地球物理データ解析　レポート #4

05-242628　三田村彰大

January 23, 2026

※ 用いた FORTRAN コードはhttps://github.com/ramutami/geo_data_analysis_report4_MITAMURAに上げている。またレポート末尾にも記載している。

問 1

まず今回用いる生データを図示すると下の Figure1のようになる。

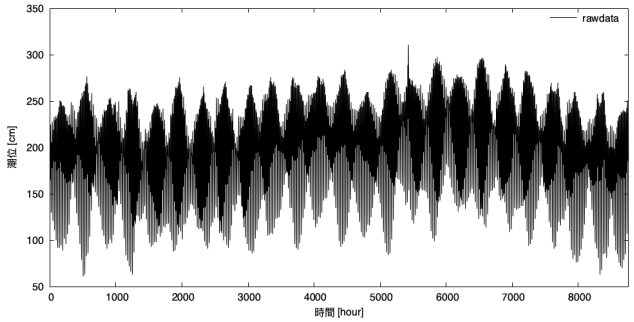


Figure1: 和歌山県串本町での 2023 年 1 月 1 日から 12 月 31 日までの毎時潮位観測値 (単位: cm)。ただし横軸は観測開始時からの経過時間 (単位：hour)

これに対しスペクトル解析を行うためにトレンドの除去と Cos20 テーパーによるテーパー処理を行う。以下にその計算手法について説明する。

まずはトレンドの除去についてであるが、今得られた潮位に関する離散データを h_i、観測時刻に関する時間データを t_i とするとき、そのトレンド $h = at + b$ を決定する係数 a, b は次のように決定される。

$$a = \frac{\sum_i \left(t_i - \overline{t}\right)\left(h_i - \overline{h}\right)}{\sum_i \left(t_i - \overline{t}\right)^2} = \frac{\overline{(t \cdot h)} - \overline{t} \cdot \overline{h}}{\overline{t^2} - \left(\overline{t}\right)^2} \quad , \quad b = \overline{h} - a \cdot \overline{t} \tag{1}$$

よってこれに対しトレンドが除去されたデータは

$$h_i^{\text{no-trend}} = h_i - (a \cdot t_i + b) \tag{2}$$

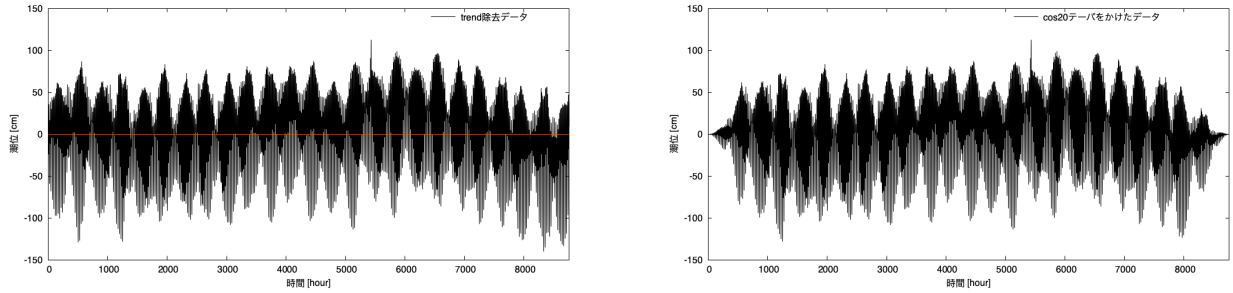
によって求めることができる。これを行なったのが Figure2aである。[*1]

次にテーパー処理についてであるが、今データの長さを N に対する Cos20 テーパーが問題文の通りに $W(n)$ で与えられる時、テーパー処理されたデータは

$$h_i^{\text{tapered}} = h_i^{\text{no-trend}} \cdot W(i) \tag{3}$$

[*1] 解析コードでは、module spectral_analysis の中の subroutine least_sq がこの計算を行っている。

で求まる。これを行ったのが Figure2bである。[*2]



(a) **(b)**

Figure2: (a) 観測データからトレンドを除去したもの。赤線は $y = 0$ の線を表す。(b) トレンドを除去したデータに Cos20 テーパーをかけたもの。

次に、整形されたデータ $\hat{h}_i = h_i^{\text{no-trend}}$ に対してフーリエ級数展開を行いそのフーリエ係数 a_p, b_p を求める。[*3]ただしフーリエ展開は

$$\hat{h}_i = \frac{a_0}{2} + \sum_{p=1} \left(a_p \cos\left(2\pi p \frac{i}{N}\right) + b_p \sin\left(2\pi p \frac{i}{N}\right) \right) \tag{4}$$

を満たし、各係数は次のように計算している。[*4]

$$a_p = \frac{2}{N} \sum_{i=0}^{N-1} \hat{h}_i \cos\left(2\pi p \frac{i}{N}\right) \quad , \quad b_p = \frac{2}{N} \sum_{i=0}^{N-1} \hat{h}_i \sin\left(2\pi p \frac{i}{N}\right) \tag{5}$$

ただし $p = N/2$ については trivial な式の修正が必要となる。[*5] これにより、片側スペクトルパワー密度 $G(f)$ が次のように求まる。[*6]

$$f_p = \frac{p}{T} \quad , \quad G(f_p) = 2T \left\{ \left(\frac{a_p}{2}\right)^2 + \left(\frac{b_p}{2}\right)^2 \right\} \tag{6}$$

これをプロットしたものが Figure3。

次に、スペクトルに対し l 平滑化を以下のように行う。

$$G^{\text{smooth}}(f_p) = \frac{1}{l} \sum_k G\left(f_{p-\frac{l-1}{2}+k-1}\right) \tag{7}$$

ただし l は奇数とする。これにより、着目している点の周囲 l 点の平均をとったスペクトルが得られる。[*7]このとき、自由度 ν は Cos20 テーパー後のスペクトルの自由度 1.792 に平滑化による自由度 l をかけたものを考え、

$$\nu = 1.792 \times l \tag{8}$$

[*2] 解析コードでは、module spectral_analysis の中の subroutine window_cos でこの処理を行なっている。ただしこのルーチンは一般の Cos($2w$) テーパーに対してテーパー処理を行えるよう、w を width 変数として受け入れている。
[*3] 解析コードでは、module spectral_analysis の中の subroutine fourier でこの処理を行なっている。
[*4] ただし、データサイズを N としたときこの計算プロセスで h_i を $i = 0, 1, \ldots, N-1$ のように番号付けしている。
[*5] 詳しくはコードを参照。
[*6] 授業では係数は $2T$ ではなく $2/T$ としていたが、スペクトルパワー密度の方が一般的なので $2T$ とした方が良い気がする。
[*7] 解析コードでは、module spectral_analysis の中の subroutine smoothify でこの処理を行なっている。

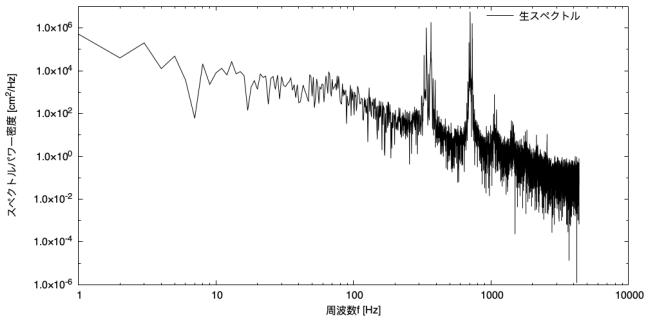


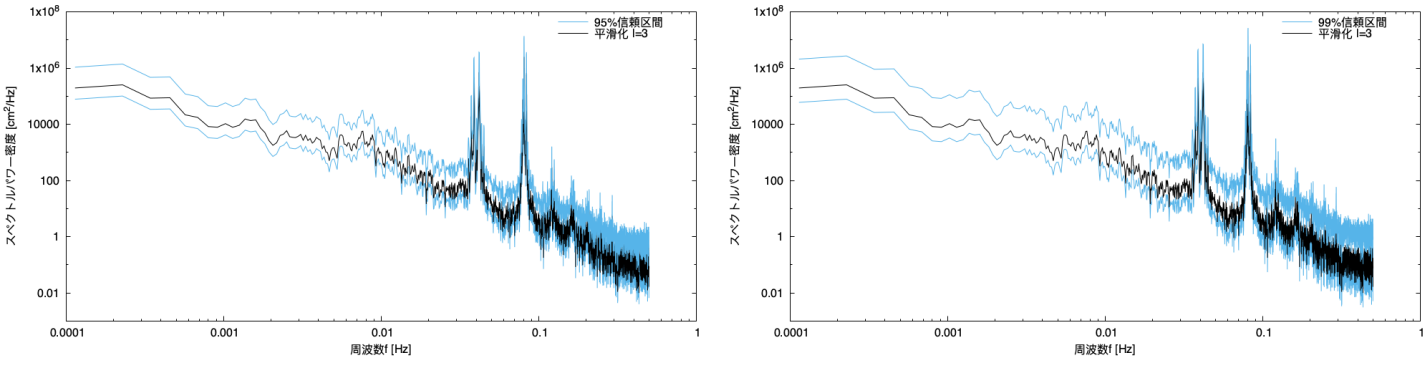
Figure3: 生スペクトル。両軸ともに対数軸となっている。

と考える。この ν に対しスペクトルの信頼区間は次のように与えられる。

$$\frac{\nu G^{\mathrm{smooth}}}{\chi^2\left(\nu, 1-\frac{\alpha}{2}\right)} \leq \hat{G} \leq \frac{\nu G^{\mathrm{smooth}}}{\chi^2\left(\nu, \frac{\alpha}{2}\right)} \tag{9}$$

