの変数を読み込み、ウェーブレット変換を行う。また、変換した結果を画面表示する。

アルゴリズムに必要な vi ファイルを作成した。それは以下の 3 個である。

・ファイルからの列読み出し

入力 ： ファイルパス(text ファイル)、列数(整数)

出力 ： 指定された列のデータ(配列)

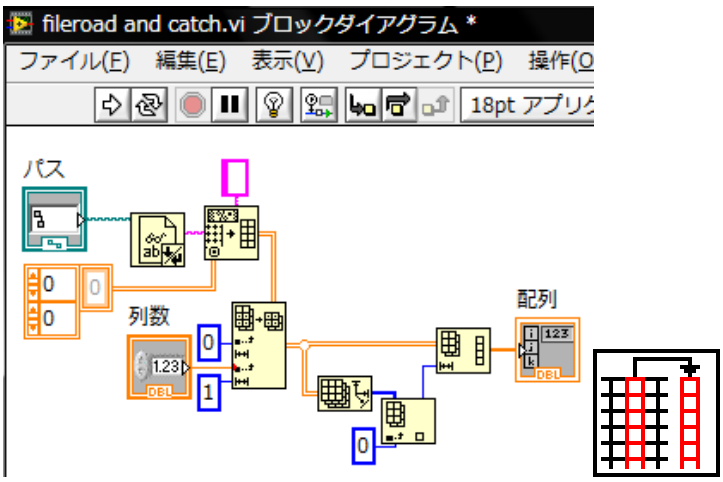


図 5.1　ファイルからの列読み出し

Text ファイルのパスからデータを読み込み、行列のデータへ変換する。行列のデータから列数の指定によりその列のデータのみを引き出し、配列データとして出力する。この vi ファイルをパーツ、アイコン化し、図 5.1 の右図のようにした。

・ファイルから行列指定し、読み込み

入力 ： ファイルパス(text ファイル)、行(整数)、列(整数)

出力 ： 指定された行列の数(実数)

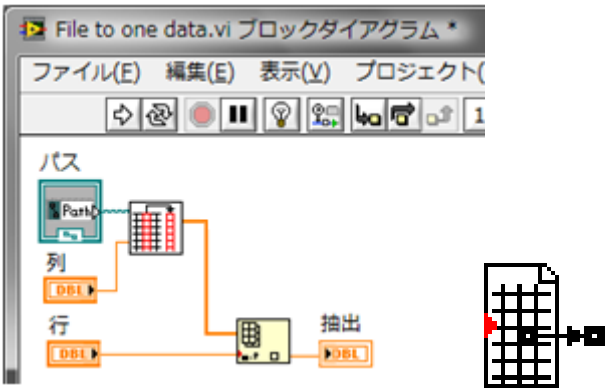


図 5.2　ファイルから行列指定し、読み込み

テキストファイルを読み込み、指定した列を配列に変換する。求められた配列データより行の値によりひとつのデータを抽出する。この vi ファイルをパーツ、アイコン化し、図 5.2 の右図のようにした。

・配列データより任意の範囲の合計を求める

入力 ： ファイルパス(text ファイル)、計算範囲の開始地点(整数)、

　　計算範囲の終わり地点(整数)

出力 ： 範囲内の総合和(実数)

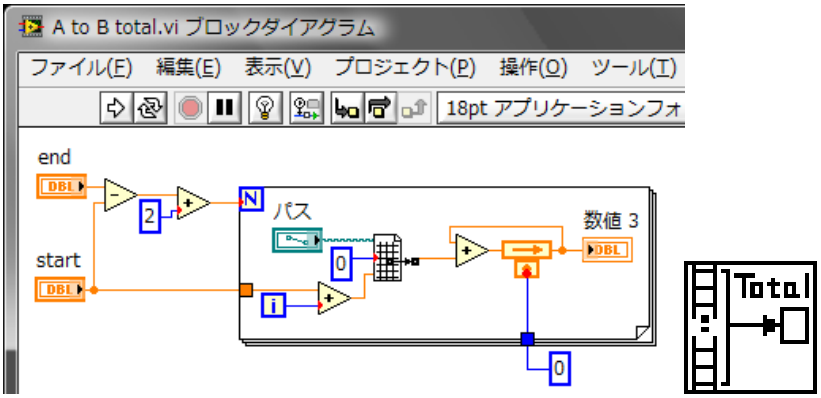


図 5.3　配列データより任意の範囲の合計を求める

Start の値を開始地点とし、start－end+2 の回数分配列のアドレスを 1 ずつシフトし、加算を繰り返す。かいつまんで言うと、配列の A 番目から B 番目までの数を全部加算するということ。これは、式 7.28 の因数分解した後の括弧内の計算を行っている。この vi ファイルをパーツ、アイコン化し、図 5.3 の右図のようにした。

5.1　ウェーブレット変換を行う

入力：変換するデータ(text ファイル)、係数のデータ(text ファイル)

出力：ウェーブレット変換後のデータ(実数の配列データ)

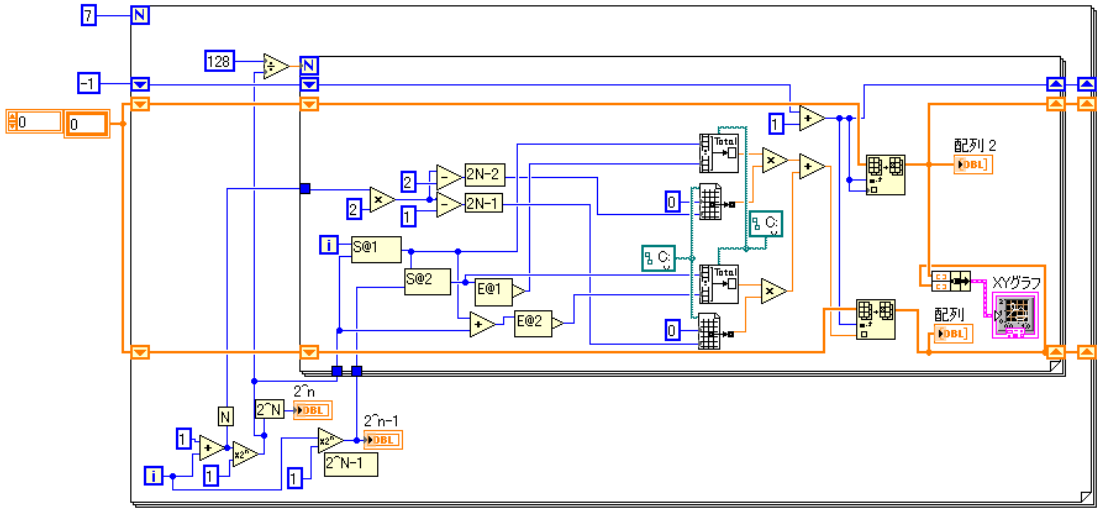


図 5.4　ウェーブレット変換のブロックダイヤグラム

外側の for 文(四角で囲んであるところ)の外では 128 個のデータの配列に 0 を代入し初期化している。また、配列の何番目に入れるかを指定するために-1 という値を設定している。N のブロックに繋がっている数字はスケール数である。8(最終)スケール目は比較しないため計算を行っていないものとする。外側の for 文と内側の for 文の外側では次のスケールへいくごとにスケールでの計算回数、合計範囲を確定するのに必要な 2 の累乗の値が変動するのでそれを算出している。内側の for 文では合計範囲の開始地点と終わり地点の計算を行い、変換するファイルよりデータを読み込みその範囲で合計を計算する。係数のファイルより読み込まれた値と乗算、加算することで変換後のデータを得ることができる。最後に得られたデータを配列へ代入し、行列変換することでグラフへ表示される。

