雲の水平断面形状のフラクタル次元：気象衛星画像を用いた時空間変動の解析

岡山大学　環境生命自然科学学科　地球科学コース
50M24759 熊本　庄真

2026/02/15

要旨

インド洋東部から太平洋西部の熱帯域における雲の水平断面形状のフラクタル次元の時空間変動を調べた．解析には，気象衛星ひまわりが 2011 年から 2023 年に観測した 1 時間間隔の IR1（10.3–11.3 μm）赤外画像を用い，観測された赤外放射の輝度温度を再解析データの鉛直温度分布と対応させることで雲頂圧力を推定した．さらに，雲頂より下は地表まで雲が存在していると仮定し，いくつかの圧力面における雲の水平断面形状を決定した．決定した断面形状に対してボックスカウント法を用いて，フラクタル次元を算出した．雲のフラクタル次元には季節変化が存在した．また，一部の地域ではエルニーニョとの関係が示唆された．

abstract

We measured the fractal dimension of clouds in the tropics from the eastern Indian Ocean to the western Pacific Ocean. This analysis utilized hourly infrared images (IR1, 10.3-11.3 μm) observed by the Geostationary Meteorological Satellite (HIMAWARI) from 2011 to 2023. By comparing the brightness temperature of observed infrared radiation with the vertical temperature profile from JRA-55 reanalysis data, we estimated the cloud-top pressure. Then, the horizontal cross-sectional shape of clouds at several pressure levels are derived from the horizontal distribution of cloud-top pressure. We measured the fractal dimension of the cross-sectional shape of clouds by using the box-counting method. The fractal dimension of clouds shows seasonal variation which differ by regions. Also, the influence of El Ninõ on the fractal dimension of clous is inferred.

目 次

第1章　序論

雲は太陽放射と惑星放射の両方に作用することで，惑星の気候に大きな影響を及ぼしている．例えば、雲量が約 50%，つまり地球の半分が雲で覆われているとき，その放射強制力は $20\mathrm{W/m^2}$ にもなり，これは温暖化で問題とされる二酸化炭素が倍増したときの放射強制力の 10 倍以上にもなる．しかしながら，雲がどのような物理過程によって規定されているのかは明らかになっていないため，温暖化したときに雲が増えるのか減るのかといったことさえわかっていない．本研究では，その雲を支配する物理を明らかにする条件の一つになり得るものとして，雲の幾何学的特徴に着目した.

1.1　雲の幾何学的特徴

雲の幾何学的特徴に着目した研究として，Lovejoy (1982) が雲の周囲長と面積を用いて雲の断面形状がフラクタルであると指摘した．また,Yano and Takeuchi (1987) も、気象衛星 GMS-3 の赤外画像を用いて解像度によって雲の周囲長がどのように変わるかを測定することで，熱帯収束帯における雲のフラクタル次元が約 1.5 であるとの結果を得た.

フラクタルとは図形の一部を拡大したときに全体と同じ構造が現れる図形である．図 1.1 に示すコッホ雪片は，フラクタルの代表的な例の一つであり，一部を拡大すると全体と同じ形が繰り返し現れる.

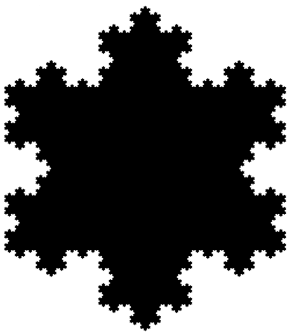


図 1.1: コッホ雪片

フラクタルはフラクタル次元という特徴量をもつ．これはフラクタル図形が，物差しを用いて長さを測定したとき，物差しの大きさによって測定される長さが変化するという性質に関連するものである．フラクタル図形は，あらゆるスケールの凸凹をもつが，測定可能な凸凹のスケールは物差しの大きさによって決まる．すなわち，小さい物差しを使えば，小さい凸凹まで測定することができるようになり，小さい物差しを使うほど図形の長さは長くなる．フラクタルにおいて，物差しの長さを ϵ としたとき，測られる図形の長さは ϵ^{-D} に比例する．この D をフラクタル次元と呼び，値が大きいほど図形は複雑であることを意味する．直線のフラクタル次元は 1，コッホ曲線のフラクタル次元は約 1.26 であり，空間を埋め尽くすほど複雑な曲線では 2 に近づく．

本研究では，インド洋東部から太平洋西部にかけての熱帯域を対象に雲のフラクタル次元を求め，その時空間変動を明らかにすることを目的とした．

第2章　研究手法

2.1　解析に用いたデータ

本研究では，気象衛星ひまわりの赤外線画像を緯度・経度座標にマッピングしたデータを使用した．気象衛星ひまわりは，東アジア・西太平洋地域を対象とする日本の静止気象衛星である．地球静止軌道上から，可視および赤外の複数波長帯を用いて撮像観測をおこなっている．解析には，高知大学気象情報頁で公開されている，気象衛星ひまわりの観測した輝度温度を緯度・経度座標にマッピングしたデータを使用した．データは解像度 1800 × 1800 ピクセル，領域は北緯 70° から南緯 20°，東経 70° から 160° の範囲である．本研究では、IR1（10.3〜11.3 μm）の波長帯で観測された輝度温度を使用し，2011 年 1 月 1 日 0 時から 2023 年 12 月 31 日 23 時までの期間について解析をおこなった．図 2.1 に示す西から W1(南緯 12.8 °〜北緯 12.8 °，東経 70 °〜95.6 °)，W2(南緯 12.8 °〜北緯 12.8 °，東経 91.45 °〜117.05 °)，W3(南緯 12.8 °〜北緯 12.8 °，東経 112.9 °138.5 °)，W4(南緯 12.8 °〜北緯 12.8 °，東経 134.35 °〜160 °) と名付けた 4 つの 513 × 513 ピクセルの領域を切り出しそれぞれの領域について，領域内の雲のフラクタル次元を求めた．

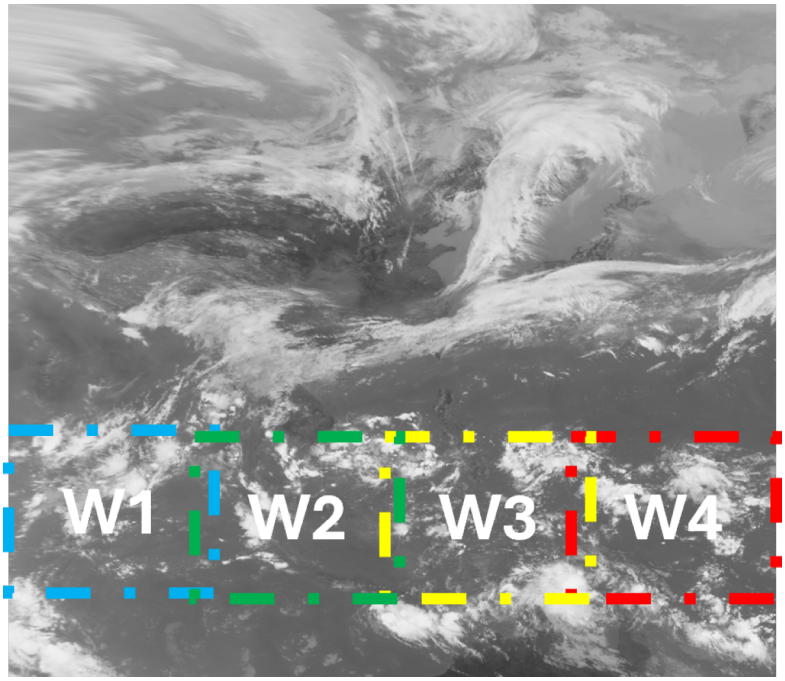


図 2.1: 2025-05-01 03:00(GMT) IR1

2.2　雲の形状の決定方法

ひまわりの IR1 波長帯では、雲や地表から射出された赤外放射が観測される．この波長帯で大気は光学的に薄く雲は光学的に厚いと仮定できるため，観測された放射は雲があれば雲頂，雲がなければ地表から放射されたものであると考えられ，輝度温度は雲頂または地表の温度と考えられる．本研究では，輝度温度を鉛直温度分布と対応させることで，雲頂の気圧を推定した．輝度温度を気圧に変換するときに用いる鉛直温度分布は，気象庁 55 年長期再解析データ JRA-55 を用いた．なお，JRA-55 とひまわりの赤外画像では時間，空間解像度が異なるため，解像度の荒い JRA-55 のデータを内挿して，赤外画像と同一時刻・同一地点鉛直温度分布を求めた．