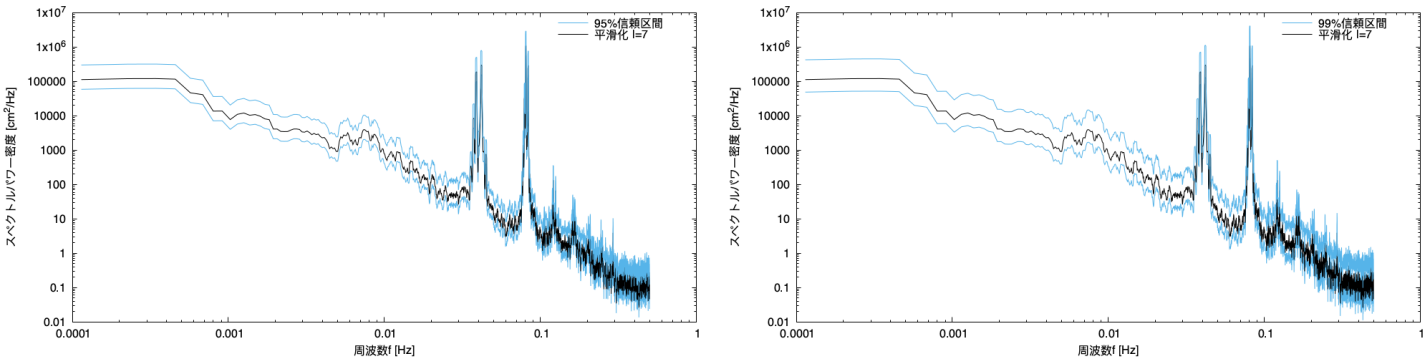
で推定される。ただし実際の解析コードにおいては、信頼区間 $95\%, 99\%$ それぞれの場合に対して $\chi^2(\nu)$ の値をあらかじめ python で計算し、それを FORTRAN コード中に手打ちで組み込んでいる。[*8] こうして得られた平滑化済みスペクトルとその信頼区間を図示したものが Figure4。また強いスペクトルが見られる範囲を拡大し、信頼区間のみを描画したものが Figure5

[*8] めんどくさかった...



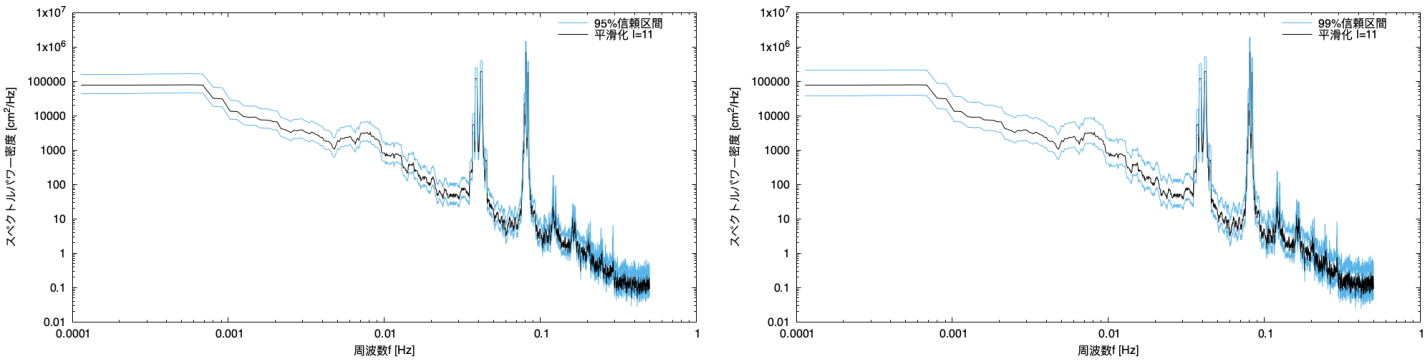
(a)

(b)



(c)

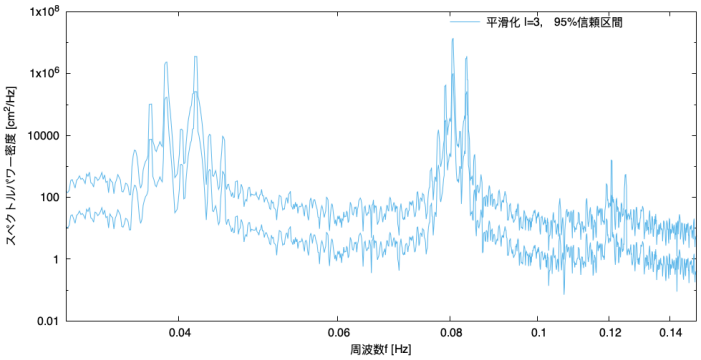
(d)



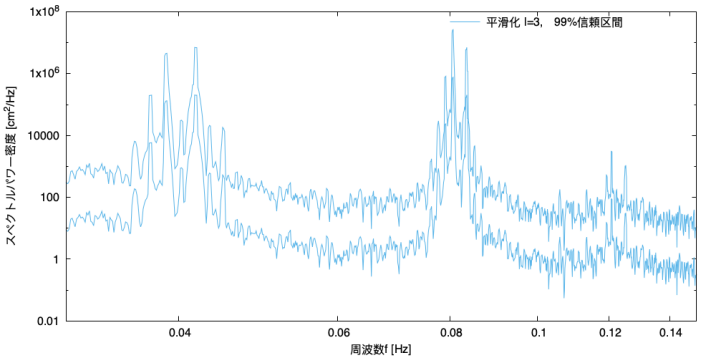
(e)

(f)

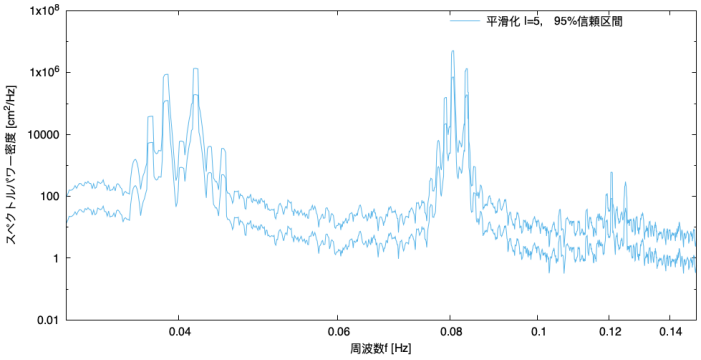
Figure4: 平滑化済みスペクトルとその信頼区間。左列が 95% 信頼区間、右列が 99% 信頼区間を示している。また上段 (a)(b) は $l = 3$ で平滑化、中段 (c)(d) は $l = 7$ で平滑化、上段 (e)(f) は $l = 11$ で平滑化している。



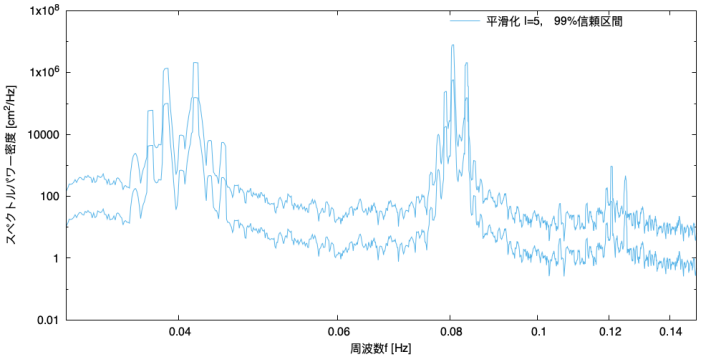
(a)



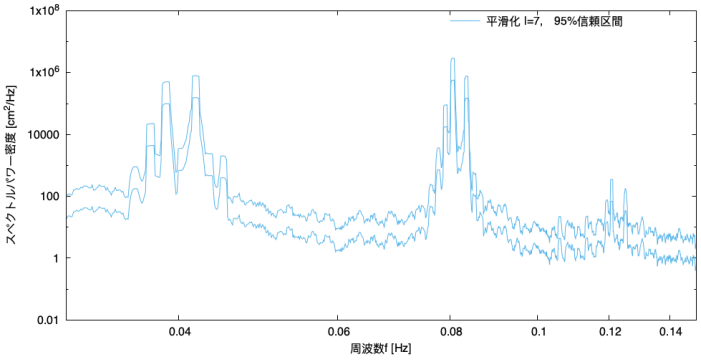
(b)