実行した結果を図 5.5 に示す。

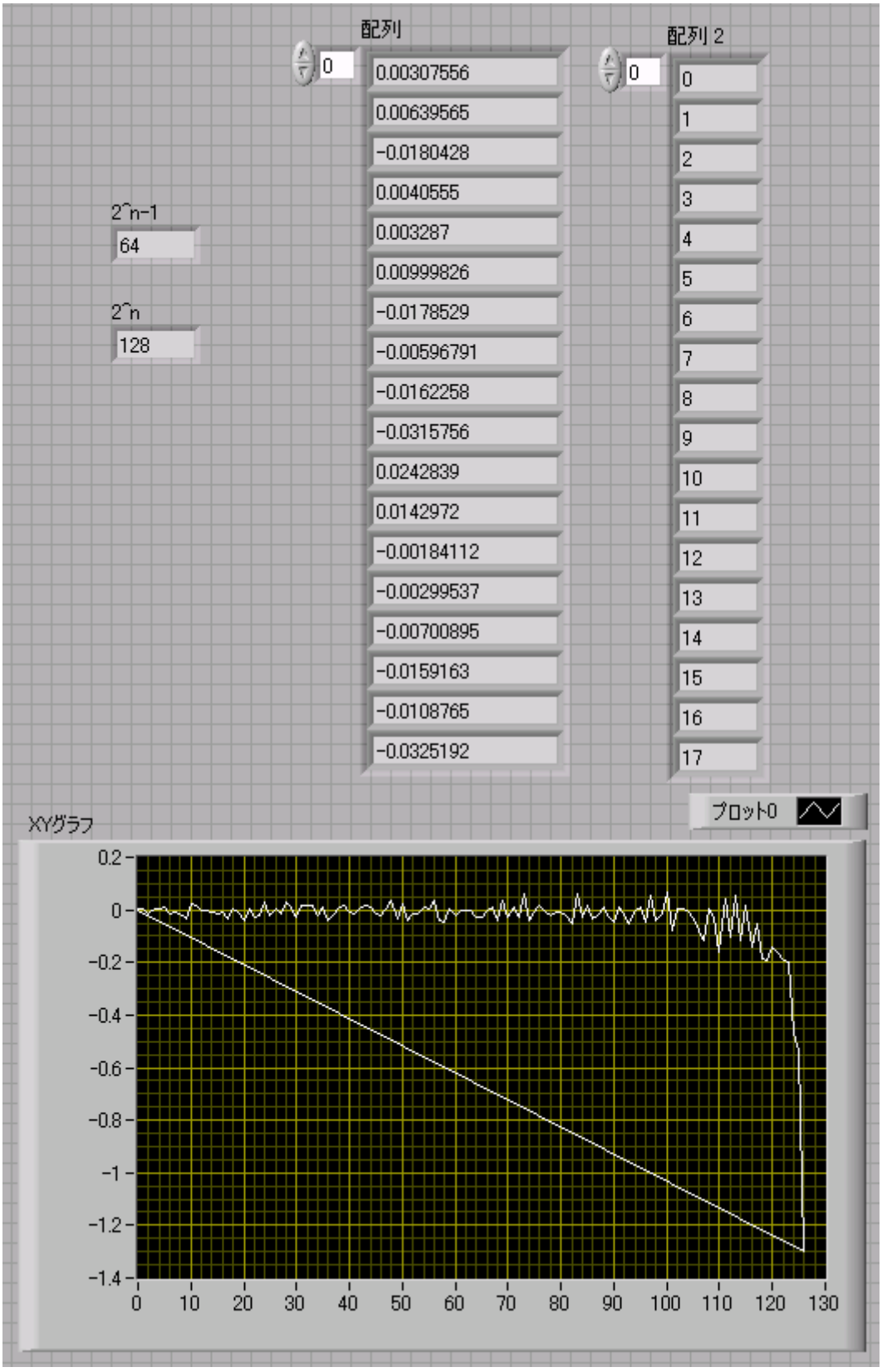

配列
0
0.00307556
0.00639565
-0.0180428
0.0040555
0.003287
0.00999826
-0.0178529
-0.00596791
-0.0162258
-0.0315756
0.0242839
0.0142972
-0.00184112
-0.00299537
-0.00700895
-0.0159163
-0.0108765
-0.0325192
配列 2
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
2^n-1
64
2^n
128
XYグラフ
プロット0
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
0
10
20
30
40
50
60
70
80
90
100
110
120
130

図 5.5　実行結果

6. 差分吸収分光法の基本式

光源から出た光は大気中でさまざまな成分の影響を受け、そのスペクトルは

$$I(\lambda) = Io(\lambda) \sum_{n=1}^{m} Tn(\lambda) \tag{6.1}$$

と表わされる。ここで、Io は光源スペクトル、Tn は各成分の透過率、m は成分の数である。ここでは成分を NO_2(以下、ガス)に限定する。観測されるスペクトル I は、吸収のある波長範囲内では次式のように示される。

$$I(\lambda) = Io(\lambda) T_g(\lambda) T_m(\lambda) T_a(\lambda) \tag{6.2}$$

ここで Tg、Tm、Ta はそれぞれのガス、空気分子、エアロゾル粒子による透過率である。また、透過率と光学的厚さはランバートビアの法則で示され、

$$T = \exp(-\tau) \tag{6.3}$$

である。

空気分子による光学的厚さは次式で示される。

$$\tau_m(\lambda) = 0.00194\lambda^{-(3.916 + 0.074\lambda + 0.05\lambda^2)} \tag{6.4}$$

また、エアロゾル粒子による光学的厚さは次式で示される。

$$\tau_a(\lambda) = B\lambda^{-A} \tag{6.5}$$

ここで、B は混濁定数(波長 1μm における光学的厚さ)、A はオングストロームパラメータと呼ばれ 1 前後の値を取る。ガスによる光学的厚さは、

$$T_g(\lambda) = \sigma(\lambda) Nl \tag{6.6}$$

となる。ここで、σ はガスの吸収断面積、N は分子数密度、l は光路長である。

よって、コラム量 Nl は、

$$Nl = \frac{\tau_g(\lambda)}{\sigma(\lambda)} \tag{6.7}$$

によって導出できる。

コラム量とは、底面積 $1m^2$ の気柱の中に含まれる分子の数である。

7. 解析手順

7.1 NO_2 の光学的厚さ τ_g の導出

NO_2 の濃度を算出するにあたって、光学的厚さ τ_g の導出が必要である。観測スペクトルと参照スペクトルを使用し、光学的厚さ τ_g の導出を行う。参照スペクトルには光源のスペクトルを使用する。まず、観測スペクトルを I_1、NO_2 の透過率を T_{g1} とすると次式のようになる。

$$I_1(\lambda) = kI_{ref}(\lambda)T_{g1}(\lambda)T_m(\lambda)T_a(\lambda) \tag{7.1}$$

次に、参照スペクトルを I_2、NO_2 の透過率を T_{g2} とすると次式のようになる。

$$I_2(\lambda) = kI_{ref}(\lambda) \tag{7.2}$$

観測スペクトルと参照スペクトルの比をとることにより、次式を導出できる。

$$\frac{I_1(\lambda)}{I_2(\lambda)} = T_{g1}(\lambda)T_m(\lambda)T_a(\lambda) \tag{7.3}$$

ここで、$T_m(\lambda)$、$T_a(\lambda)$は値が大きく上下しなく、スペクトルの値の変化を比較する本研究では結果に大きく影響しないと考えられ、バイアスと仮定する。すると、