本研究では，ひまわりが観測した雲頂より下は地表まで雲が存在していると仮定して，各圧力面における雲の断面形状を決定した．ひまわりの赤外画像で見えるのは雲頂だけで，雲頂より下は見ることができない．そのため，ひまわりの赤外画像だけからだと，雲頂より下に雲があるのかないのかを知ることはできない．実在する雲には，積乱雲のように上かあら下までつながっている雲もあるが，巻雲のように上層にだけあってその下に雲がない場合もある．また，複数の雲が異なる高度に存在し，上から見るとそれらが重なっている場合もある．したがって，

上から下までつながった雲が存在すると仮定することは，明らかに正しくない．上から下まで雲が存在すると仮定したことは，高い圧力面における雲の断面形状の決定で問題を生じさせている可能性があることに，注意が必要である．

2.3 フラクタル次元の測定方法

2.3.1 ボックスカウント法

雲のフラクタル次元の算出には，ボックスカウント法を用いた．ボックスカウント法とは，対象とする図形をさまざまな大きさの格子で覆い，図形の境界を含む格子の数，すなわち境界長さが格子サイズによってどのように変化するかを調べることで，フラクタル次元を求める手法である．フラクタル図形の境界長さ N と格子サイズ ϵ, フラクタル次元 D の関係は

$$N \propto \epsilon^{-D} \tag{2.1}$$

に従うことから，境界長さと格子サイズを両対数グラフにプロットすると直線状に並び，その傾きからフラクタル次元を求めることができる．(松下，2002)

2.3.2 本研究の方法

本研究は，ボックスカウント法によってフラクタル次元を測定したが，フラクタル次元を決定するときに，測定をおこなった全ての格子サイズの境界長さを用いるのではなく，３つの格子サイズにおける境界長さのみを用いてフラクタル次元を決定した．これは，実際に格子サイズを変えて境界長さを測定した結果を両対数グラフにプロットすると，ほとんどの場合においてデータがきれいに直線上に並ばないことが多かったためである (図 2.2, 2.3).

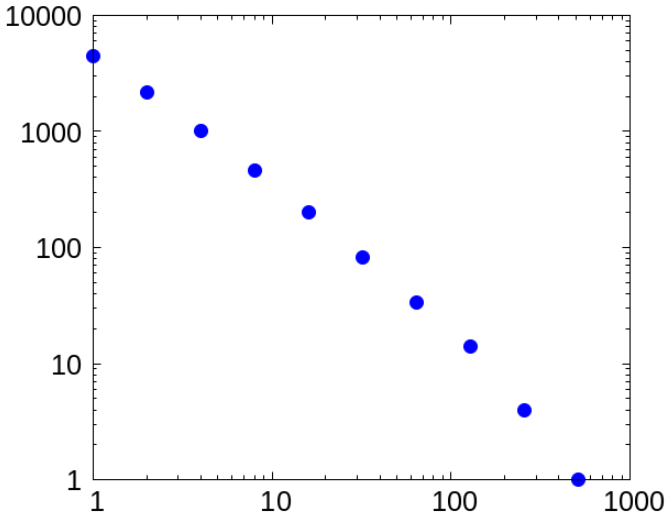


図 2.2: コッホ雪片をボックスカウント法で数えた結果

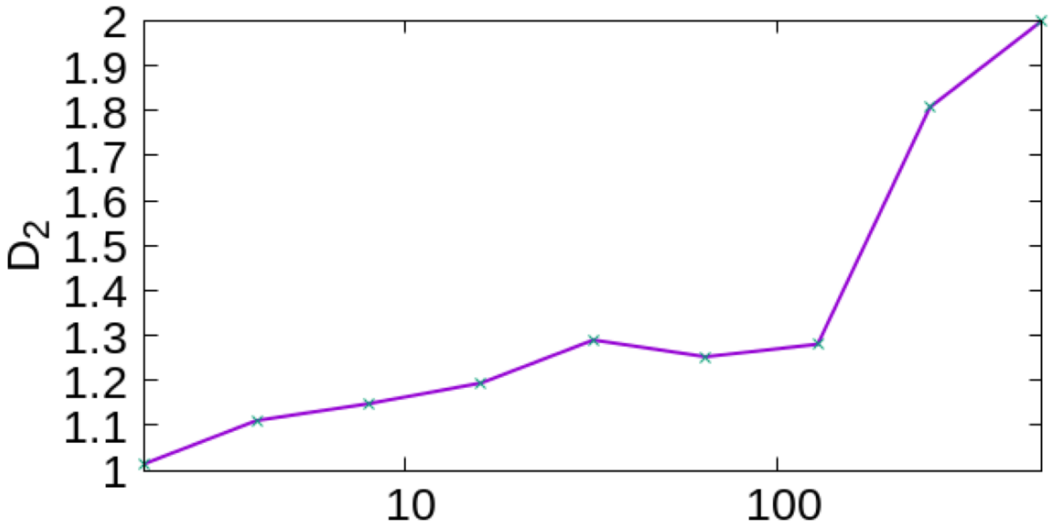


図 2.3: 隣り合う２つの格子サイズだけから求めたフラクタル次元．横軸は用いた２つの格子サイズのうちの小さい方のサイズ，縦軸はフラクタル次元．

図 2.2 は，コッホ雪片をボックスカウント法で数えた結果である．縦軸は境界長さ，横軸は格子の大きさである．一見すると直線に乗っているように見えるが，上に凸な曲線になっている．これは，隣り合う２つの格子サイズの境界長さだけからフラクタル次元を求め (以下，２つの格子サイズの境界長さだけから求めたフラクタル次元を D_2 と書く)，その結果を格子サイズの順番に並べるとはっきりする (図 2.3)．式 (2.1) が成り立つなら，D_2 は格子サイズに依らず一定の値をとるはずだが，ボックスカウント法で求めた結果は，格子サイズを大きくしていくと D_2 が１くらいから２くらいまでほぼ単調に増加していく．これと同様のことは，気象衛星ひまわりの赤外画像でも見られ，Yano and Takeuchi (1987) の結果にも見られる．コッホ雪片のように決まったフラクタル次元を持つ図形に対して，図 2.3 のようにフラクタル次元が格子サイズによって変化する原因のひとつとして，格子サイズが大きくなるにつれて，ほぼすべての格子が雲の境界を含むようになることが考えられる．すべての格子が雲の境界を含むとき，ボックスカウント法で測定されるフラクタル次元は 2 になる．このことが，格子サイズを大きくしたときにフラクタル次元が大きくなった原因と考えた．

本研究では，フラクタル次元を求めるにあたって，全ての格子サイズの境界長

さを用いるのではなく，精度よくフラクタル次元を求めることができると推定される格子サイズだけを用いることにした．用いる格子サイズの抽出は D_2 を用いて以下の方法によっておこなった．まず，観測の影響を強く受ける最も小さい格子サイズと必ず値が 1 になる最も大きい格子サイズを使って求めた D_2 を除外した．次に D_2 が 1 より小さいものと，1.9 より大きいものも除外した．これは，はフラクタル次元は 1 未満や 2 より大きい値をとらないからである．その後，隣り合う D_2 の差が最も小さくなる部分を選び，隣り合う D_2 を計算するのに用いられた連続する 3 つの格子サイズにおける境界長さを用いてフラクタル次元を求めた．以上の手順によって，もっとも直線に近い 3 つの格子サイズの境界長さからフラクタル次元が得られる．