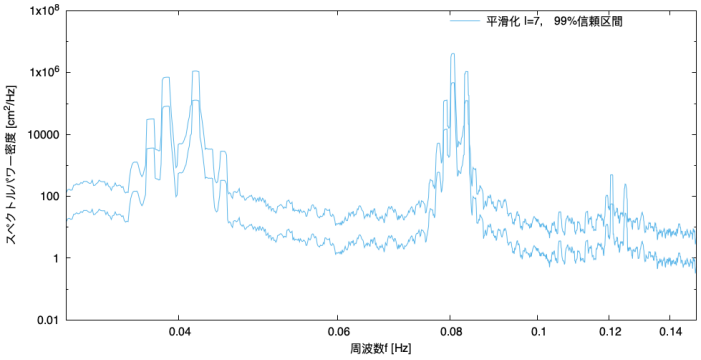
(c)



(d)



(e)

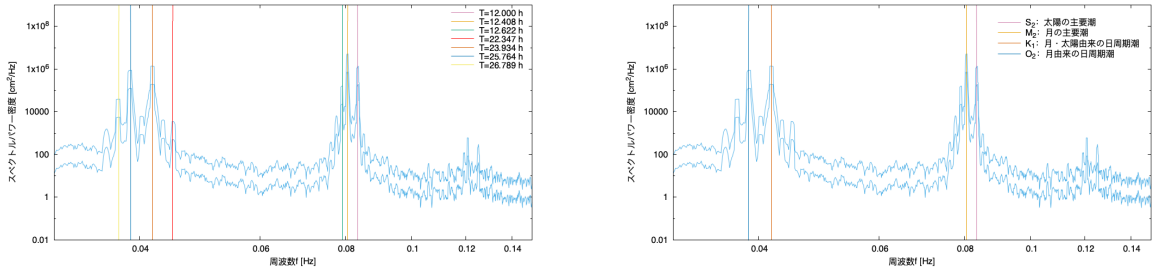


(f)

Figure5: スペクトルの信頼区間のみを図示したもの。左列が 95% 信頼区間、右列が 99% 信頼区間を示している。また上段 (a)(b) は $l = 3$ で平滑化、中段 (c)(d) は $l = 5$ で平滑化、上段 (e)(f) は $l = 7$ で平滑化している。

今、Figure5を見ると、平滑化を強くするほどピークの有意性が強く出る代わりにピーク位置が分かりにくくなる（ピークが潰れる）ことがわかる。そこで今回は、ピークの有意性とピーク位置がどちらも損なわれない「ちょうど良い」グラフとして、Figure5cの 95% 信頼区間で $l = 5$ 平滑化を行なったものを用いてピークを推定することにする。この図を用いれば、有意なピークはの Figure6aように推定できる。ただしそれぞれのピークの周期は図中に示した通りである。

これを見ると、スペクトルは周期が半日と一日となるような位置に幾つかのピークを持っていることがわかる。これは平衡潮の半日周期を持つものと日周期を持つものに由来していると考えられ、今地球力学の授業を参考にすればその中でも特に主要な 4 分潮の位置は Table1のように与えられる。よってこれらの分潮のピーク位置をスペクトル図に書き込んだものが Figure6bである。これを見ると、実際に主要 4 分潮が最も強い 4 つのピークとして現れていることが確認でき、解析が正しそうであると推察できる。



(a) **(b)**

Figure6: (a) 信頼区間で $l = 5$ 平滑化を行なったスペクトルに対し 95% 信頼区間を示したもの。図の縦線は有意なピーク位置を示しており、凡例にその周期を示している。(b) 潮汐から予想されるピークの位置を a の図に書き加えたもの。それぞれの潮汐ピークの成因は凡例に示したとおり。

分潮	期限	周期
O_1	月・赤緯	25h50m
K_1	月、太陽・赤緯	23h56m
M_2	月・主要潮	12h25m
S_2	太陽・主要潮	12h00m

Table1: 主要な平衡潮の期限・周期。地球力学の授業を参考にした。

問２

まずは与えられたデータを整形する。自分は冬季平均気温が担当なので、12 月 2 月の気温から各地点の平均気温を求め、地点 j の i 番目の年度における冬季平均気温を z_{ij} としてデータ行列 z_{ij} を作成する。その後、各地点における時系列の平均を各列の平均 $\overline{z_{\cdot,j}}$ として求め、それを元の行列から引くことで時系列の平均を除去する。

$$Z_{ij}^{\mathrm{new}} = Z_{ij} - \frac{1}{s}\sum_{i=1}^{s} Z_{ij} \tag{10}$$

その後、分散共分散行列 $V = \frac{1}{s}Z^T Z$ を求める。これに関しては FORTRAN に搭載されている行列計算用の関数（transpose,matmul,etc）などを用いた。その次に V についての固有値問題を解くが、これについても FORTRAN の LAPACK ライブラリを用いてすぐに計算することができる。

$$V\xi_m = \lambda_i \xi_i \tag{11}$$

よってこれにより固有ベクトル ξ_i と固有値 λ_i を求め、主成分時系列 $\hat{\tau}_i$、因子負荷量 $\hat{\xi}_i$、寄与率 χ_i が次のように求まる。

$$\hat{\tau}_i = \frac{1}{\sqrt{\lambda_i}} Z\xi_i \tag{12}$$

$$\hat{\xi}_i = \sqrt{\lambda_i}\xi_i \tag{13}$$

$$\chi_i = \frac{\lambda_i}{\sum_i \lambda_i} \times 100 \tag{14}$$

この時特に、$\Delta\chi_i \approx \sqrt{2/s}$ によって寄与率の不確かさを求めることができる。これにより寄与率をプロットしたものが Figure7である。また、主成分時系列は各測定年度 $t_1, t_2, \ldots, t_s$ に対して、因子負荷量は各測定点 $r_1, r_2, \ldots, r_p$ に対してプロットすればよく、これを $i = 3$ 番目のモードまで行なったものが Figure8である。ただし図では主成分時系列に対して直線フィットを行うことで線形トレンドが存在するかどうかをチェックしている

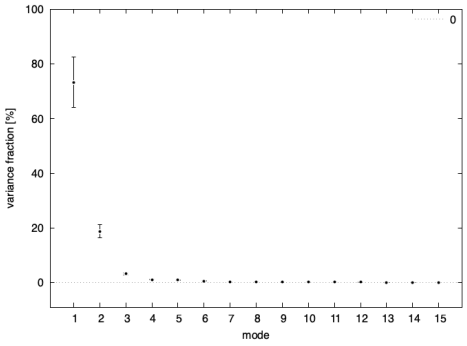
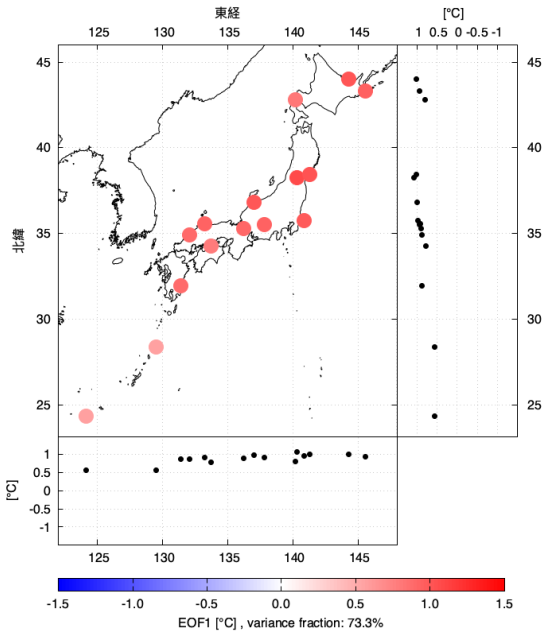
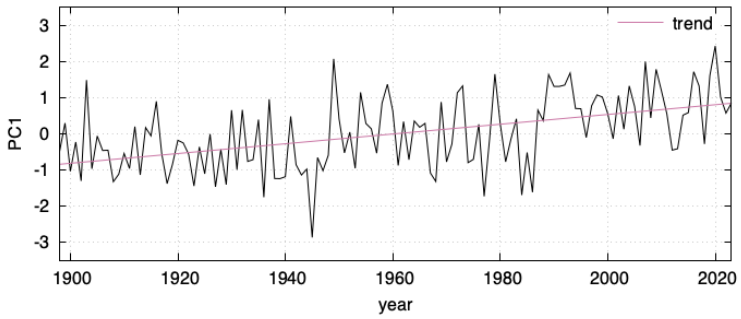
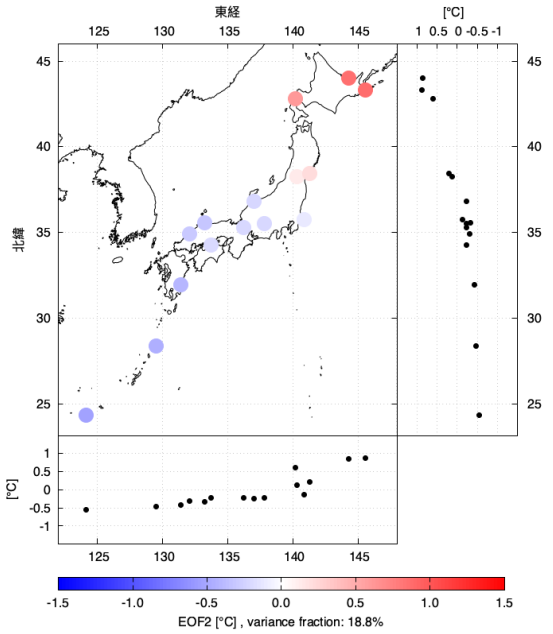
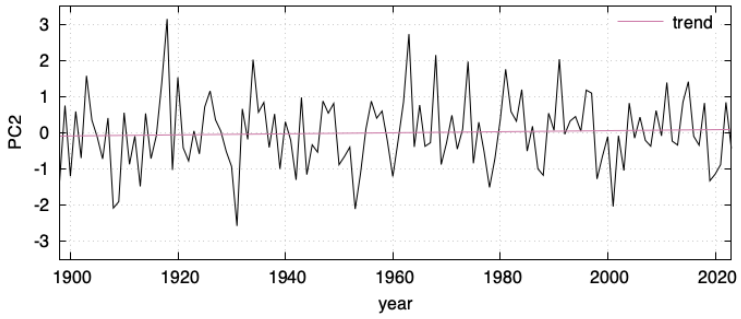