$$\frac{I_1(\lambda)}{I_2(\lambda)} = T_{g1}(\lambda) \tag{7.4}$$

になる。

ランバート･ビアの法則より、光学的厚さ τ_g は次式のようになる。

$$\tau_g(\lambda) = -\log T_{g1}(\lambda) \tag{7.5}$$

よって、

$$\tau_g(\lambda) = -\log \frac{I_1(\lambda)}{I_2(\lambda)} \tag{7.6}$$

となる。

7.2　観測、参照スペクトル

ここでは、実際に濃度の計算を行う。観測スペクトルのデータ、参照スペクトルのデータを使用する。図 7.7 に観測スペクトル、図 7.8 に参照スペクトルを示す。

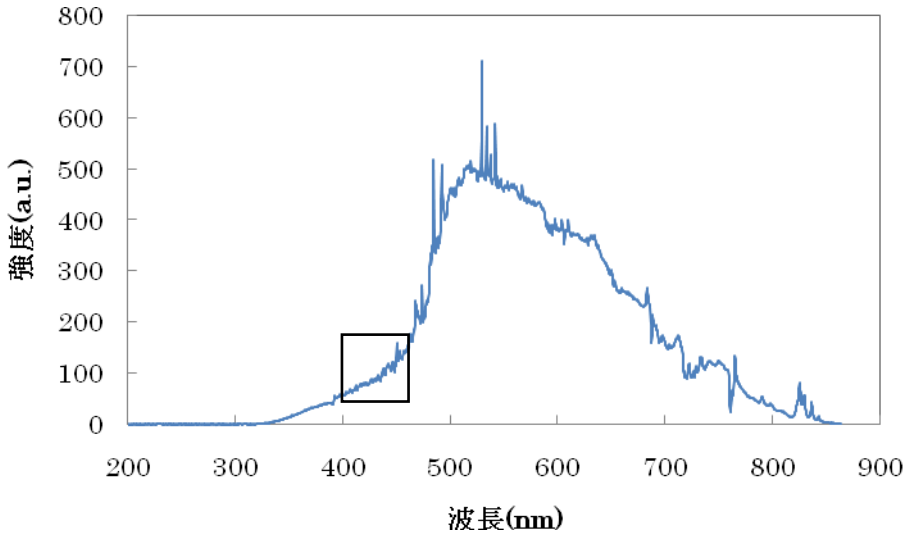


図 7.7　観測スペクトル

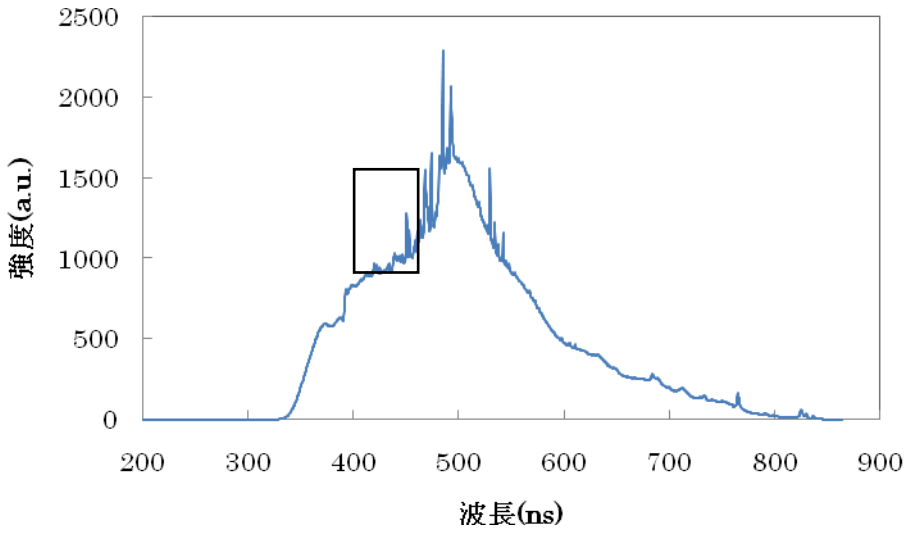


図 7.8　参照スペクトル

また、今回の波長範囲は 400nm-450nm とし、観測スペクトルと参照スペクトルを比較してみると、スペクトルの形状が酷似していることがわかる。しかし、参照スペクトルの方が観測スペクトルよりも光強度が大きい。観測スペクトルはさまざまな気体の影響を受けているので参照スペクトルに比べて強度が低くなる。

7.3　吸収断面積

吸収断面積σのデータを図 7.9 に示す。吸収断面積とは、分子一個あたりが光を吸収する度合いを断面積という概念で表わしたものである。このデータは真空中にNO_2を入れて実験したもので、吸収断面積が大きくなると式 7.4 により透過率が小さくなる。NO_2の吸収断面積では 400nm-450nm の範囲で大きく値が変動している。

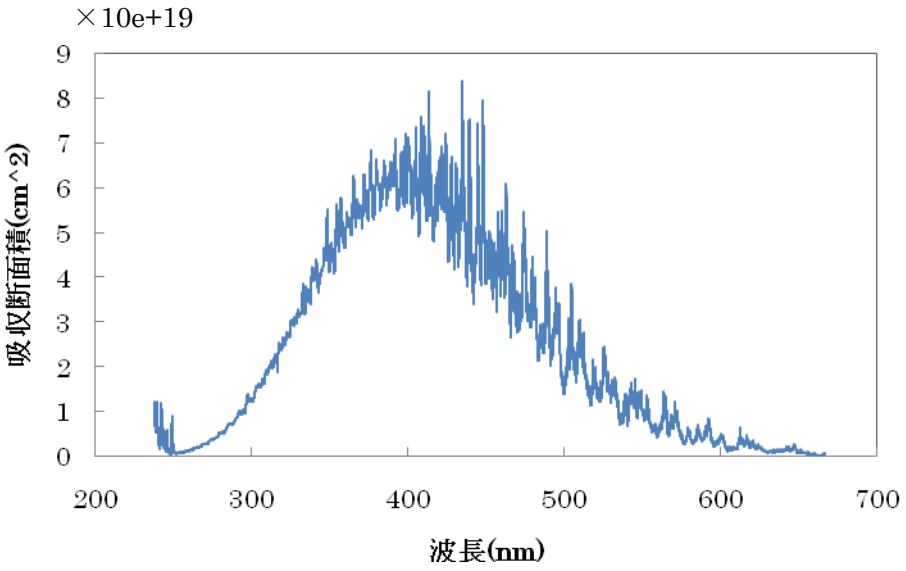


図 7.9　吸収断面積

光学的厚さτ_gの算出

式 7.6 により NO_2 の光学的厚さ τ_g を導出する。

7.4　波長の分解能の補間

濃度は光学的厚さと吸収断面積とを比較することにより算出できる。しかし、同じ波長区間においてデータ数が異なり、濃度の算出は困難である。このため、波長分解能の一致化(データの個数を合わせる)する必要がある。ここでは光学的厚さの方が吸収断面積よりもデータの数が少ないことから光学的厚さの分解能に合わせる。

以下に波長分解能の一致化の手順を示す。

1. 光学的厚さの 1 つのデータを挟む吸収断面積の 2 つのデータに注目する

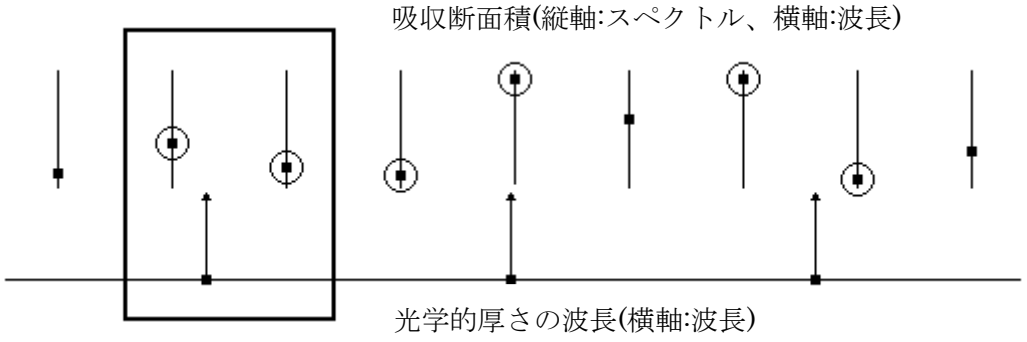
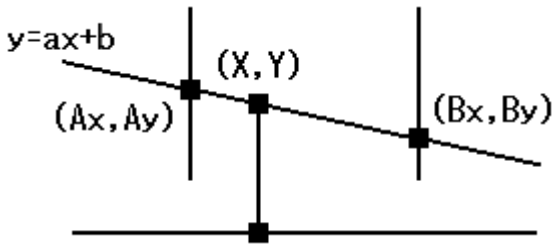


図 7.10

2. 図 7.10 の四角で囲んであるところに注目し、吸収断面積のデータそれぞれに座標を(Ax,Ay)、(Bx,By)する。このとき、2 つの点をとおる直線の傾き(a)と y 切片(b)を求める。



$$a = \frac{By - Ay}{Bx - Ax} \tag{7.11}$$

$$b = \frac{Ay - By - (Ax + Bx)a}{2} \tag{7.12}$$

3. 直線の式を求めた後、光学的厚さの分解能(X)を代入し Y を求める。
4. 各波長で同様の処理を行い、分解能を一致する。

本研究では C 言語により分解能を一致化し、そのデータを Lab VIEW で読み込んだ。

7.5　Haar のウェーブレット変換

光学的厚さと吸収断面積のスペクトルから気体濃度を求めるためには、波長ごとの特徴を比較しなければならない。波長の特徴を保存する手法は多くあるが、ここではアルゴリズムをプログラム化しやすく、変換後のデータが何を示しているかが分かりやすいという理由より正規直行マトリクスを使用した Haar のウェーブレット変換を用いる。