抽出した 3 つの格子サイズにおける境界長さが両対数プロットでより直線に近く配置されるときにフラクタル次元が精度よく求まると考えるなら，抽出した 3 つの格子サイズから計算される 2 つの D_2 の差が小さいほどフラクタル次元は精度よく求まることになる．反対に，D_2 の差が大きいほど精度は悪くなる．そこで，抽出した 3 つの格子サイズから計算される 2 つの D_2 の差がフラクタル次元の測定精度に及ぼす影響を見るため，2 つの D_2 の差に閾値を設けて画像を選別し，推定したフラクタル次元をプロットした (図 2.4-2.9)．

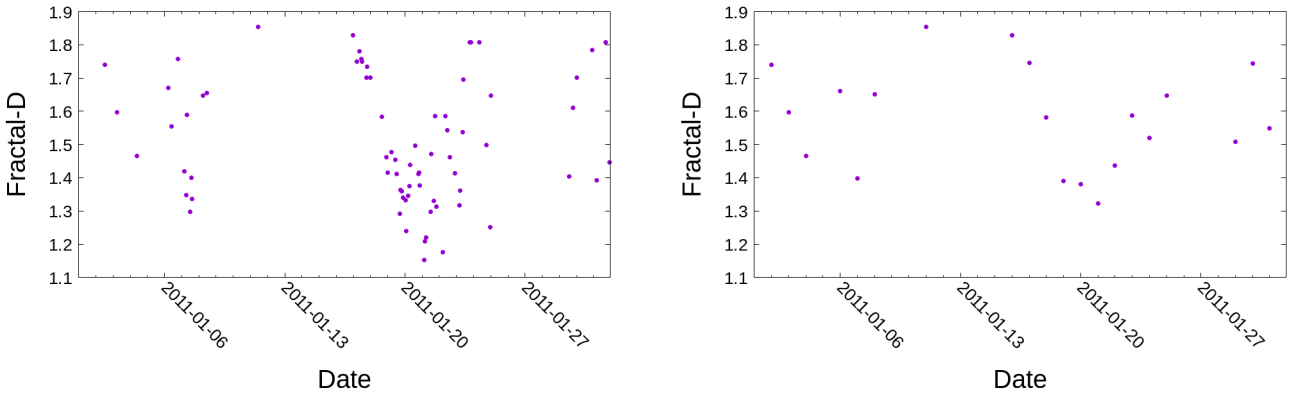


図 2.4: D_2の差 < 0.01

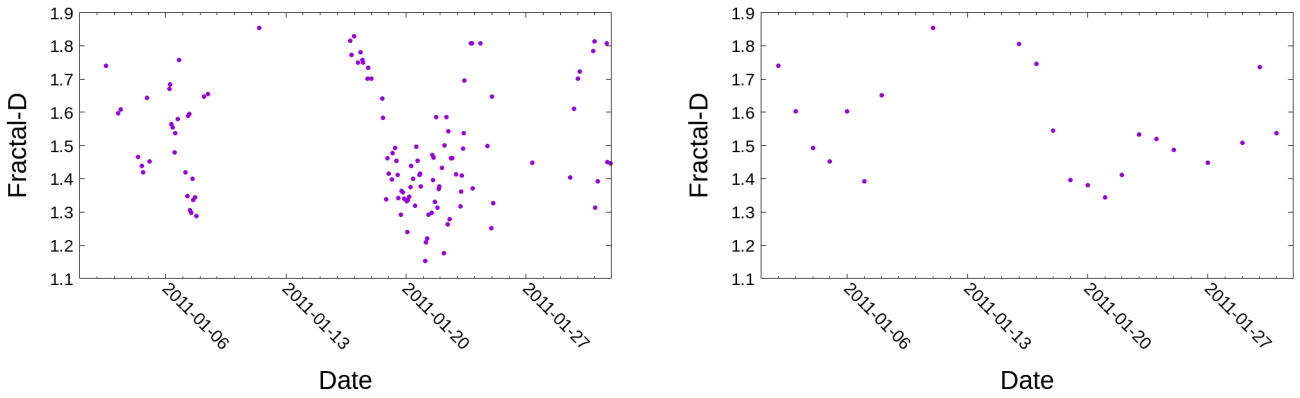


図 2.5: D_2の差 < 0.02

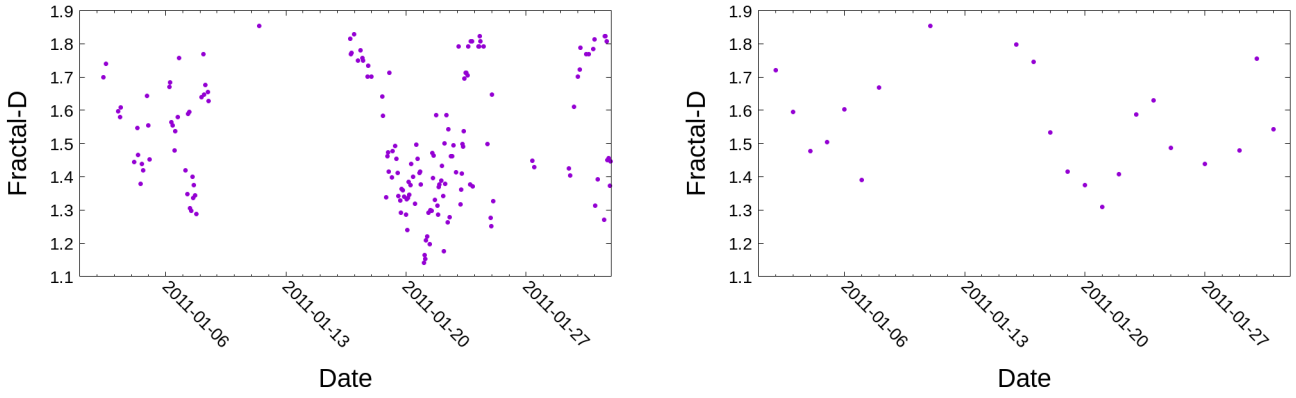


図 2.6: D_2の差 < 0.03

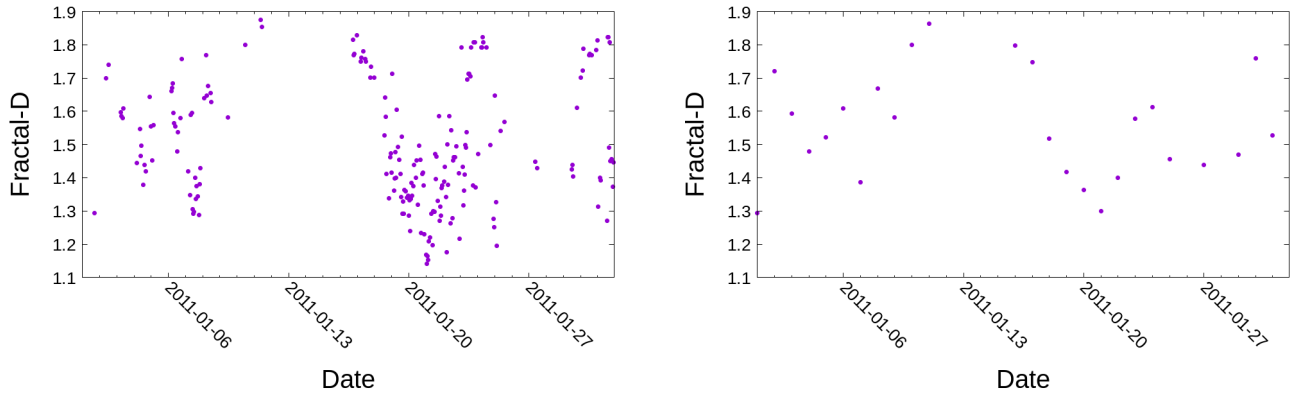


図 2.7: D_2の差 < 0.04

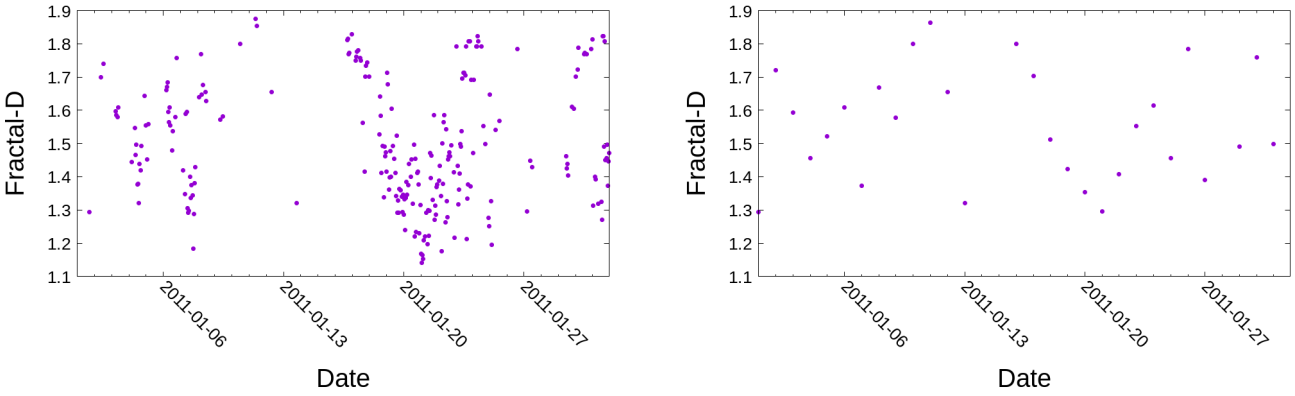


図 2.8: D_2の差 < 0.05