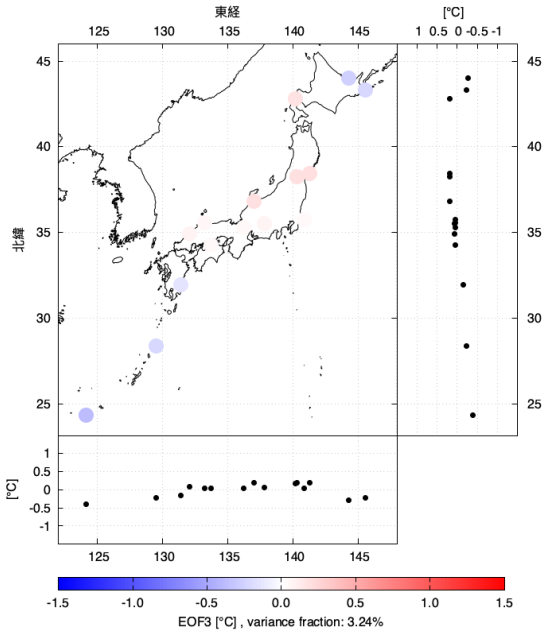
Figure7: 各モードの対する寄与率の不確かさ付きプロット。











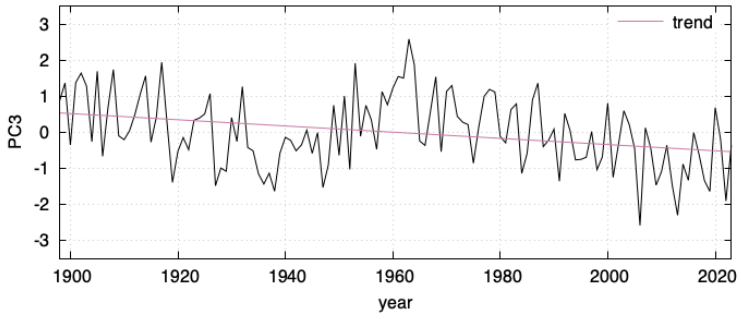


Figure8: 因子負荷量と主成分時系列。上から第１・第２・第３モードとなっている。因子負荷量の単位は °C で地図上にヒートマップで示している。地図の右と下についているグラフは２次元の地図を緯度・経度に分けて１次元に落としたもの。１次元に落とすことで因子負荷量をヒートマップではなくグラフの形でプロットすることが可能となっている。寄与率についてはそれぞれのヒートマップの下に記している。

まず Figure7の寄与率についてみると、第１、第２、第３モードはそれぞれ他のモードから独立しており、いずれも意味のあるモードであることが推察される。[*9]特に EOF1 が寄与率 73.3% 、EOF2 が寄与率 18.8%、EOF3 が寄与率 3.24% であり、分散のほとんどが EOF1 で、さらに分散の 95% 近くが上位３モードによって説明されることがわかる。

その上、ここから各モードについて見ていく。より見やすい拡大図を Figure9に用意した。まず第 1 モード（図の EOF1）についてであるが、EOF を見るとこのモードは全体として一定の値を取っており、PC score は増加トレンドを持っている。よってこのモードは日本全域で共通するような温度変化に対応したモードであり、それが地球温暖化によって徐々に増加しているということがわかる。また、沖縄列島については EOF の値が他の計測点に比べて若干低く、これは沖縄列島は冬季平均気温に関して変動が若干弱いことを示唆している。

次に EOF2 についてだが、このモードは石垣島で最も値が低く、東北方向に進むにつれ値が大きくなり根室で最大値を取っている。よってこれは高緯度ないし高東経で変動が大きくなるような因子を表していると考えられる。[*10]

最後に EOF3 についてだが、これは空間全体で小さな絶対値を持つモードであり、強いて言えば列島端点と列島中心で符号が異なる空間的な特徴を持っている。さらに主成分時系列を見ると、1940 から 1970 年にかけて上昇し、その後現代にかけて低下している。よってこのモードは、「列島中心と列島端で逆向きの変動」「1970 を境に逆向きのトレンド」を持つような成分に対応していると考えられ、おそらくは高度経済成長に起因する温度変化に相当するのではないかと推察される。すなわち列島端点（北海道東部や沖縄列島）では列島中心部に比べて経済発展に伴う大気汚染などの影響が現れにくく、また 1970 を境に高度経済成長が緩やかになったことや大気汚染対策がとられ始めたことでトレンドが逆転したことを表していると予想できる。[*11]

以上が日本の冬季年平均気温に関する主成分分析の結果と考察である。

以上

[*9] 正直この North's rule of thumb からモードが dominant かどうか判断するやり方について完全に理解できている気がしない。overlap がなければ他のモードを成分に持つことはなく、独立なモードであると言って良い、という認識であっているだろうか。

[*10] 気候学的な知見から高緯度ほど気候の年々変動が大きいことが予測されるので、おそらく経度よりは緯度による変動を表していると考えられる。

[*11] あまり確証はないが、これ以外に説明が思いつかない...