変換式は

$$\begin{vmatrix} W0 \\ W1 \\ W2 \\ \cdot \\ \cdot \\ \cdot \\ Wn-2 \\ Wn-1 \end{vmatrix} = \begin{vmatrix} A_{00},A_{01},\cdots,A_{0n-2},A_{0n-1} \\ A_{10},A_{11},\cdots,A_{1n-2},A_{1n-1} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ A_{n-20},A_{n-21},\cdots,A_{n-2n-2},A_{n-2n-1} \\ A_{n-10},A_{n-11},\cdots,A_{n-1n-2},A_{n-1n-1} \end{vmatrix} \begin{vmatrix} X0 \\ X1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ Xn-2 \\ Xn-1 \end{vmatrix}$$

X: 観測データ
W: 変換後のデータ
T: 変換マトリクス
（正規直行マトリクス）
n: データ数

$$T = \begin{vmatrix} A_{00},A_{01},\cdots,A_{0n-2},A_{0n-1} \\ A_{10},A_{11},\cdots,A_{1n-2},A_{1n-1} \\ \cdot \\ \cdot \\ \cdot \\ A_{n-20},A_{n-21},\cdots,A_{n-2n-2},A_{n-2n-1} \\ A_{n-10},A_{n-11},\cdots,A_{n-1n-2},A_{n-1n-1} \end{vmatrix}$$

変換式: $W=TX$
逆変換式: $X=T^{t}W$

(7.13)

となる。変換式に用いられる正規直行マトリクスの条件は以下の 2 つの式である。

$$1 = \sum_{i=0}^{n-1} A_{ji}^{2} \qquad 1 = \sum_{i=0}^{n-1} A_{ij}^{2} \tag{7.14}$$

$$j=1,2,3,\cdots,n-1 \tag{7.15}$$

$$|T| = 0 \tag{7.16}$$

また、例として 16 個のデータで比較する場合、16×16 のマトリクスが必要になる。

係数(●、■)と無効係数(0)の関係が図 7.17 のように表わされる。

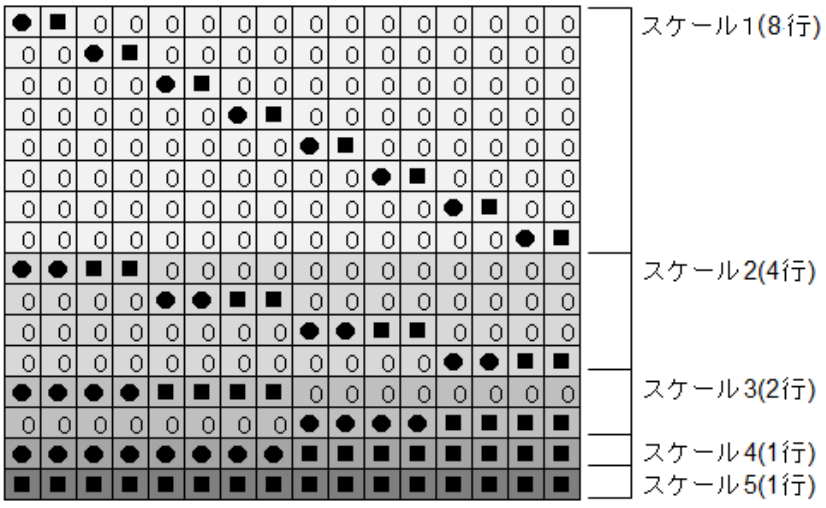


図 7.17　マトリクス(16×16)

●には負、■には正の絶対値が同じ値が入る。1 行目の係数の数を時間とし、係数をその時間に応じた大きさとすると、図 7.18 のようなパルス波形として表現できる。

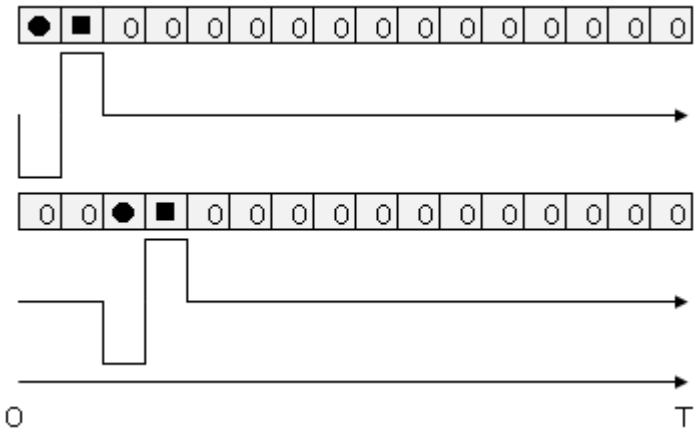


図 7.18　係数に対応するパルス波形

これを基準とする。1 周期ずつシフトすることによって変換するデータの位相差の情報を保存できる。最後までシフトすると 8 行になり、これをスケール 1 という。次に、周期を 2 倍し、同じ方法で波形情報を保存すると 4 行になる。これをスケール 2 という。最終スケール(16 行目)はすべて正の値で、直流分を表している。つまり、ウェーブレット変換を行った後、最後の値を 0 として逆変換を行った場合、直流分(全体の平均値)が除去されたデータとなる。

式　と式　の条件より条件に当てはまる係数が多数あるが、最小の周期、変換結果の理解のしやすさを考慮し、スケール 1 の係数を $1/\sqrt{2}$ とした。

データ数を n とし、スケールを s とした場合次式のようになる。

$$n = 2^{s-1} \tag{7.19}$$

このとき、式で示すようにデータ数は 2 の累乗でなければならない。

係数を k とした場合、係数とスケールの関係は次式で示される。

$$|k| = \left(\frac{1}{\sqrt{2}}\right)^s \tag{7.20}$$

行数を h とした場合、スケールに対する行数は次式で示される。

$$h = \frac{n}{2^s} \tag{7.21}$$

また、最終スケールのみ直前のスケールの正の値となり、行数は 1 となる。

本研究では 128 個のデータを比較したためスケール数は 8 となった。

7.5.1　Haar のウェーブレット変換の演算量の削減

Haar のウェーブレット変換では上記で示したように、正規直交行列が用いられる。Fを変換後のデータ、fを変換前のデータ、nをデータ数、mをデータ番号、変換行列の要素を$t_{x,y}$とした場合、変換行列を使用した場合の計算式は以下のようになる。

$$F(m) = \sum_{i=0}^{n-1}(t_{m,i} \times f(i)) \tag{7.22}$$

プログラム上の演算で、加算、乗算などを行ったときを 1 回とする。このとき、計算量を M 回とした場合、以下の式で示すことができる。

$$M = n^2 \qquad (n \neq 1) \tag{7.23}$$

演算量を削減するということは算出速度の向上が期待され、よりリアルタイムでの観測が可能になる。本研究では無効係数における演算の削減と同一係数での計算の簡略化を行った。

7.5.1.1　無効係数における演算の削減

有効係数と無効係数の関係は図 7.17 のようになる。変換行列の半数以上は無効係数である 0 で占められている。

1 行目の計算は以下のように示すことができる。

$$F(0) = t_{0,0} \times f(0) + t_{0,1} \times f(1) + 0 + \cdots + 0 = t_{0,0} \times f(0) + t_{0,1} \times f(1) \tag{7.24}$$

つまり、無効係数での計算は結果に影響しない。

無効係数での演算を削減し、計算量の削減率を R とした場合、以下の式で示すことができる。

$$R = \frac{s}{n} \tag{7.25}$$

7.5.1.2　同一係数での計算の統一化

本研究で作成した変換行列はスケールの番号が増えるにつれて周期が倍になっている。そのため、スケール 2 以降の各行に連続する同じ係数がある。16×16 の変換行列のスケール 2 での計算を例とすると以下のように示すことができる。

$$F_{(2)} = t_{2,0} \times f(0) + t_{2,1} \times f(1) + t_{2,2} \times f(2) + t_{2,3} \times f(3) + 0 + \cdots + 0 \tag{7.26}$$