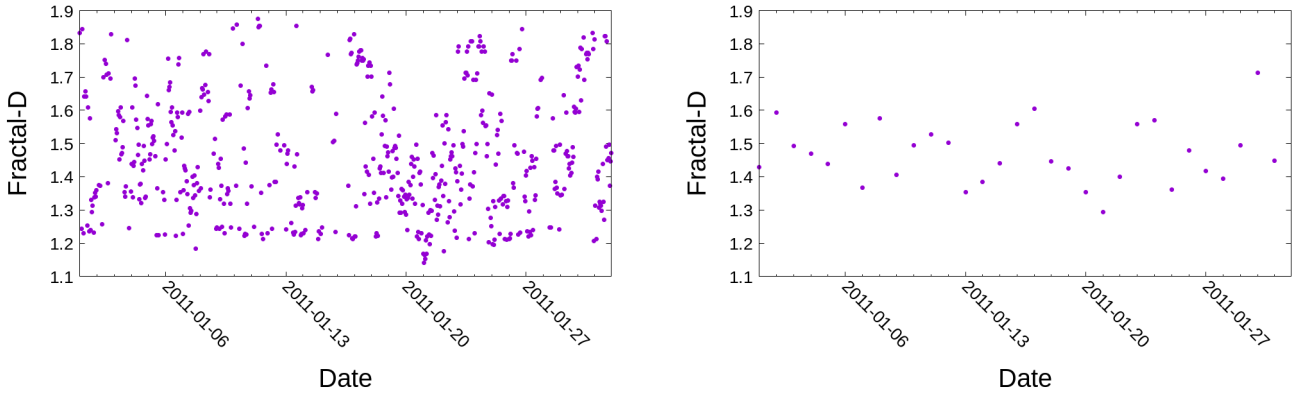


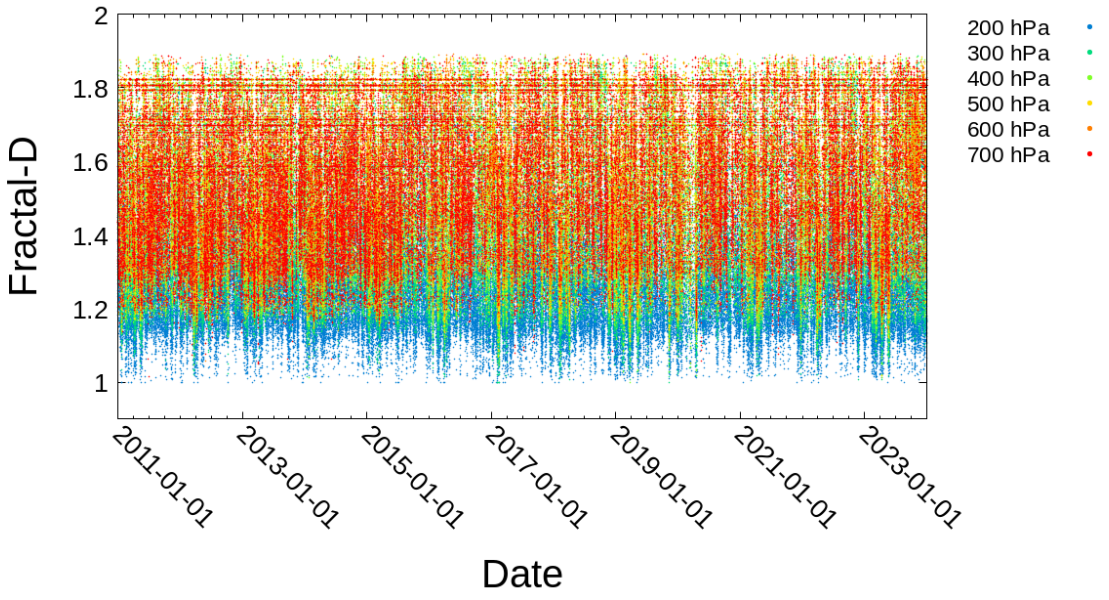
図 2.9: D_2の差 < 0.1

図 2.4 から 2.9 は，2011 年 1 月の各画像で測定されたフラクタル次元である．左は各時刻の画像から求めたフラクタル次元，右は各時刻の画像から求めたフラクタル次元を日毎に平均したもの，それぞれをプロットした．横軸は日付，縦軸はフラクタル次元である．D_2 の差が小さいほど，フラクタル次元が精度よく求まっているとするなら，図 2.4 がもっとも正解に近いフラクタル次元を求めたものということになる．これに対して図 2.9 の D_2 の差が 0.1 以下のフラクタル次元は，図 2.4 のフラクタル次元と分布が異なっており，D_2 の差が大きいと正しいフラクタル次元が求まらないことを示唆しているように見える．一方，図 2.8 の D_2 の差が 0.05 以下のものは，図 2.4 の 0.01 以下のものと比べて，その分布はあまり変わらないように見え，D_2 の差が大きいものを含むことの影響は小さいように見える．そこで本研究では D_2 の差が 0.05 以下のものだけを用いることとし，D_2 の差が 0.05 より大きいものは除外することとした．