AppendixA: 因子負荷量の拡大図

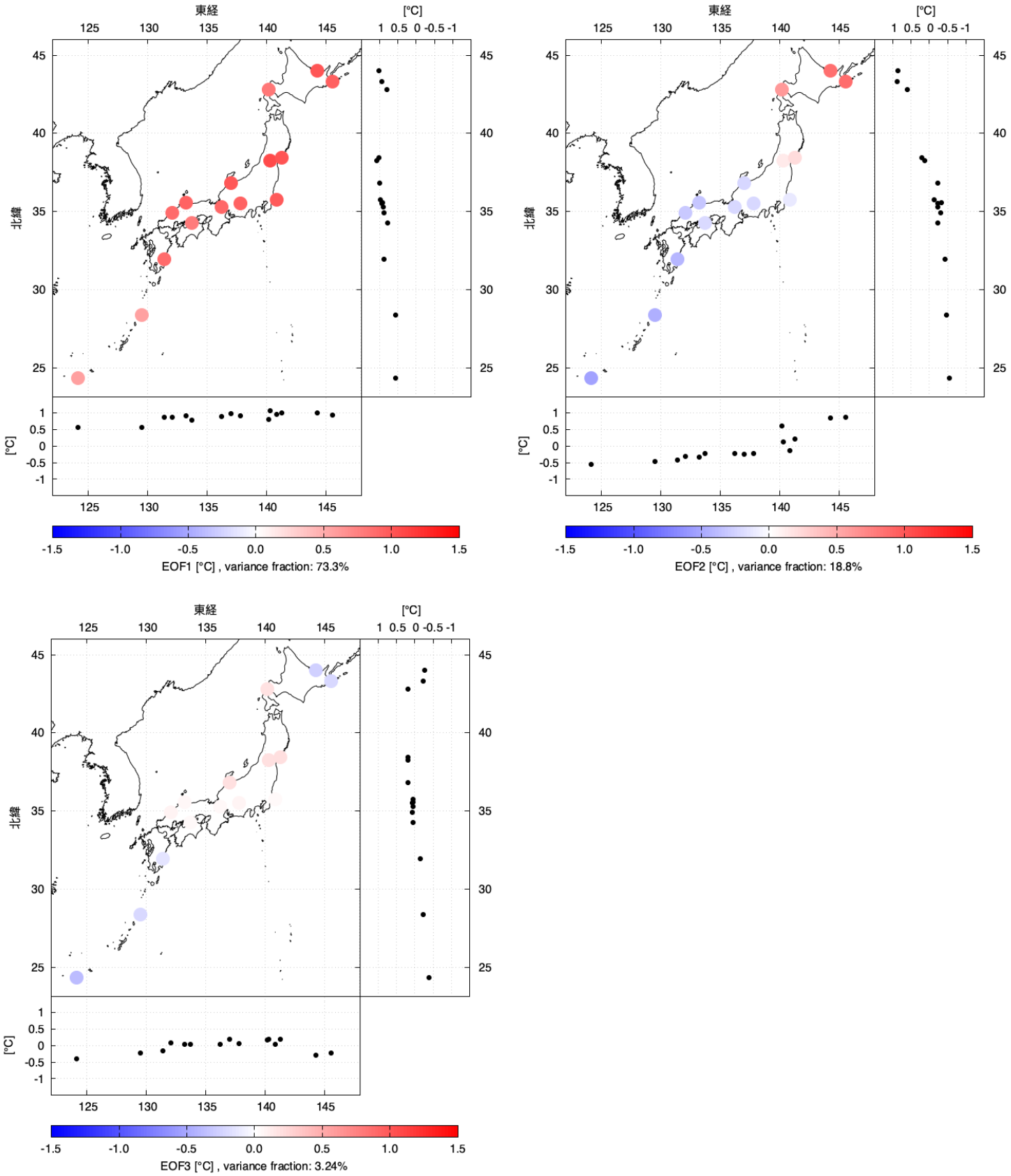


Figure9: 図を拡大して因子負荷量を見やすくしたもの。

AppendixB: スペクトル解析のコード

FORTRAN：Spectrum.f90

```
!===========cosテーパーを用いたスペクトル解析のmodule=================

!本課題の為に作成
!二次元離散データを受け入れ、DFTしたスペクトルを返す

!=====================================================================



module spectrum_analysis
    use iso_c_binding
    implicit none

    type two_dim_data
        real(8) :: x_t,y_t
    end type
    real(8) , parameter :: pi = 4.0*atan(1.0)

    contains

    subroutine spectrum(x_in,y_in,width,file_name,smoothify_size_in)
        !=================================================================

            !cosテーパーを用いたスペクトル解析のsubroutine
            !入力変数についての説明：
            !x_in,y_in：二次元離散データx,y
            !width：cosテーパーの幅。サイズをNとした時、N/widthに対してcosテーパーをかける。
            ! & 出力先のファイル名,平滑化のサイズ
            !出力データ1：データ+トレンド
            !一応、notrend.datも。
            !生スペクトルの.dat
            !平滑化と信頼区間の.dat


        !=================================================================

        !========定義===================================================
            implicit none
            real(8) , intent(in) :: width
            real(8) , allocatable , intent(in) :: x_in(:),y_in(:)
            type(two_dim_data),allocatable::spect_dat(:),spect_smooth(:)
            real(8) , allocatable :: x(:),y(:),x_temp(:),y_temp(:),a_p(:),b_p(:),spect_lower(:),
                spect_upper(:),&
            & chi_0975(:),chi_0025(:),chi_0995(:),chi_0005(:)
            real(8) :: a,b,s_a,s_b,T_max,T,ene_max,ene,safe_trend,safe_n,tot_time,ene_min,dist
            character(*),intent(in) :: file_name
            integer :: i,lb,ub,j,k,l,p,smoothify_size,nearlest_p(12)
            integer , intent(in),optional:: smoothify_size_in
            character(128) :: char_temp
            real(8) :: delta_x,degfree_cos20
        !=================================================================

        !========データの整形===========================================
            !x_n=(n-1)(delta_x)となるように揃える。
            lb = lbound(x_in,1)
            ub = ubound(x_in,1)
            allocate(x_temp(lb:ub),y_temp(lb:ub))
            j=1
            do i = lb,ub
                    x_temp(i) = x_in(i)
                    y_temp(i) = y_in(i)
                    j = j+1
            end do
            allocate(x(1:j-1),y(1:j-1))
            do i = lbound(x,1),ubound(x,1)
```

```
            x(i) = x_temp(i)-x_temp(lbound(x,1))
            y(i) = y_temp(i)
        end do
    !=====================================================================

    !================トレンドの除去=========================================
        !まずは、トレンドを出す。(一応検定もやる。)
        call least_sq(x,y,a,b,s_a,s_b)
        write(char_temp,'("./",(a),"/trend.dat")') file_name
        open(10,file=char_temp,status='replace',action='readwrite',position='append')
        write(10,*) 'original data size n',size(x)
        write(10,*) 'original data trend a :',a
        write(10,*) 's dev of a (s_a):',s_a
        write(10,*) '|a|/s_a',abs(a)/s_a
        write(10,*) 'if |a|/s_a > t(n,alpha/2) > > > 有意'
        write(10,*) 'every ::7*---------------------------*'
        write(10,*) 'below: x,y,ax+b'
        do i = lbound(x,1),ubound(y,1)
            write(10,*) x(i),y(i),a*(x(i))+b
        end do
        close(10)
        safe_trend = abs(a)/s_a
        safe_n = size(x)

        !トレンドの除去
        do i = lbound(x,1),ubound(x,1)
             y(i) = y(i) - (a*x(i) + b)
        end do
        call least_sq(x,y,a,b,s_a,s_b)
        write(char_temp,'("./",(a),"/no_trend.dat")') file_name
        open(10,file=char_temp,status='replace',action='readwrite',position='append')
        write(10,*) 'below: x,y(no_trend)'
        do i = lbound(x,1),ubound(x,1)
            write(10,*) x(i),y(i)
        end do
        close (10)
    !=====================================================================



    !===================テーパーをかける=====================================
        call window_cos(x,y,width)
        write(char_temp,'("./",(a),"/windowed.dat")') file_name
        open(10,file=char_temp,status='replace',action='readwrite',position='append')
        write(10,*) 'x,y(windowed)'
        do i = lbound(x,1),ubound(x,1)
            write(10,*) x(i),y(i)
        end do
        close(10)
    !=====================================================================

    !===================フーリエ展開=========================================
        do i = lbound(x,dim=1),ubound(x,dim=1)
            x_temp(i)=x(i)
            y_temp(i)=y(i)
        end do
        !y_tempに対してフーリエ級数を求める。
        call fourier (y_temp,a_p,b_p)
    !=====================================================================

    !===================periodogram[data^2/Hz]計算=========================
        allocate(spect_dat(0:ubound(a_p,1)),spect_smooth(0:ubound(a_p,1)))
        delta_x = x_temp(lbound(x_temp,1)+1)-x_temp(lbound(x_temp,1))
        tot_time = x_temp(ubound(x_temp,1)) + delta_x
        write(char_temp,'("./",(a),"/spect.dat")') file_name
        open(10,file=char_temp,status='replace',action='readwrite',position='append')
        do p = lbound(a_p,1) ,ubound(b_p,1)
            spect_dat(p)%x_t = real(p,8)/tot_time
            spect_dat(p)%y_t = 2.0*tot_time*((a_p(p)/2.0)**2.0 + (b_p(p)/2.0)**2.0)
            write(10,*) spect_dat(p)%x_t,spect_dat(p)%y_t
        end do
```

```
            close(10)
        !=========================================================================


        !==================平滑化、信頼区間→出力================================
            if(present(smoothify_size_in)) then
                smoothify_size = smoothify_size_in
            end if
            allocate(chi_0975(30*2+1),chi_0025(30*2+1),chi_0995(39*2+1),chi_0005(39*2+1))
            degfree_cos20=1.792
            !自由度nu=n*1.792に対するchi(nu,0.975)（下に2.5%を含む）をchi_0025(n)として出力
            !計算はpythonによる。
                chi_0025( 3 )= 0.9777583801485471
                chi_0025( 5 )= 2.6790372327592693
                chi_0025( 7 )= 4.730919506478876
                chi_0025( 9 )= 6.991107083542632
                chi_0025( 11 )= 9.392878162254968
                chi_0025( 13 )= 11.898836235052244
                chi_0025( 15 )= 14.48557588087762
                chi_0025( 17 )= 17.13731106136141
                chi_0025( 19 )= 19.84280341994206
                chi_0025( 21 )= 22.593717948364606
                chi_0025( 23 )= 25.383671391406732
                chi_0025( 25 )= 28.20764727653172
                chi_0025( 27 )= 31.061618560407368
                chi_0025( 29 )= 33.942294421203826
                chi_0025( 31 )= 36.84694468176403
                chi_0025( 33 )= 39.77327463086176
                chi_0025( 35 )= 42.71933362245316
                chi_0025( 37 )= 45.68344694358409
                chi_0025( 39 )= 48.66416409884615
                chi_0025( 41 )= 51.66021892266808
                chi_0025( 43 )= 54.67049837320031
                chi_0025( 45 )= 57.69401780496762
                chi_0025( 47 )= 60.72990114887334
                chi_0025( 49 )= 63.777364859545564
                chi_0025( 51 )= 66.83570479030558
                chi_0025( 53 )= 69.9042853686178
                chi_0025( 55 )= 72.98253059769327
                chi_0025( 57 )= 76.06991652132136
                chi_0025( 59 )= 79.16596487127077
            !自由度nu=n*1.792に対するchi(nu,0.05)をchi_0975(n)として出力
                chi_0975( 3 )= 13.447710703608488
                chi_0975( 5 )= 18.96380557029714
                chi_0975( 7 )= 24.099737509471836
                chi_0975( 9 )= 29.01820282364994
                chi_0975( 11 )= 33.79107892760641
                chi_0975( 13 )= 38.45766282092232
                chi_0975( 15 )= 43.04222288118531
                chi_0975( 17 )= 47.56099331415186
                chi_0975( 19 )= 52.02546826567394
                chi_0975( 21 )= 56.44413935175792
                chi_0975( 23 )= 60.82349106825315
                chi_0975( 25 )= 65.16860823840008
                chi_0975( 27 )= 69.4835657173505
                chi_0975( 29 )= 73.77168877462564
                chi_0975( 31 )= 78.03573303482955
                chi_0975( 33 )= 82.27801241529771
                chi_0975( 35 )= 86.5004923296101
                chi_0975( 37 )= 90.70485903176217
                chi_0975( 39 )= 94.89257216719133
                chi_0975( 41 )= 99.06490524915479
                chi_0975( 43 )= 103.22297728776792
                chi_0975( 45 )= 107.36777782653188
                chi_0975( 47 )= 111.50018699192152
                chi_0975( 49 )= 115.62099171896261
                chi_0975( 51 )= 119.73089900818947
                chi_0975( 53 )= 123.83054685203304
                chi_0975( 55 )= 127.9205133126795
                chi_0975( 57 )= 132.00132411985754
                chi_0975( 59 )= 136.07345907323648
```

```
!自由度nu=n*1.792に対するchi(nu,0.995)をchi_0995(n)として出力
    chi_0995( 3 )= 17.434838124612522
    chi_0995( 5 )= 23.524699086767832
    chi_0995( 7 )= 29.12900924337867
    chi_0995( 9 )= 34.453735183040855
    chi_0995( 11 )= 39.59054290558648
    chi_0995( 13 )= 44.58978608119794
    chi_0995( 15 )= 49.482665638027306
    chi_0995( 17 )= 54.2901058501206
    chi_0995( 19 )= 59.0269507072728
    chi_0995( 21 )= 63.70418402279958
    chi_0995( 23 )= 68.33020440472163
    chi_0995( 25 )= 72.91160487565068
    chi_0995( 27 )= 77.45367388554563
    chi_0995( 29 )= 81.96073055385443
    chi_0995( 31 )= 86.43635663646658
    chi_0995( 33 )= 90.88356163716229
    chi_0995( 35 )= 95.30490321073377
    chi_0995( 37 )= 99.70257682239857
    chi_0995( 39 )= 104.07848374777461
    chi_0995( 41 )= 108.43428348572884
    chi_0995( 43 )= 112.77143474124577
    chi_0995( 45 )= 117.09122788525215
    chi_0995( 47 )= 121.39481096291254
    chi_0995( 49 )= 125.68321075187812
    chi_0995( 51 )= 129.95734997562215
    chi_0995( 53 )= 134.21806149670633
    chi_0995( 55 )= 138.4661001134766
    chi_0995( 57 )= 142.7021524370405
    chi_0995( 59 )= 146.92684521711692
!自由度nu=n*1.792に対するchi(nu,0.005)をchi_0005(n)として出力
    chi_0005( 3 )= 0.504608708960929