ここで

$$t_{2,0} = t_{2,1} = -t_{2,2} = -t_{2,3} \tag{7.27}$$

であるから式 7.26 は次式のように示すことができる。

$$F_{(2)} = t_{2,0}(f(0) + f(1)) - t_{2,0}(f(2) + f(3)) = t_{2,0}(f(0) + f(1) - f(2) - f(3)) \tag{7.28}$$

また、このときの削減率 R は以下の式となる。

$$R = \frac{n+2}{2n} \tag{7.29}$$

この二つの手法により最終的な削減率は以下の式のようになった。

$$R = \frac{s}{n} \times \frac{n+2}{2n} = \frac{8}{128} \times \frac{128+2}{2\times128} = 0.0317 = 3.17\% \tag{7.30}$$

7.5.2　アルゴリズムのプログラム化

DSP にて濃度を算出する過程でウェーブレット変換を行う。そして、最小二乗法によりコラム量を算出する。得られたコラム量と距離を用いることにより濃度を算出することができる。

7.5.2.1　ウェーブレット変換

DSP 内で処理する内容を C 言語により記述した。演算量の削減をするようプログラムを作成した。変換マトリクスの計算の特徴としてスケールが増えるごとに係数の変化、周期の倍化に加え、行が増えるにつれ計算範囲のシフトがある。

本研究では 128 個のデータを比較した。256 個のデータを比較しようとした場合、吸収断面積の特徴が出ている波長の範囲を超えてしまう。それに加え、LabVIEW と DSP の通信にて 256 個のデータを返そうとするとエラーが表示されできなくなる。もし、256 個のデータを比較したいのであれば前半 128、後半 128 個のデータをそれぞれ変換して比較しなければならない。波長の特徴が出ている範囲は 400.21nm から 449.97nm でデータ数は 146 個である。ここで、後ろのデータを除去し 128 個とした。このときの波長の範囲は 400.21nm から 443.80nm である。

表 7.31 にスケールと係数、計算範囲、演算回数をまとめたものを記す。A,B と記述してあるものは A が負の係数、B が正の係数を示している。例えば 7 のスケールの正の係数は 13 番目、計算の範囲は 64 から 127 番目のデータとなる。

スケール N	係数 (a 番目)	合計範囲の開始地点 start	合計範囲の終わり end	演算回数
1	0,1	2i,2i+1	2i,2i+1	64
2	2,3	4i,4i+2	4i+1,4i+3	32
3	4,5	8i,8i+4	8i+3,8i+7	16
4	6,7	16i,16i+8	16i+7,16i+15	8
5	8,9	32i,32i+16	32i+15,32i+31	4
6	10,11	64i,64i+32	64i+31,64i+63	2
7	12,13	0,64	63,127	1
8	14	0	127	1

表 7.31　スケールによる計算範囲の変化

ここでスケールを N とすると係数、合計範囲の開始地点、終わり地点、演算回数の関係は以下の通りになる。

係数：$2N-2, 2N-1$

合計範囲の開始地点：$2^N1, 2^N1+2^{N-1}$

合計範囲の終わり：$2^N1+2^{N-1}-1, 2^N1+2^N-1$

演算回数：$128/2^N$

プログラム作成で大まかな計算ごとに区分けし、それぞれに対応する関数を作成した。

それは以下の３つである。

・指定された範囲の合計値を求める。

入力　配列データ(indata)、合計範囲の開始地点(start)、合計範囲の終わり(end)

戻り値　合計した値

```
float ma(float indata[SIZE],int start,int end){
        int i,k;
        float ans=0.0;                          初期値設定
        k=end-start+1;                          繰り返し回数算出
        for(i=0;i<k;i++){
                ans += indata[start+i];         順番に加算
        }
        return ans;                             算出結果を戻す
}
```

・２の累乗の値を求める。

入力　２をＮ乗するか(N)

戻り値　算出結果

```
int P(int N){
        int i,ans=1;                            初期値設定
        for(i=0;i<N;i++){                       N回繰り返し
         ans *= 2;                              2を乗算する
        }
        return ans;                             算出結果を戻す
}
```

・ウェーブレット変換後の値を代入する。

入力　変換する配列データ(indata)、変換後のデータを入れる配列(outdata)

出力　変換後のデータ(outdata)

```
void wave(float indata[SIZE],float outdata[SIZE]){
        int N,i,a,twin;
        float H[14];                                    係数配列設定
        H[1]=(1/sqrt(2));                               係数算出開始
        H[0]=-H[1];
        for(i=2;i<8;i++){
                H[i*2-1]=H[i*2-3]*(1/sqrt(2));
                H[i*2-2]=-H[i*2-1];
        }                                               係数算出終わり
                a=0;
                for(N=1;N<=7;N++){
                        twin=P(N);                      2のN乗
                        for(i=0;i<(128/twin);i++){
                        outdata[a]=ma(indata,twin*i,twin*i+P(N-1)-1)*H[2*N-2]
                                +ma(indata,twin*i+P(N-1),twin*(i+1)-1)*H[2*N-1];
                                                        変換後のデータ代入
                                a++;                    配列番号の加算
                        }
                }
                outdata[127]=ma(indata,0,127)*H[13];    最終スケール
}
```

7.6　最小二乗法

最小二乗法は、測定で得られた数値の組を、適当なモデルから想定される一次関数、対数曲線など特定の関数を用いて近似するときに、想定する関数が測定値に対してよい近似となるように、残差の二乗和を最小とするような係数を決定する方法、あるいはそのような方法によって近似を行うことである。

最小二乗法により算出される傾きの値はコラム量となる。

本研究では 1 次方程式への近似を行った。

算出方法は以下のとおりである。

$$(x, y) = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \tag{7.31}$$

という測定結果が得られたとする。求めたい一次方程式の式を

$$y = ax + b \tag{7.32}$$

とおくと、a と b は次式で求められる。

$$a = \frac{n\sum_{k=1}^{n} x_k y_k - \sum_{k=1}^{n} x_k \sum_{k=1}^{n} y_k}{n\sum_{k=1}^{n} x_k^2 - (\sum_{k=1}^{n} x_k)^2} \tag{7.33}$$

$$b = \frac{\sum_{k=1}^{n} x_k^2 \sum_{k=1}^{n} y_k - \sum_{k=1}^{n} x_k y_k \sum_{k=1}^{n} x_k}{n\sum_{k=1}^{n} x_k^2 - (\sum_{k=1}^{n} x_k)^2} \tag{7.34}$$

このとき、直流分を示す最終スケールを比較しない場合、スペクトルマッチングを行ったとき、y 切片が存在するのは不自然であるため y 切片を示す b は必要ない。

ここで求めたい一次方程式を

$$y = ax \tag{7.35}$$

と仮定した場合は以下のようになる。

$$e = \sum_{i=1}^{n} (y_i - ax_i)^2 \tag{7.36}$$

$$\frac{de}{da} = -2\sum_{i=1}^{n} (x_i y_i - ax_i^2) = 0 \tag{7.37}$$

$$a = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2} \tag{7.38}$$

e は誤差を示している。切片がある式として仮定した場合と無いと仮定した場合の結果は 0.01%の違いしかなく、プログラム化したときに演算量の削減が期待される。

最小二乗法を C 言語で実現するため２つの関数により作成した。それは以下の３つである。

・２つの異なった配列を乗算し、総和を求める。

入力　配列１(a)、配列２(b)、比較するデータ数(rate)

戻り値　総和

```
float j(float a[SIZE],float b[SIZE],int rate){
        int i;
        float c=0.0;
        for(i=0;i<rate;i++){
                c+=a[i]*b[i];
        }
        return c;
}
```

・二つの配列データより最少二乗法により傾きを求める。

入力　y 軸データ(a)、x 軸データ(b)、比較するデータ数(rate)

戻り値　傾き

```
float saij(float a[SIZE],float b[SIZE],int rate){
        float ans,x,y,xy,xx;
        xy=j(a,b,rate);
        xx=j(b,b,rate);
        ans=xy/xx;
        return ans;
}
```