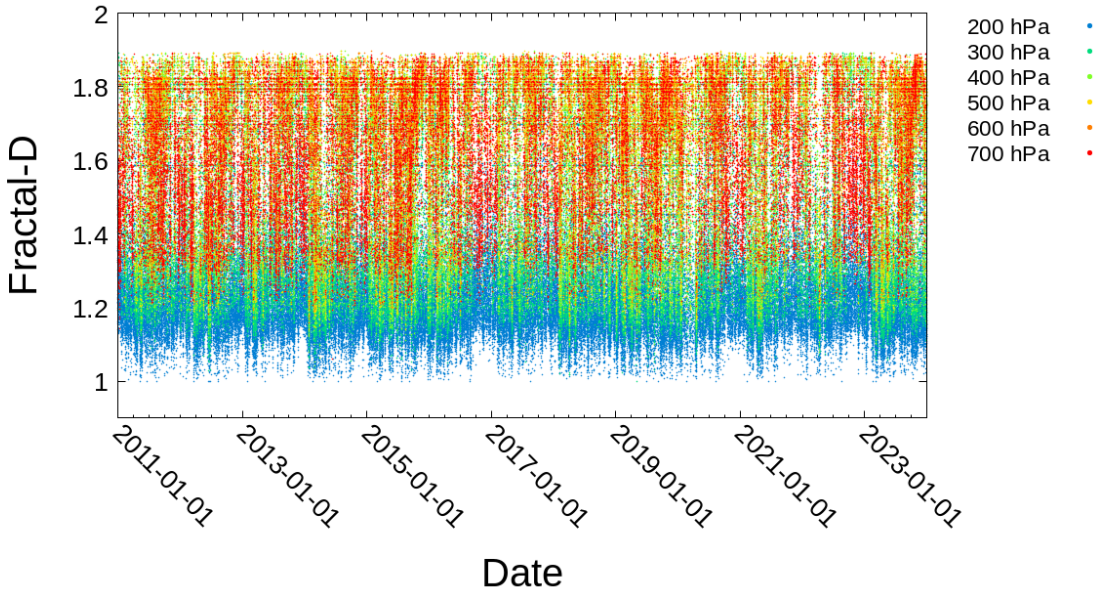
第3章　結果

3.1　時系列データ

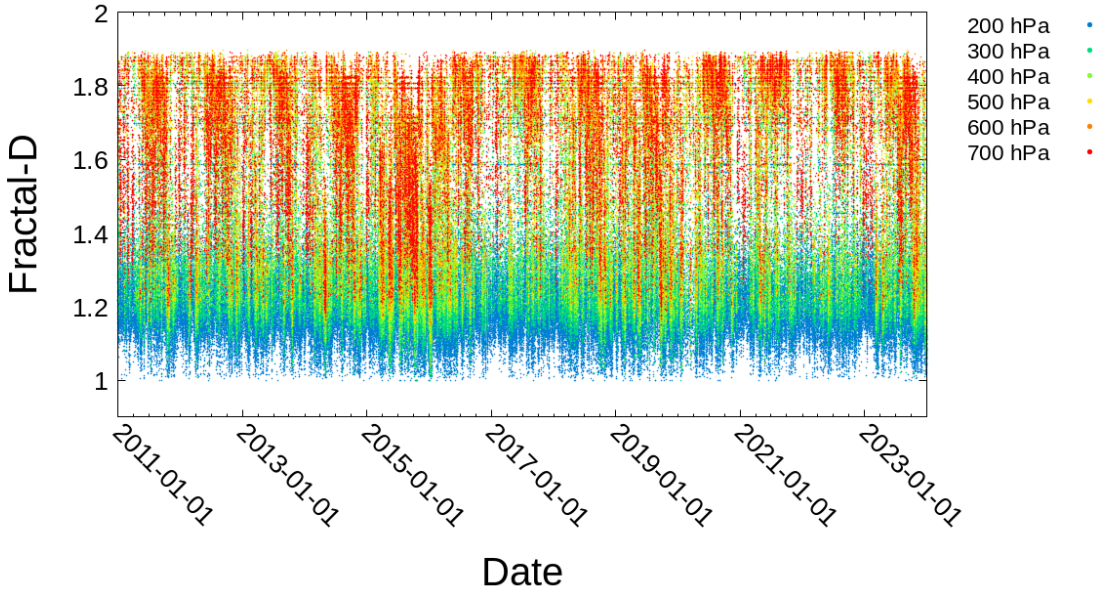
図 3.1 は，各時刻の赤外画像から求めたフラクタル次元である．図 3.2 は，各時刻の画像から求めたフラクタル次元を月毎に平均したものである．圧力が低くなる，すなわち高度が高くなるほどフラクタル次元が小さくなる傾向が見て取れる．この傾向は W1 から W4 全ての領域に共通してみられる．ただし，雲頂より下の雲が見えていないため，高い圧力面におけるフラクタル次元は正しくない可能性がある．圧力面によってフラクタル次元が変化する傾向は，雲頂より下の雲が見えていないことによる人工的なものである可能性があることに，注意する必要がある．



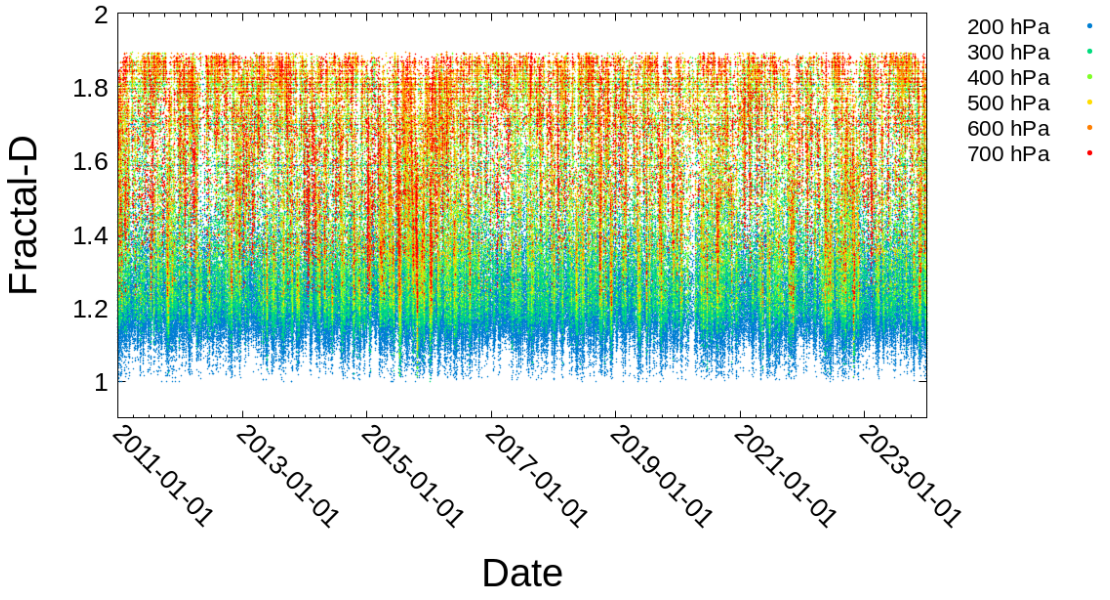
(A)


Fractal-D
2
1.8
1.6
1.4
1.2
1
2011-01-01
2013-01-01
2015-01-01
2017-01-01
2019-01-01
2021-01-01
2023-01-01
Date
200 hPa
300 hPa
400 hPa
500 hPa
600 hPa
700 hPa

(B)

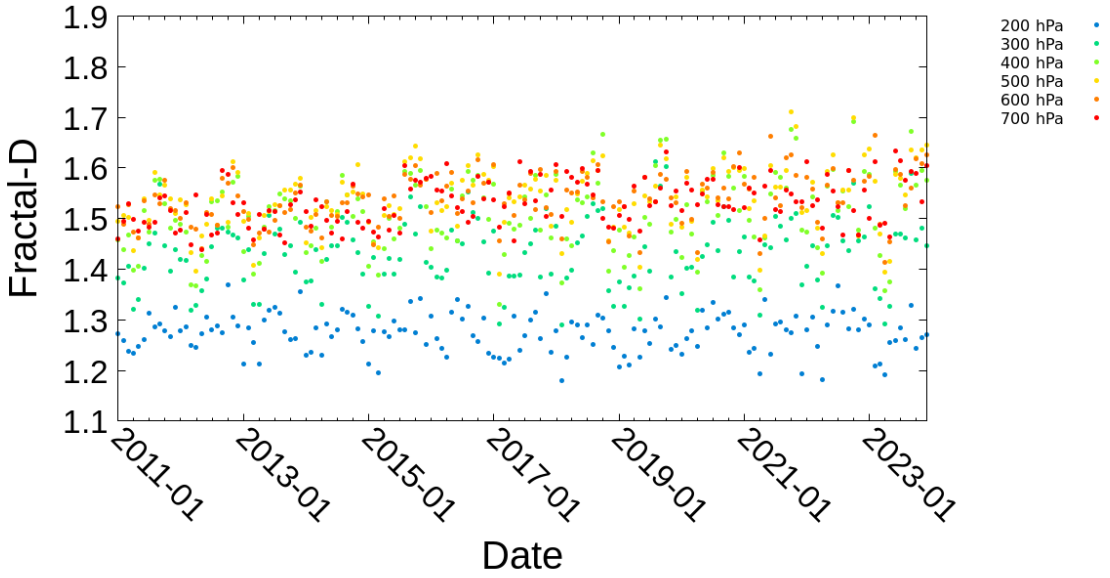

Fractal-D
2
1.8
1.6
1.4
1.2
1
2011-01-01
2013-01-01
2015-01-01
2017-01-01
2019-01-01
2021-01-01
2023-01-01
Date
200 hPa
300 hPa
400 hPa
500 hPa
600 hPa
700 hPa