    chi_0005( 5 )= 1.7186989734675253
    chi_0005( 7 )= 3.338634448457607
    chi_0005( 9 )= 5.212506825731709
    chi_0005( 11 )= 7.262910025584489
    chi_0005( 13 )= 9.444843220251949
    chi_0005( 15 )= 11.72956060390223
    chi_0005( 17 )= 14.097421116872098
    chi_0005( 19 )= 16.53431640136076
    chi_0005( 21 )= 19.029714497721372
    chi_0005( 23 )= 21.575509626807097
    chi_0005( 25 )= 24.165306850120572
    chi_0005( 27 )= 26.79395649790195
    chi_0005( 29 )= 29.45723957010198
    chi_0005( 31 )= 32.15164835268916
    chi_0005( 33 )= 34.87422927600011
    chi_0005( 35 )= 37.62246772736964
    chi_0005( 37 )= 40.39420190310766
    chi_0005( 39 )= 43.18755723167056
    chi_0005( 41 )= 46.00089566982287
    chi_0005( 43 )= 48.83277594866411
    chi_0005( 45 )= 51.681922012891285
    chi_0005( 47 )= 54.54719768061609
    chi_0005( 49 )= 57.42758608863616
    chi_0005( 51 )= 60.322172863459485
    chi_0005( 53 )= 63.23013222489234
    chi_0005( 55 )= 66.15071542107624
    chi_0005( 57 )= 69.08324103420856
    chi_0005( 59 )= 72.02708680003882
!


allocate(spect_lower(0:ubound(a_p,1)),spect_upper(0:ubound(a_p,1)))

do i = 1,smoothify_size
    !平滑化→出力
    call smoothify(spect_dat,spect_smooth,2*i+1)
    write(char_temp,'("./",(a),"/spect_smooth",i2.2,".dat")') file_name,2*i+1
    open(10,file=char_temp,status='replace',action='readwrite',position='append')
    do p = lbound(a_p,dim=1),ubound(a_p,dim=1)
```

```
                    write(10,*) spect_smooth(p)%x_t,spect_smooth(p)%y_t
                end do
                close(10)

                !95%信頼区間も含めた出力
                write(char_temp,'("./",(a),"/spect_95_",i2.2,".dat")') file_name,2*i+1
                open(10,file=char_temp,status='replace',action='readwrite',position='append')
                write(10,"(a,a,a,a)") '#frequency/','lower bound/','calculated value/','upper bound/'
                do p = 0,ubound(a_p,1)
                    spect_lower(p) = degfree_cos20*real(2*i+1,8)*spect_smooth(p)%y_t/chi_0975(2*i+1)
                    spect_upper(p) = degfree_cos20*real(2*i+1,8)*spect_smooth(p)%y_t/chi_0025(2*i+1)
                    write(10,*) spect_smooth(p)%x_t,spect_lower(p),spect_smooth(p)%y_t,spect_upper(p)
                end do
                close(10)

                !99%信頼区間も含めた出力
                write(char_temp,'("./",(a),"/spect_99_",i2.2,".dat")') file_name,2*i+1
                open(10,file=char_temp,status='replace',action='readwrite',position='append')
                write(10,"(a,a,a,a)") '#frequency/','lower bound/','calculated value/','upper bound/'
                do p = 0,ubound(a_p,1)
                    spect_lower(p) = degfree_cos20*real(2*i+1,8)*spect_smooth(p)%y_t/chi_0995(2*i+1)
                    spect_upper(p) = degfree_cos20*real(2*i+1,8)*spect_smooth(p)%y_t/chi_0005(2*i+1)
                    write(10,*) spect_smooth(p)%x_t,spect_lower(p),spect_smooth(p)%y_t,spect_upper(p)
                end do
                close(10)

            end do

        !=======================================================================

    end subroutine

    subroutine least_sq(x,y,a,b,s_a,s_b)
        !=======================================================================

            !最小二乗法によりトレンドをy=ax+bを算出する。
            !sa,sbはa,bの不確かさ

        !=======================================================================

        implicit none
        real(8), allocatable , intent(in):: x(:),y(:)
        real(8) :: a,b,ave_x,ave_y,ave_x2,ave_xy,s_y_p2,s_a_p2,s_b_p2,s_a,s_b
        integer :: n,i

        n = size(x)
        ave_x = sum(x)/real(n,8)
        ave_y = sum(y)/real(n,8)

        ave_x2 = 0
        do i =lbound(x,dim=1),ubound(x,dim=1)
            ave_x2 = ave_x2 + x(i)**2
        end do
        ave_x2 = ave_x2/(real(n,8))

        ave_xy = 0
        do i =lbound(y,dim=1),ubound(y,dim=1)
            ave_xy = ave_xy + x(i)*y(i)
        end do
        ave_xy = ave_xy/(real(n,8))

        a = (ave_xy-ave_x * ave_y)/(ave_x2 - ave_x**2)
        b = ave_y - a*ave_x

        s_y_p2 = 0
        do i = lbound(x,dim=1),ubound(x,dim=1)
            s_y_p2 = s_y_p2 + (y(i)-a*x(i)-b)**2
        end do
        s_y_p2 = s_y_p2/(real(n-2,8))

        s_a_p2 = (s_y_p2)/(real(n,8)*(ave_x2-ave_x**2))
```

```
        s_a = sqrt(s_a_p2)

        s_b_p2 = (ave_x2*s_y_p2)/(real(n,8)*(ave_x2-ave_x**2))
        s_b = sqrt(s_b_p2)

    end subroutine

    subroutine window_cos (x,y,width)
        !=====================================================================

            !cos20テーパー。
            !レポート問題ではn,Nを用いて与えられているが,以下ではx,delta(x)を用いて計算している。
            !ややこしいが、式は等価。

        !=====================================================================

        implicit none
        real(8) , allocatable , intent(in) :: x(:)
        real(8) , allocatable , intent(inout) :: y(:)
        real(8) , intent(in) :: width
        real(8) :: t,center,delta_x,xlb,xub
        integer :: i

        xub = x(ubound(x,dim=1))
        xlb = x(lbound(x,dim=1))
        t=x(ubound(x,dim=1))-x(lbound(x,dim=1))
        delta_x = x(lbound(x,dim=1)+1)-xlb
        center = sum(x)/real(size(x),8)

        do i = lbound(x,dim=1),ubound(x,dim=1)
            if((1.0-1.0/width)*(xub+delta_x)+0.5*delta_x < x(i)+delta_x) then
                y(i) = y(i)*(0.5 - 0.5*cos(width*pi*(x(i)+0.5*delta_x)/(xub+delta_x)))
            else if ( x(i) < (xub+delta_x)/width - 0.5*delta_x) then
                y(i) = y(i)*(0.5 - 0.5*cos(width*pi*(xub-x(i)+0.5*delta_x)/(xub+delta_x)))
            end if

        end do


    end subroutine

    subroutine fourier(y,a_p,b_p)
        !==================フーリエ展開=========================================
            !等間隔離散データy(:)に対しフーリエ係数を求める
            !ここについては講義資料に若干疑問がある。
            !t_n=(n-1)delta_tというデータを考えるのであれば、T=(N-1)delta_tではないのか？？？
        !=====================================================================

        implicit none
        real(8) , allocatable , intent(in) :: y(:)
        real(8) , allocatable :: a_p(:),b_p(:)
        integer :: p,i,ubound_p

        if (mod(size(y),2) == 0) then
            ubound_p = size(y)/2
        else
            ubound_p = (size(y)-1)/2
        end if

        if (allocated(a_p)) then
            deallocate(a_p)
        end if

        if (allocated(b_p)) then
            deallocate(b_p)
        end if

        allocate(a_p(0:ubound_p),b_p(0:ubound_p))
        do p = lbound(a_p,dim=1),ubound(a_p,dim=1)
            a_p(p) = 0
            b_p(p) = 0
```

```
420             do i = 1,size(y)
421                 a_p(p) = a_p(p)+y(i)*cos(2.0*pi*real(p*(i-1),8)/real(size(y),8))
422                 b_p(p) = b_p(p)+y(i)*sin(2.0*pi*real(p*(i-1),8)/real(size(y),8))
423             end do
424             a_p(p) = 2.0*a_p(p)/real(size(y),8)
425             b_p(p) = 2.0*b_p(p)/real(size(y),8)
426         end do
427 
428         if (mod(size(y),2) == 0) then
429             b_p(size(y)/2) = 0.0
430             !なんかaN/2は上の式の半分であるべき、みたいな話を見かけたが...
431             a_p(size(y)/2) = 0.5 * a_p(size(y)/2)
432         end if
433 
434 
435     end subroutine
436 
437     subroutine smoothify(spect,spect_smooth,l)
438         !====================平滑化=========================================
439             !平滑化ルーチン。
440             !本当はデータの端っこの平滑化についてみみっちい条件を考えなくてはいけない。
441             !しかし今回のスペクトル解析では端っこの周波数は注目に値しないので雑にやっている。
442         !====================================================================
443         implicit none
444         type(two_dim_data) , allocatable , intent(in) :: spect(:)
445         type(two_dim_data) , allocatable :: spect_smooth(:)
446         integer :: l,p,i
447 
448         if (allocated(spect_smooth)) then
449             deallocate(spect_smooth)
450         end if
451         allocate(spect_smooth(lbound(spect,1):ubound(spect,1)))
452 
453         do p = lbound(spect,1),ubound(spect,1)
454             spect_smooth(p)%x_t = spect(p)%x_t
455 
456             spect_smooth(p)%y_t = 0
457             do i = 1,l
458                 spect_smooth(p)%y_t = spect_smooth(p)%y_t +spect(p-((l-1)/2)+(i-1))%y_t
459             end do
460             spect_smooth(p)%y_t = spect_smooth(p)%y_t/l
461 
462         end do
463 
464     end subroutine
465 
466 end module
467 
468 
469 
470 program main
471     use spectrum_analysis
472     implicit none
473     integer::io,filesize,i
474     real(8),allocatable::raw_time(:),raw_height(:),raw_hour(:)
475 
476     !=============file読み込み=========================================
477 
478         !rawdataをrawdata_timeとrawdata_heightに読み込む。
479         open(10,file='./rawdata/rawdata.dat',status='old',action='read')
480         filesize=0
481         do
482             read(10,'(A)',iostat=io)
483             if(is_iostat_end(io)) then
484                 exit
485             end if
486             filesize=filesize+1
487         end do
488         allocate(raw_time(filesize),raw_height(filesize))
489         rewind(10)
490         do i = 1,filesize
```

```
491             read(10,*) raw_time(i),raw_height(i)
492         end do
493         close(10)
494 
495     !=================================================================
496 
497     !=============データ整形=========================================
498 
499         !与えられたデータにトビがないか確認する。
500         !日時データの飛びが+1か+77(=100-23)になっていれば良い。
501         !そうでないのは、月跨ぎの箇所に限られるはず
502         do i = 1,size(raw_time)-1
503             if (.not. (raw_time(i+1)-raw_time(i) == 1.0 .or. raw_time(i+1)-raw_time(i) == 77.0)) then
504                 !write(*,*) "There's a gap here!",raw_time(i),raw_time(i+1)
505                 !>> gap は月跨ぎの箇所のみ
506                 !>> よってデータにとびはない。
507             end if
508         end do
509         allocate(raw_hour(size(raw_time)))
510         do i = 1,size(raw_hour)
511             raw_hour(i) = real(i,8)
512         end do
513 
514     !=================================================================
515 
516     !===================DFT===========================================
517         call spectrum(raw_hour,raw_height,real(10,8),'./dataout',29)
518     !=================================================================
519 
520 
521 
522 end program
```