8. コラム量算出

・LabVIEW

以上のアルゴリズムをもとに、LabVIEW と DSP を使用し、コラム量を算出した結果を図 8.1 に示す。このときのブロックダイヤグラム及び動作説明は 4.1 の通信にて記載されている。

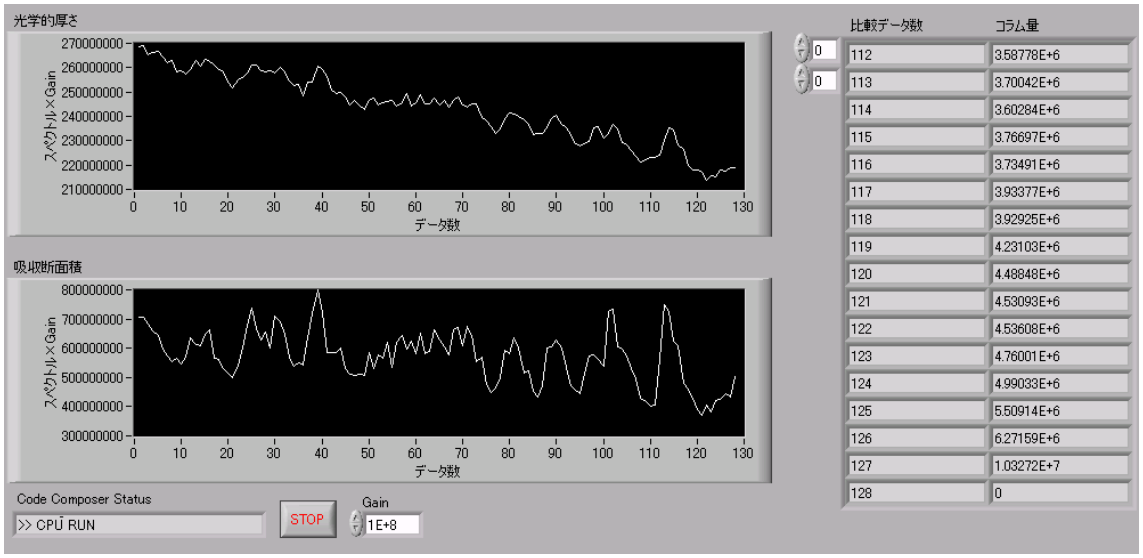


図 8.1　DSP でコラム量算出

図 8.1 の左の二つのグラフは光学的厚さ及び吸収断面積は DSP へ送信するために整数化したものである。右の表は、比較データとコラム量の関係を示している。比較データとは最小二乗法を施すデータの範囲の事を示しており、今回の場合はウェーブレット変換後のデータの後ろの方から比較対象外にしている。つまり、128 個目のデータは直流バイアス部分を表しているので、これを比較対象外にした場合、直流バイアス部分が取り除かれることになる。また、127 個目のデータは周期 1 の周波数成分となっている。比較するデータを少なくすることで低周波成分を除去でき、バイアス部分が取り除かれた結果を得ることができる。123 から 128 個目までは 5 スケール分となっている。128 個でのデータの比較は直流バイアス部を含んでいるため、正確な値が得られないことから行っていない。

LabVIEW では E+10 以降は表示ができなく、エラーとなってしまう。図 8.1 の E+6 と表示されているのは誤りで、正確には E+17 である。

同じ計算を Excel でシミュレーションを行った。その結果を表 8.2 に示す。

比較データ数	コラム量
112	3.58778E+17
113	3.70042E+17
114	3.60284E+17
115	3.76697E+17
116	3.73491E+17
117	3.93377E+17
118	3.92924E+17
119	4.23103E+17
120	4.48848E+17
121	4.53093E+17
122	4.53607E+17
123	4.76001E+17
124	4.99033E+17
125	5.50914E+17
126	6.27159E+17
127	1.03272E+18

表 8.2　Excel でのシミュレーション

図 8.1 の結果と比較すると同じ結果になった。

図 8.1 の結果をグラフとして図 8.3 に示す。

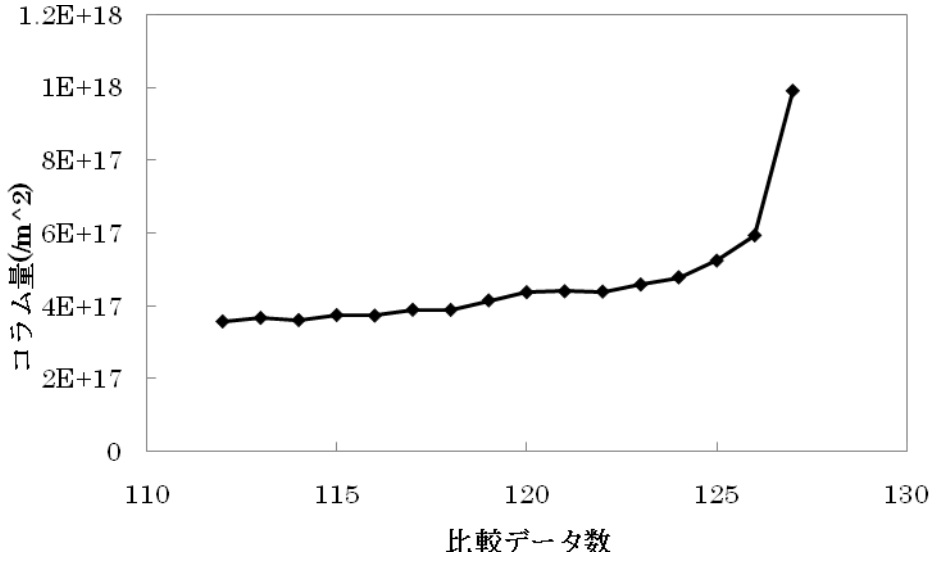


図 8.3　比較データ数とコラム量の関係

9. 最適解

この結果より、123 個目以降の結果は値が不安定、なおかつ他のデータと比べ、値が大きくずれているため最適なコラム量でないと判断できる。

9.1 複数のデータとの比較

最適な比較数を出すために、複数のデータに対し同じ処理を行う。そして、各々の波形に共通する特徴を見いだせると考える。そのために Excel で観測データを入力しウェーブレット変換を用いた場合と DOAS 法を用いた場合のそれぞれのコラム量を算出する雛型を作り、複数のデータの結果を比較した。

まず、2002 年 1 月 23 日 6 時 35 分、40 分、45 分の 3 つのデータの比較データ数に対するコラム量の関係を図 9.1 に示す。

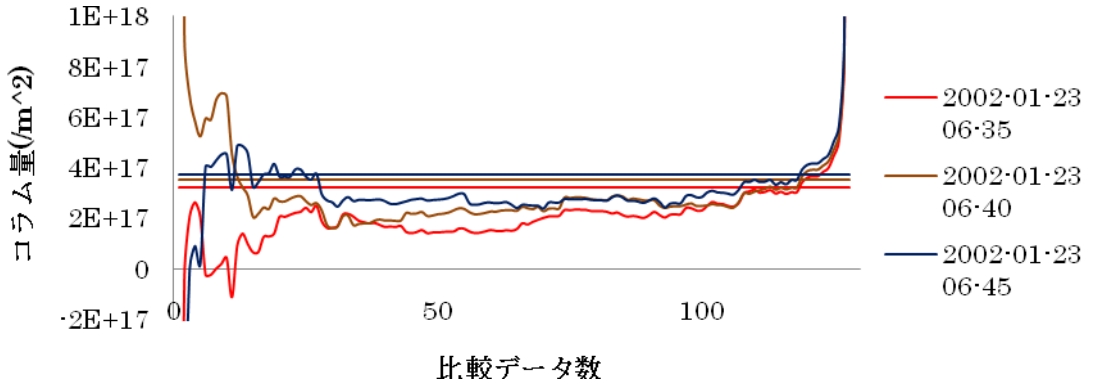


図 9.1　比較データ数とコラム量の関係

図 9.1 の波形において値が変動しない値がある。それは DOAS 法で求めた値であり、目安としてグラフにあるため横軸の比較データ数には依存しない。

図 9.1 から比較データ数の 90 以下の波形には特徴が似ている部分も見られるが、比較数が少なくなるにつれて値が大きく変動している。これは同じデータ数でコラム量を算出した場合に短時間でコラム量が大きく変化することは無いことからこの区間には最適な比較データ数は存在しない。比較データ数 90 以降のデータの波形に共通する特徴があるのが分かる。図 9.2 に結果の末端データ部分を拡大した図を示す。