(C)

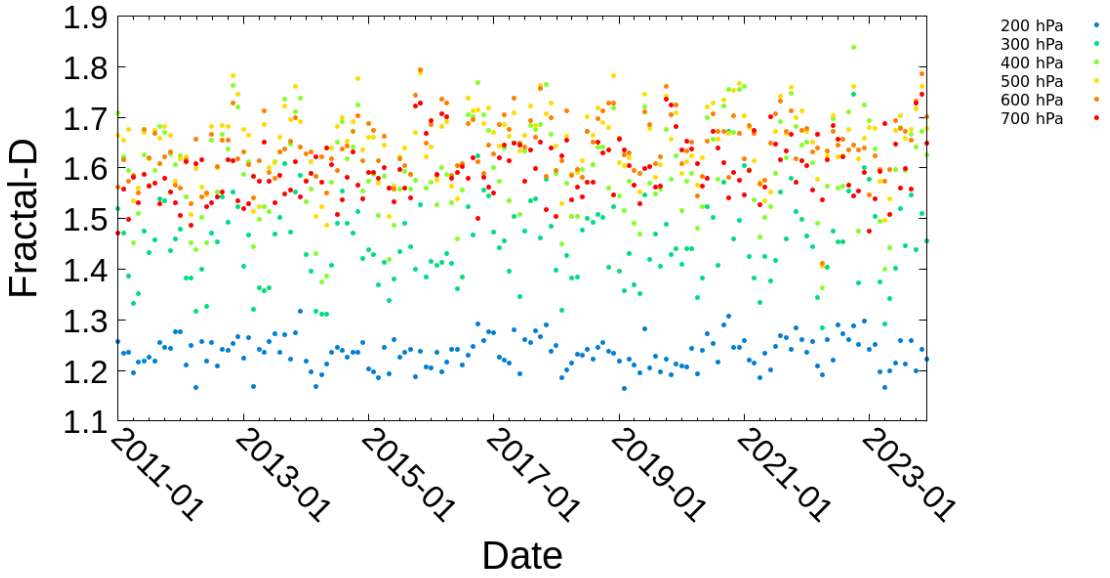


(D)

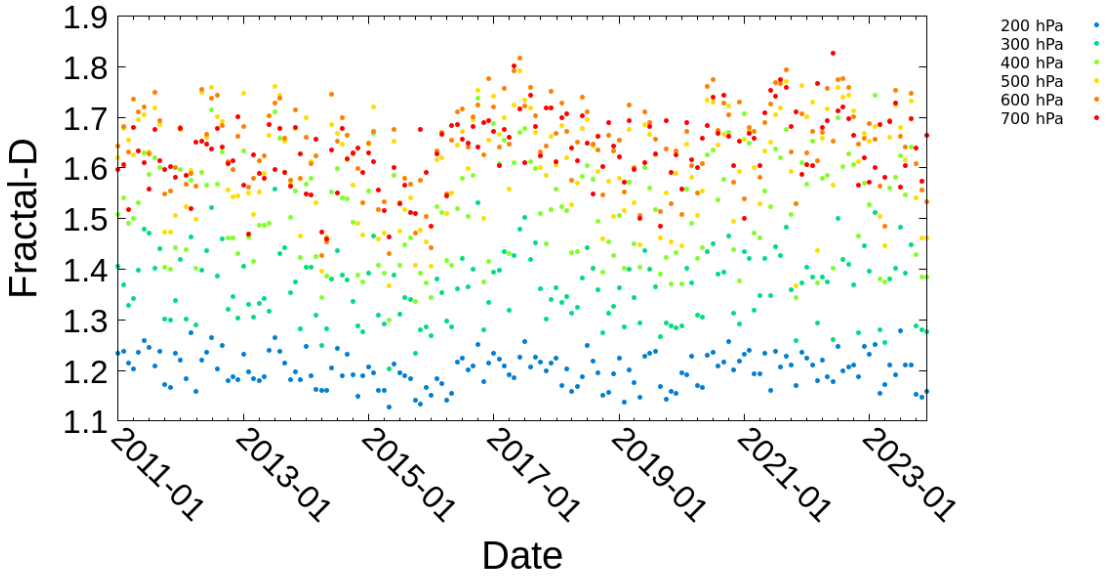
図 3.1: ひまわり赤外画像 IR1 から求めたフラクタル次元．縦軸がフラクタル次元，横軸が時間．（A）W1,（B）W2,（C）W3,（D）W4.



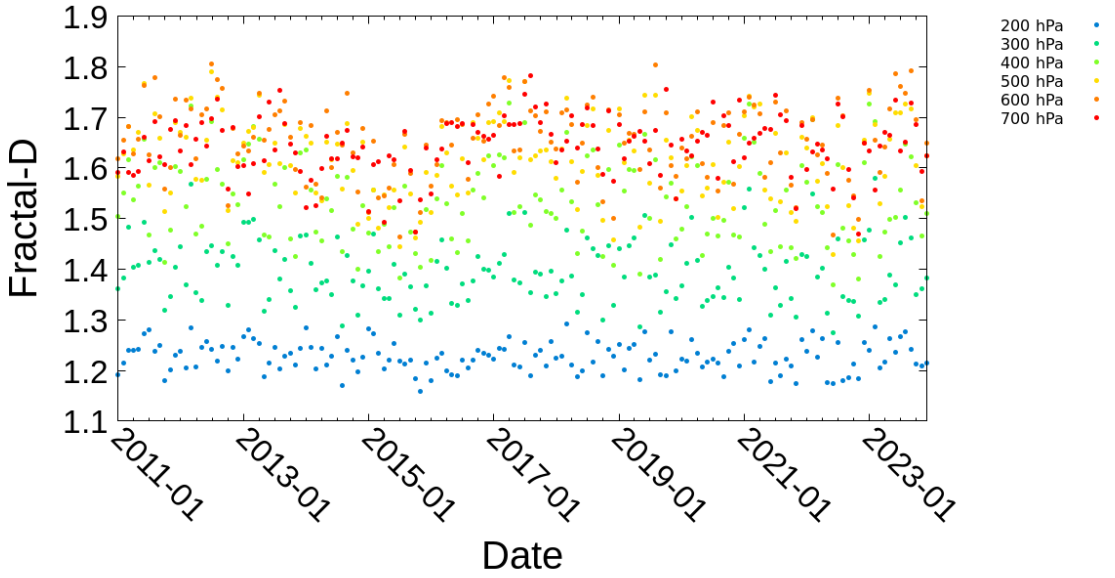
(A)


Fractal-D
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
2011-01
2013-01
2015-01
2017-01
2019-01
2021-01
2023-01
Date
200 hPa
300 hPa
400 hPa
500 hPa
600 hPa
700 hPa

(B)


Fractal-D
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
2011-01
2013-01
2015-01
2017-01
2019-01
2021-01
2023-01
Date
200 hPa
300 hPa
400 hPa
500 hPa
600 hPa
700 hPa

(C)



(D)

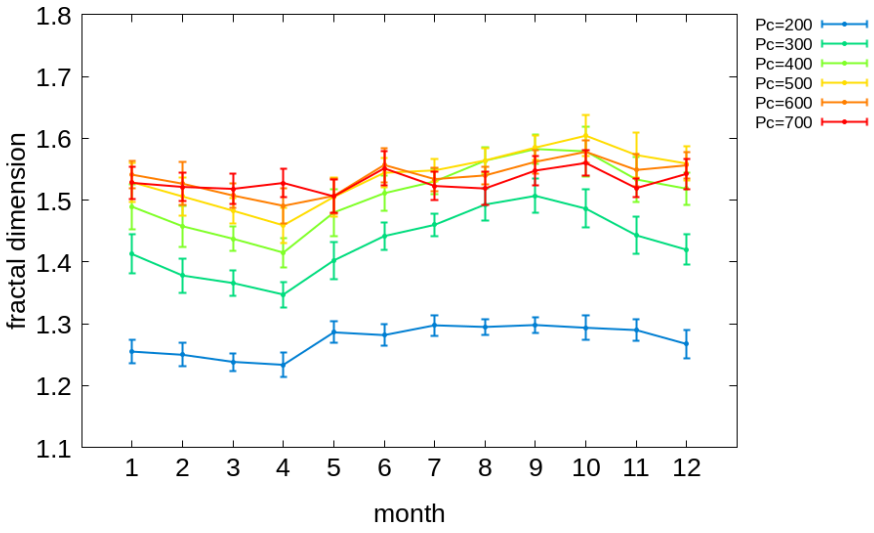
図 3.2: ひまわり赤外画像 IR1 から求めた月平均のフラクタル次元．縦軸がフラクタル次元，横軸が時間．(A)W1, (B)W2, (C)W3, (D)W4