AppendixC:PMC のコード

FORTRAN：PMC.f90

```
    program main
    implicit none

    !=======変数定義=======================
        !とっ散らかっているのは新たに定義した変数を都度加えているから。（申し訳ないです）
        integer::io,filesize,i,j,EOF_size,num_mode
        real(8),allocatable::raw_data(:,:),winter_temp(:,:),Z_data(:,:),V_mat(:,:)
        real(8) :: latitude(15),longtitude(15),temp12(15),temp1(15),temp2(15)
        character(20) :: temp,site(15)
        integer,allocatable :: year_temp(:),month(:),year(:)
        real(8) :: z_column_ave(15),temp_real,a,b,sa,sb
        integer:: S_timesize,P_spacesize
        real(8) , allocatable :: temp_mat(:,:),eigen_values(:),eigen_vectors(:,:),temp_vect(:)
        real(8),allocatable ::PC_score(:,:),EOF(:,:),var_frac(:),delta_varfrac(:),temp_vect_2(:),PC_trend
            (:,:)

        !lapack関係
        integer :: lwork,liwork,lda,info
        real(8), allocatable :: work(:)
        integer, allocatable :: iwork(:)
        real(8) :: work_query(1)
        integer :: iwork_query(1)

    !=======変数定義=======================



    !=======データ整形=======================
        !地点jの名称をベクトルsite(j)に収納する。
        !ある地点jの緯度軽度をベクトルlatitude(j),longtitude(j)に収納する。
        !与えられたデータから、地点jのある年iにおける冬季平均気温z_ijを求め、行列Z_data(i,j)に収納する。
        open(10,file='./rawdata/rawdata.dat',status='old',action='read')
        read(10,*) temp,site
        read(10,*) temp,latitude
        read(10,*) temp,longtitude
        filesize=0
        do
            read(10,'(A)',iostat=io)
            if(is_iostat_end(io)) then
                exit
            end if
            filesize=filesize+1
        end do
        allocate(year_temp(filesize),month(filesize),raw_data(filesize,15))
        rewind(10)
        read(10,'(A)')
        read(10,'(A)')
        read(10,'(A)')
        do i = 1,filesize
            read(10,*) year_temp(i),month(i),raw_data(i,:)
        end do
        close(10)
        allocate(winter_temp(maxval(year_temp,1)-minval(year_temp,1),15))
        allocate(year(size(winter_temp,1)))
        !冬の平均気温を算出する。
        j=0
        do i = 1,size(year_temp)
            if (month(i) == 12) then
                temp12 = raw_data(i,:)
            else if (month(i) == 1) then
                temp1 = raw_data(i,:)
            else if (month(i) == 2) then
                temp2 = raw_data(i,:)
            end if