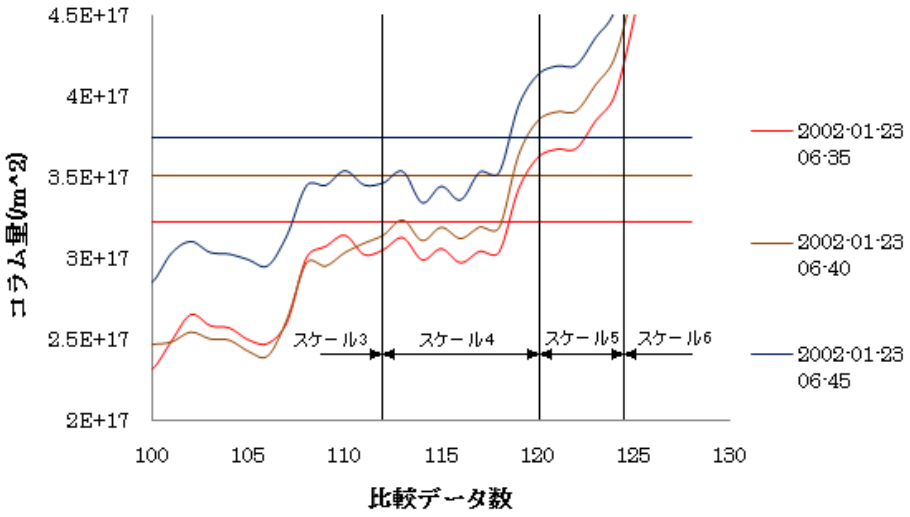


図 9.2　低周波成分を含んだ部分のコラム量

完全に波形の特徴が一致しているわけではないが、おおよそ一致している。また、DOAS 法で求めた値を目安とした場合、最適な比較データ数が 120 付近に存在すると予測される。

しかし、このグラフは 5 分おきに取得したデータを使用しているので、数時間後のデータと比較した場合、共通しない波形の特徴が観測されるかもしれない。そのため、同日の 15 時 00 分に観測したデータに対し同じ処理を行ったグラフを図 9.3 に示す。

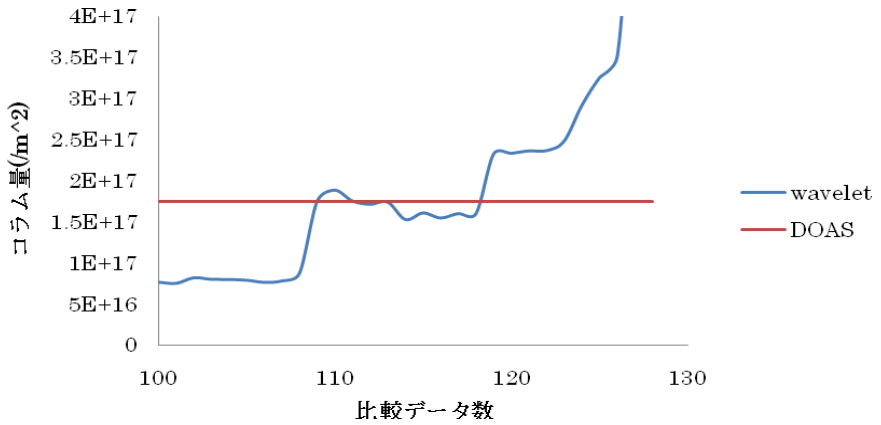


図 9.3　比較データ数とコラム量の関係

図 9.2 と図 9.3 を比較すると波形の特徴に多少の変化は見られるが、大きく変化していないためこの範囲内に最適な比較データ数が存在すると考えられる。

この範囲には、スケールの区切りであるスケール 3 までの比較データ数 112、スケール 4 までの 120、スケール 5 までの 124 の 3 つの閾が存在する。スケールの間で比較するということは故意の周波数成分が中途半端に削除された結果であるから正しい値が出てくるとは考えられない。故に、DOAS 法の値を目安にするとスケール 4 までの周波数成分で比較した値が最適な比較データ数だと予測される。

9.2　低周波成分の分布

データの末端部分はバイアス部分を示している。ということは、その部分を比較対象外にすることで削除できることは示した。吸収断面積には気体の成分のみで光学的厚さにはそれ以外の成分がバイアス部分として大きく影響している。縦軸を光学的厚さ、横軸を吸収断面積としたスペクトルマッチングを行った際に、末端部分のデータのプロットは他のものより大きく外れる。図 9.4 に実際にどのように散布しているかを示す。

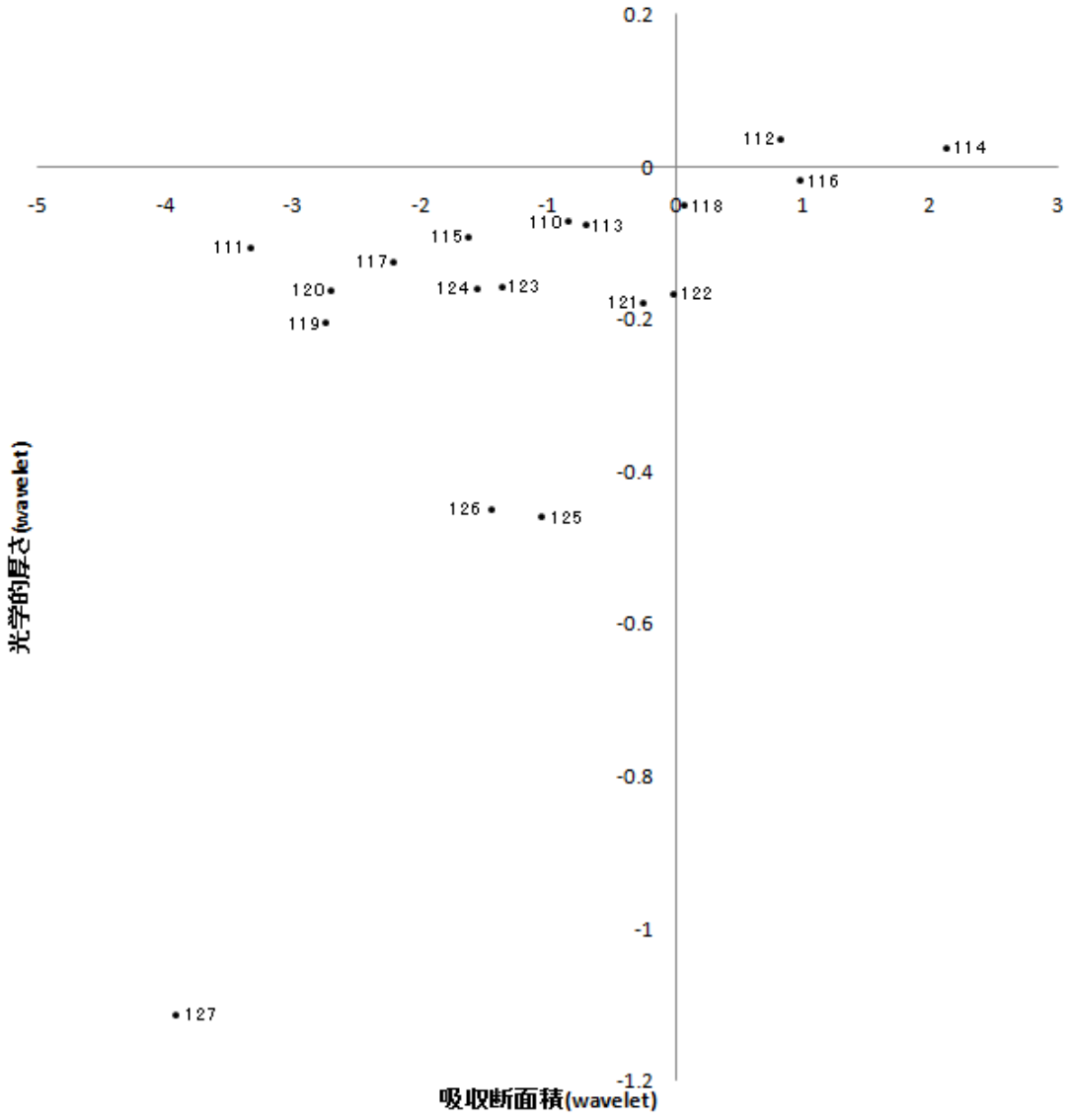


図 9.4　低周波成分のみのプロット

図 9.4 に比較データ数 127~110 までをプロットした。このことから 119 以上のプロットは他のデータよりも外れて見える。また、このときのデータは図 9.1 の 2002 年 1 月 23 日 6 時 45 分と同じである。