3.2 季節変化

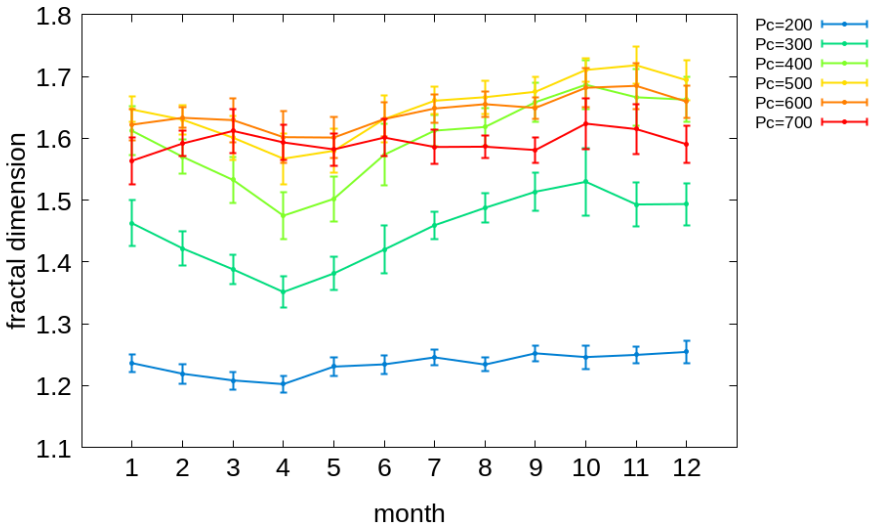
月平均したフラクタル次元をさらに各月ごとに平均した結果を図 3.3 に示す．エラーバーは式 (3.1) で求めた 95%信頼区間を表している．

$$95\%\text{信頼区間} = [x - 1.96 \times \sqrt{s^2 \div n}, x + 1.96 \times \sqrt{s^2 \div n}] \tag{3.1}$$

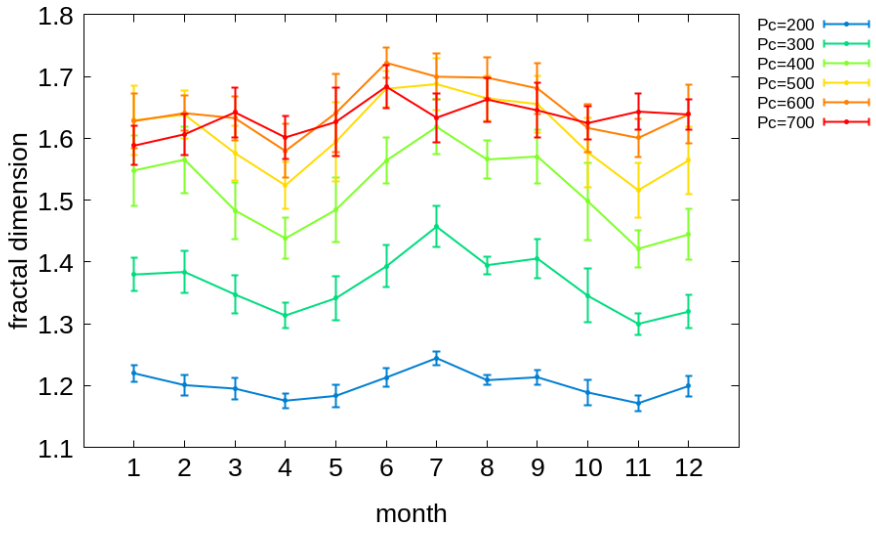
ここで x は月平均の平均，s^2 は不偏分散，n は平均した月数である．一部の圧力面，特に 300hPa の圧力面におけるフラクタル次元には，4-5 月頃に小さくなり 7-10 月頃に大きくなるという季節変化が見られた．


fractal dimension
month
Pc=200
Pc=300
Pc=400
Pc=500
Pc=600
Pc=700

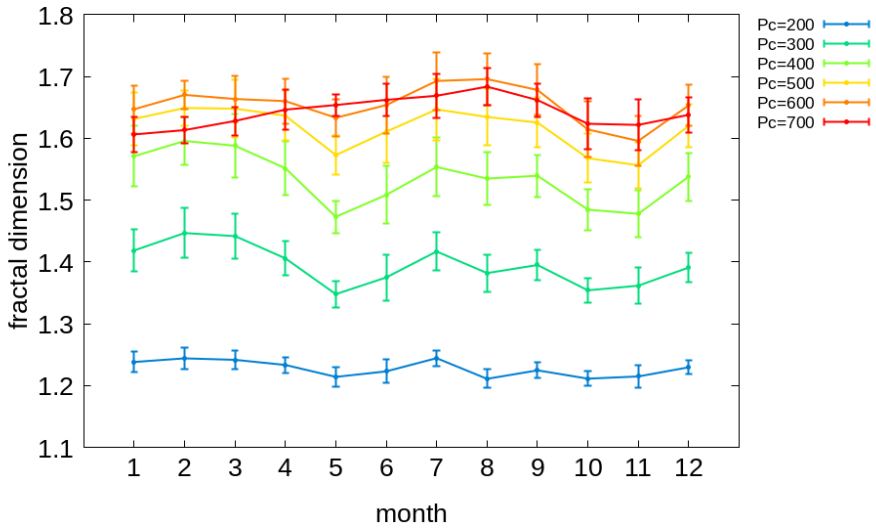
(A)


fractal dimension
month
Pc=200
Pc=300
Pc=400
Pc=500
Pc=600
Pc=700

(B)



(C)

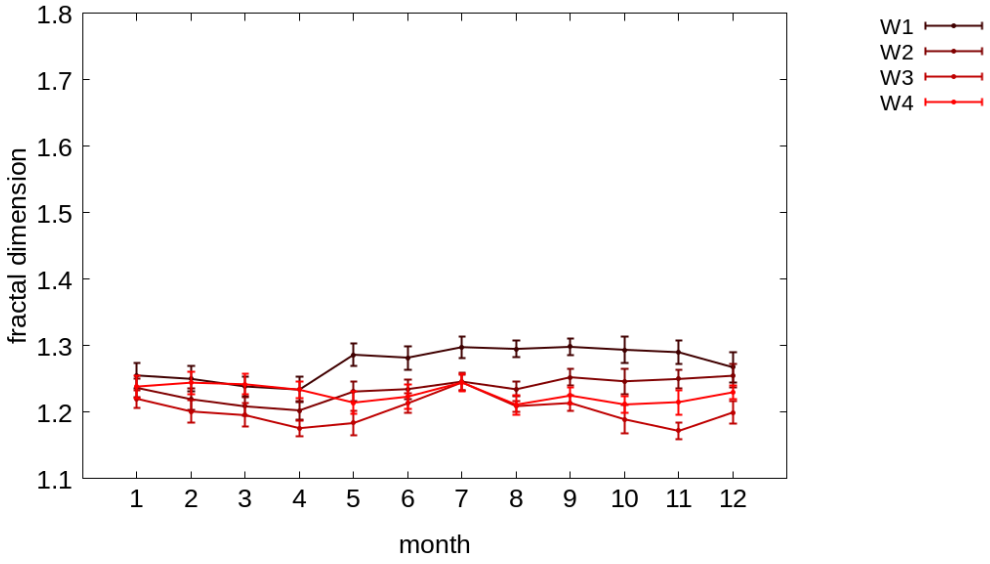


(D)