```

```
65              if (month(i) == 2) then
66                  j = j+1
67                  year(j) = year_temp(i)
68                  winter_temp(j,:) = (temp12+temp1+temp2)/3.0_8
69              end if
70
71          end do
72
73
74          open(10,file='./dataout/wintertemp.dat',status='replace',action='write')
75          write(10,'(A4,1X,*(A16,1X))') 'year',site(:)
76          do j = 1,size(year)
77              write(10,'(I4,1X,*(F19.16,1X))') year(j),winter_temp(j,:)
78          end do
79          close(10)
80
81          allocate(Z_data(size(year),15))
82          do i = 1,size(year)
83              do j = 1,15
84                  Z_data(i,j) = winter_temp(i,j)
85              end do
86          end do
87
88          open(10,file='./dataout/Z_data.dat',status='replace',action='write')
89          do i = 1,size(Z_data,dim=1)
90              write(10,'(*(F19.16,1X))') Z_data(i,:)
91          end do
92          close(10)
93      !=======データ整形=========================
94
95      !=======(1):trend除去=========================
96          !Zから時系列の平均を除去する。
97          do j = 1,size(Z_data,2)
98              Z_column_ave(j)=sum(Z_data(:,j))/real(size(Z_data,1),8)
99              Z_data(:,j) = Z_data(:,j) - Z_column_ave(j)
100         end do
101
102         open(10,file='./dataout/Z_data_ave_removed.dat',status='replace',action='write')
103         do i = 1,size(Z_data,dim=1)
104             write(10,'(*(F19.16,1X))') Z_data(i,:)
105         end do
106         close(10)
107
108     !=======(1):trend除去=========================
109
110     !=======(2):V計算=================================
111         !V=1/s Z^T Z を計算。
112         P_spacesize = size(Z_data,2)
113         S_timesize = size(Z_data,1)
114         allocate(V_mat(P_spacesize,P_spacesize))
115         V_mat = matmul(transpose(Z_data),Z_data)
116         V_mat = V_mat/real(S_timesize,8)
117
118         !一応、ちゃんと対称行列になっている確認する。
119         open(10,file='./dataout/V_check.dat',status='replace',action='write')
120         allocate(temp_mat(size(V_mat,1),size(V_mat,1)))
121         temp_mat = transpose(V_mat)-V_mat
122         do i = 1,size(temp_mat,1)
123             write(10,'(*(F19.16,1X))') temp_mat(i,:)
124         end do
125         close(10)
126     !=======(2):V計算=================================
127
128     !=======(3):固有値問題を解く=================================
129         !LAPACKのDSYEVDを使う。
130         !固有値は昇順で返してくれるので便利。
131         !固有ベクトルも正規化されているので便利。
132         !対象の行列は固有ベクトルで上書きされてしまうので、temp_matにVを収納する。
133
134         !必要作業領域の確認(trivial)
135         lwork=-1
```

```
136         liwork=-1
137         temp_mat = V_mat
138         lda = size(temp_mat,1)
139         allocate(eigen_values(size(temp_mat,1)))
140         call dsyevd('V','U', size(temp_mat,1),temp_mat, lda, eigen_values, work_query, lwork, iwork_query,
                 liwork, INFO)
141         lwork  = int(work_query(1))
142         liwork = iwork_query(1)
143         write(*,*) "eigen calc status (0=OK) : ", info
144         allocate(work(lwork))
145         allocate(iwork(liwork))
146
147         !固有値・固有ベクトルを求める
148         temp_mat = V_mat
149         call dsyevd('V','U', size(temp_mat,1),temp_mat, lda, eigen_values, work, lwork, iwork, liwork,
                INFO)
150         write(*,*) "eigen calc status (0=OK) : ", info
151         !固有値について降順にする
152         allocate(temp_vect(size(V_mat,1)))
153         do j = 1, size(eigen_values)/2
154             i = size(eigen_values) - j + 1
155             ! 固有値 swap
156             temp_real = eigen_values(j)
157             eigen_values(j) = eigen_values(i)
158             eigen_values(i) = temp_real
159             ! 固有ベクトル列swap
160             temp_vect(:) = temp_mat(:,j)
161             temp_mat(:,j) = temp_mat(:,i)
162             temp_mat(:,i) = temp_vect(:)
163         end do
164         write(*,*)
165         !固有ベクトルxiiをeigen_vectors行列に(xi1,xi2,...,xim)のように収納
166         allocate(eigen_vectors(size(V_mat,1),size(eigen_values)))
167         do j = 1,size(eigen_values)
168             eigen_vectors(:,j) = temp_mat(:,j)
169         end do
170
171         !確認
172         open(10,file='./dataout/eigen_check.dat',status='replace',action='write')
173         write(10,*) "eigen_val,V*eigen_val-eigen_val*eigen_vect,norm(eigen_vect)"
174         write(10,*) "eigen_val should be in descending order"
175         write(10,*) "V*eigen_val-eigen_val*eigen_vect should be 1"
176         write(10,*) "norm(eigen_vect) should be 0"
177         do j = 1,size(eigen_values)
178             write(10,*) eigen_values(j),sum((matmul(V_mat,eigen_vectors(:,j))-eigen_values(j)*
                    eigen_vectors(:,j))**2.0_8),sqrt(sum(eigen_vectors(:,j)**2.0_8))
179         end do
180         close(10)
181
182     !=======(3):固有値問題を解く================================
183
184     !=======(4):寄与率、PCscore、EOFの計算=================================
185         !以下モードに関するiterationはjで表す(j-th mode)
186         !PC_score(i,j)行列にPC_scoreを(tau1,tau1,...)のように収納。二次元目(j)がmode番号に対応。
187         !EOF(i,j)行列にEOFを(xi1,xi2,...)のように収納。二次元目(j)がmode番号に対応。
188         !var_frac(j)に寄与率を収納。(j)がmode番号に対応。
189         !delta_varfrac(j)にnorth's rule of thumbからvar_fracの不確かさを求めたものを収納。
190
191         num_mode = size(eigen_values)
192         EOF_size = size(eigen_vectors,1)
193         allocate(PC_score(S_timesize,num_mode),EOF(EOF_size,num_mode),var_frac(num_mode),delta_varfrac(
                num_mode))
194
195         do j = 1,num_mode
196             var_frac(j) = eigen_values(j)*100.0_8/sum(eigen_values)
197             EOF(:,j) = sqrt(eigen_values(j))*eigen_vectors(:,j)
198             PC_score(:,j) = (1.0_8/sqrt(eigen_values(j)))*matmul(Z_data,eigen_vectors(:,j))
199             delta_varfrac(j) = var_frac(j)*sqrt(2.0_8/S_timesize)
200         end do
201
202         open(10,file='./dataout/PC_score.dat',status='replace',action='write')
```

```
        write(10,'(A)') 'PC_score'
        write(10,'(A4,1X,*(I19,1X))')'mode',[(i, i=1,num_mode)]
        do i = 1,S_timesize
            write(10,'(I4,1X,*(F19.16,1X))') year(i),PC_score(i,:)
        end do
        close(10)


        open(10,file='./dataout/EOF.dat',status='replace',action='write')
        write(10,'(A)') 'EOF per mode'
        write(10,'(A16,1X,A19,1X,A19,1X,*(I19,1X))') 'site','logtitude','latitude',[(i, i=1,num_mode)]
        do i = 1,EOF_size
            write(10,'(A16,1X,F19.14,1X,F19.16,1X,*(F19.16,1X))') site(i),longtitude(i),latitude(i),EOF(i
                ,:)
        end do
        close(10)

        open(10,file='./dataout/kiyo.dat',status='replace',action='write')
        write(10,'(A)') '寄与率'
        write(10,'(A4,1X,A19,1X,A19)') 'mode_num','variance_fraction','delta_var_frac'
        do j = 1,num_mode
            write(10,'(I4,1X,F19.16,1X,F19.16)') j,var_frac(j),delta_varfrac(j)
        end do
        close(10)

    !=======(4):寄与率、PCscore、EOFの計算================================

    !=======(5):PC_scoreのtrend算出=================================
        deallocate(temp_vect)
        allocate(temp_vect(S_timesize),temp_vect_2(S_timesize),PC_trend(S_timesize,num_mode))
        temp_vect_2 = real(year,8)

        do j = 1,num_mode
            temp_vect = PC_score(:,j)
            call least_sq(temp_vect_2,temp_vect,a,b,sa,sb)
            PC_trend(:,j) = a*temp_vect_2+b
        end do

        open(10,file='./dataout/PC_trend.dat',status='replace',action='write')
        write(10,'(A)') 'PC_trend'
        write(10,'(A4,1X,*(I19,1X))')'mode',[(i, i=1,num_mode)]
        do i = 1,S_timesize
            write(10,'(I4,1X,*(F19.16,1X))') year(i),PC_trend(i,:)
        end do
        close(10)


    !=======(5):PC_scoreのtrend算出=================================

    stop
    contains

    subroutine least_sq(x,y,a,b,s_a,s_b)

        implicit none

        real(8), allocatable , intent(in):: x(:),y(:)
        real(8) :: a,b,ave_x,ave_y,ave_x2,ave_xy,s_y_p2,s_a_p2,s_b_p2,s_a,s_b
        integer :: n,i

        n = size(x)
        ave_x = sum(x)/real(n,8)
        ave_y = sum(y)/real(n,8)

        ave_x2 = 0
        do i =lbound(x,dim=1),ubound(x,dim=1)
            ave_x2 = ave_x2 + x(i)**2
        end do
        ave_x2 = ave_x2/(real(n,8))

        ave_xy = 0
```

```
273         do i =lbound(y,dim=1),ubound(y,dim=1)
274             ave_xy = ave_xy + x(i)*y(i)
275         end do
276         ave_xy = ave_xy/(real(n,8))
277 
278         a = (ave_xy-ave_x * ave_y)/(ave_x2 - ave_x**2)
279         b = ave_y - a*ave_x
280 
281         s_y_p2 = 0
282         do i = lbound(x,dim=1),ubound(x,dim=1)
283             s_y_p2 = s_y_p2 + (y(i)-a*x(i)-b)**2
284         end do
285         s_y_p2 = s_y_p2/(real(n-2,8))
286 
287         s_a_p2 = (s_y_p2)/(real(n,8)*(ave_x2-ave_x**2))
288         s_a = sqrt(s_a_p2)
289 
290         s_b_p2 = (ave_x2*s_y_p2)/(real(n,8)*(ave_x2-ave_x**2))
291         s_b = sqrt(s_b_p2)
292 
293 
294     end subroutine
295 
296 
297 end program main
```