9.3　相関度

比較データ数とコラム量の理想のグラフの波形は比較データ数に関わらず、コラム量が一定であることだ。つまり、スペクトルマッチングを行ったときにプロットが一直線に並んでなければならない。しかし、求めたい気体成分以外の成分も含まれていることからそうはならない。そこで、最小二乗法により求めた近似式との相関を求めた。横軸を比較データ数、縦軸を相関度としたグラフを図 9.5 に示す。

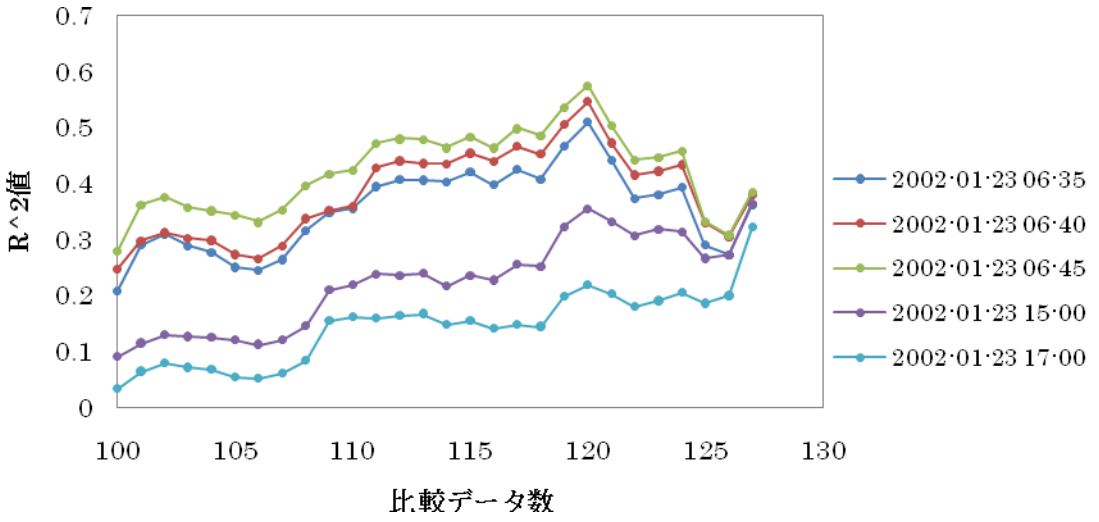


図 9.5　比較データ数と R^2 値

図 9.5 より、時間が経過するにつれ、R^2 値が減少しているのが分かる。また、比較データ数 127 の位置での値は大きく変化していない。これはデータの平均値の成分を削除した場合の値より、十分にガス以外の成分が削除されていないことになる。よってこれ以外で最も R^2 値が高いのは比較データ数 120 個の位置である。よって、最適な比較データ数を 120 個とした。

10. LabVIEW と DSP

この結果をもとに、120 個で比較した場合のコラム量を算出するプログラムを作成した。変更点は以下の通りである。

LabVIEW：配列データ(16 個のデータ)を受取り、複数のデータを画面表示するブロックダイヤグラムを配列データ(2 個のデータ)を受取り、1 個目のデータのみを画面表示するよう変更した。

DSP：プログラムの output を 2 個のデータの配列とし、1 個目には 120 個で比較した場合のコラム量、2 個目は 0 として返した。

```
for(i=0;i<16;i++){
        output[i]=0;
}
for(i=0;i<16;i++){
        output[i]=saij(Wkou,Wdan,112+i)*GAIN;
}
                ↓
output[0]=0;
output[1]=saij(Wkou,Wdan,120)*GAIN;
```

変更した後の実行結果を図 10.1 に示す。

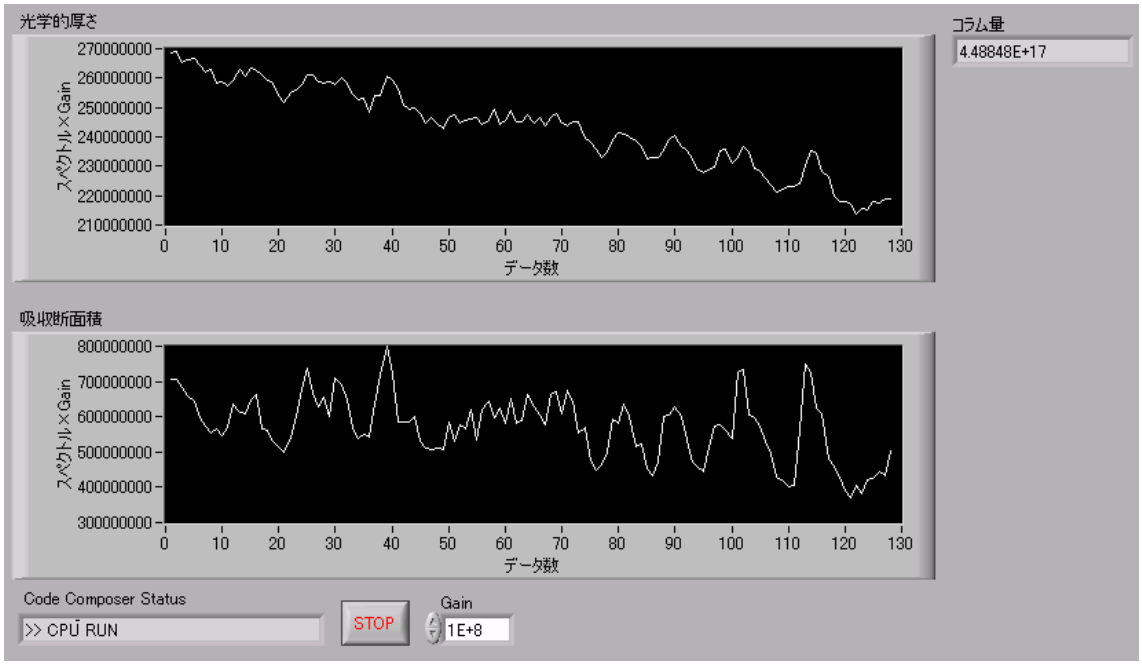


図 10.1　最適な比較データ数での算出

変更前の図 10.1 の結果の 120 個で比較した場合のときとコラム量が一致している。

11.　まとめ

測定されたスペクトルデータと二酸化窒素の吸収断面積のデータを基に、DSPでの解析処理をC言語、データ読み込みと送受信をLabVIEWでプログラムを作成し解析を行った。Haarのウェーブレット変換をDSPにおいて行い、コラム量を算出した。最小二乗法を行う範囲によって値が異なるため、最適な比較データ数を求めた。

12. 参考文献

[1]大川　善邦：波形の特徴抽出のための数学的処理、CQ出版社　(2005年)、p35-p62

[2]Code Composer Studio IDE Getting Started Guide User’s Guide

[3]TMS320C6713 DSP Starter Kit(DSK)インストレーションガイド、日本語版(2008年6月)

[4]F.Evangekisti, A.Baronecelli, P.Banasoni, G.Givanelli, and F.Ravegnani. Differential optical absorption spectrometer for measurement of tropospheric pollutants. Appl. Oct. , Vol. 34,pp.2737-2744,1995.

[5]A.C.Vandaele, C.Hermans, P.C.Simon, M.Carleer, R.Colin, S.Fally, and M.F.Merienne.Measurements of the NO2 absorption cross-section from 42000cm^-1 to 10000cm^-1 (238-1000nm) at 220 K and 294 K. J. Quant. Spec. Radi. Trans., Vol.59,pp.171-184,1998