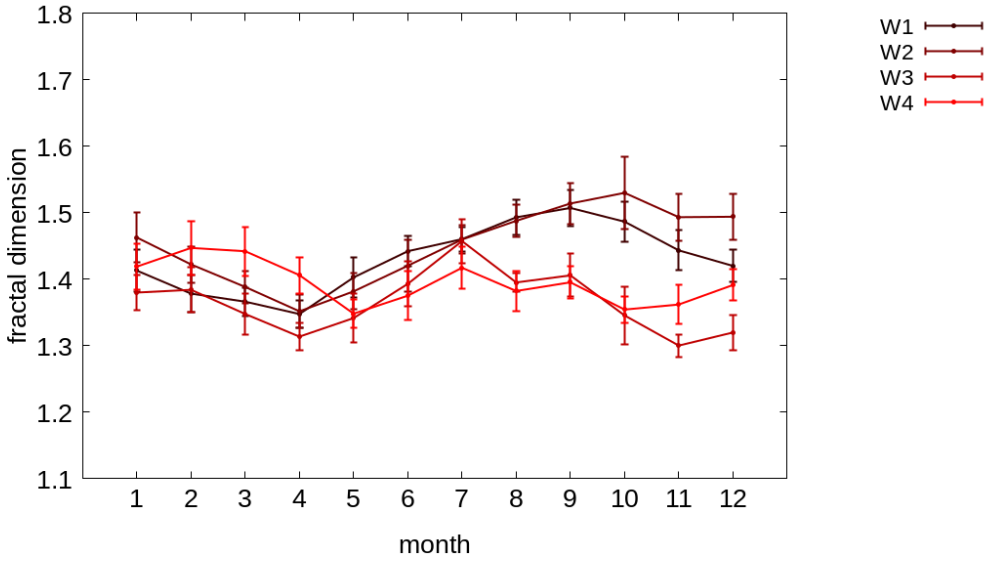
図 3.3: フラクタル次元の季節変化．縦軸がフラクタル次元，横軸が月．(A)W1, (B)W2，(C)W3，(D)W4.

3.3 領域依存性

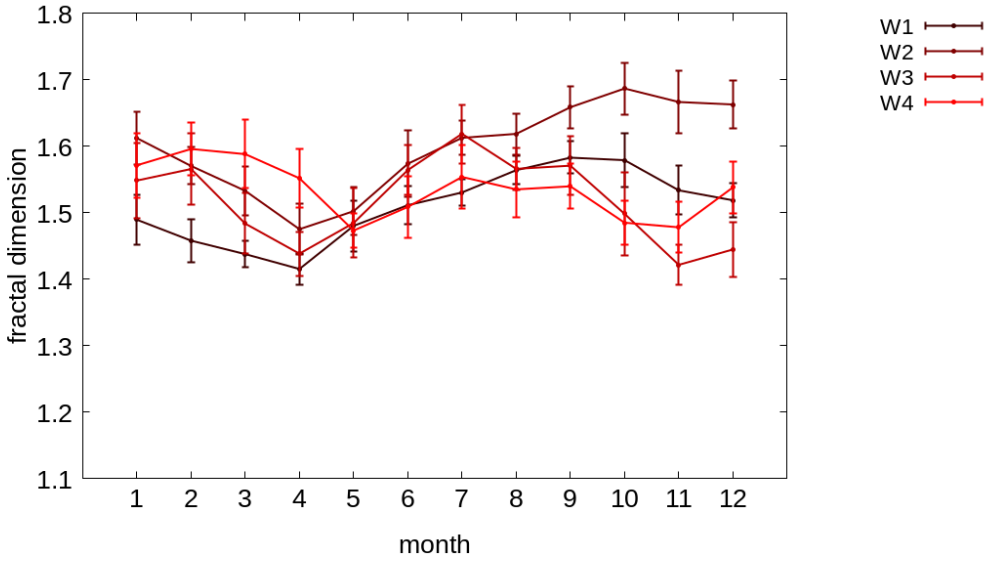
図 3.4 は，各圧力面ごとのフラクタル次元の季節変化である．



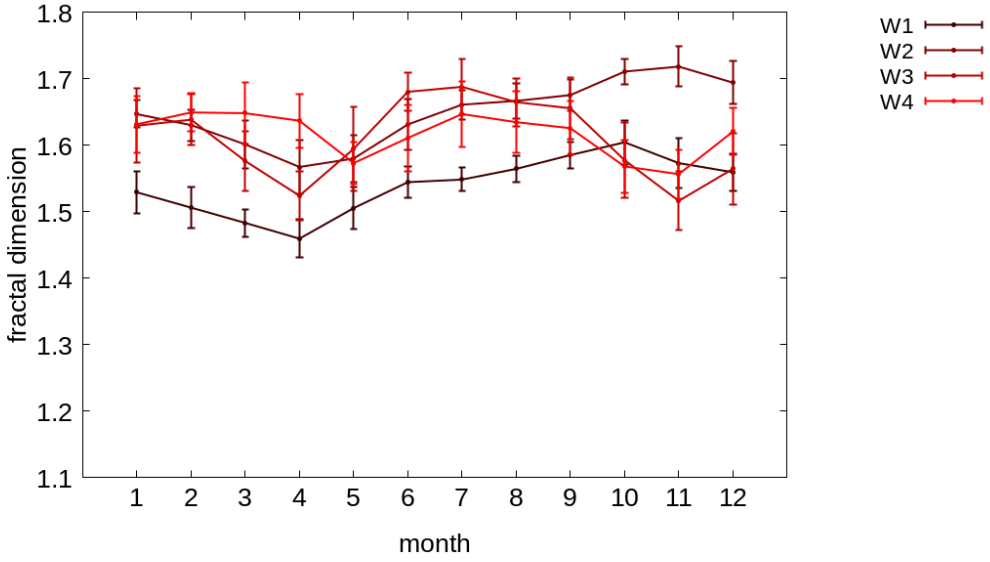
(A)



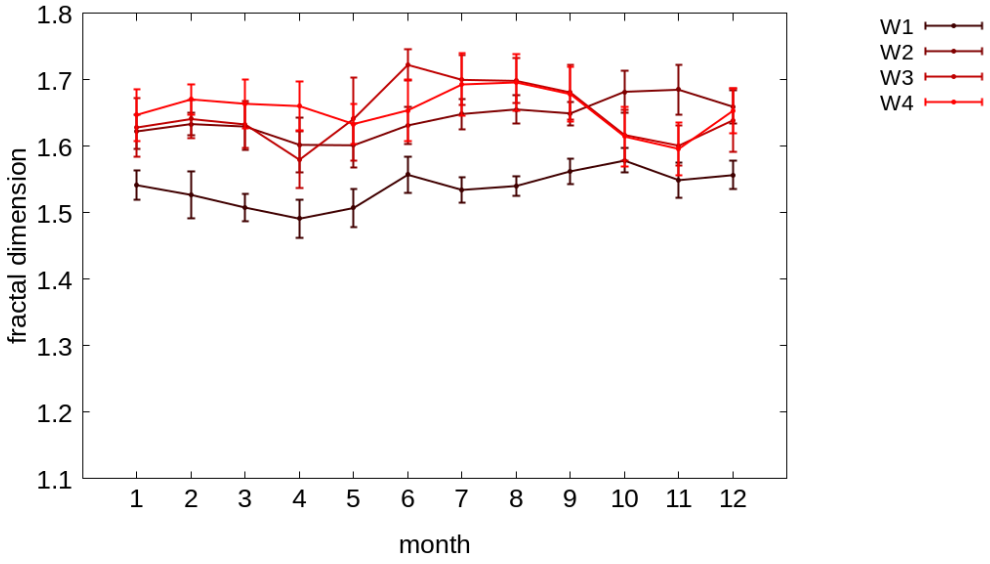
(B)


fractal dimension
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1 2 3 4 5 6 7 8 9 10 11 12
month
W1
W2
W3
W4

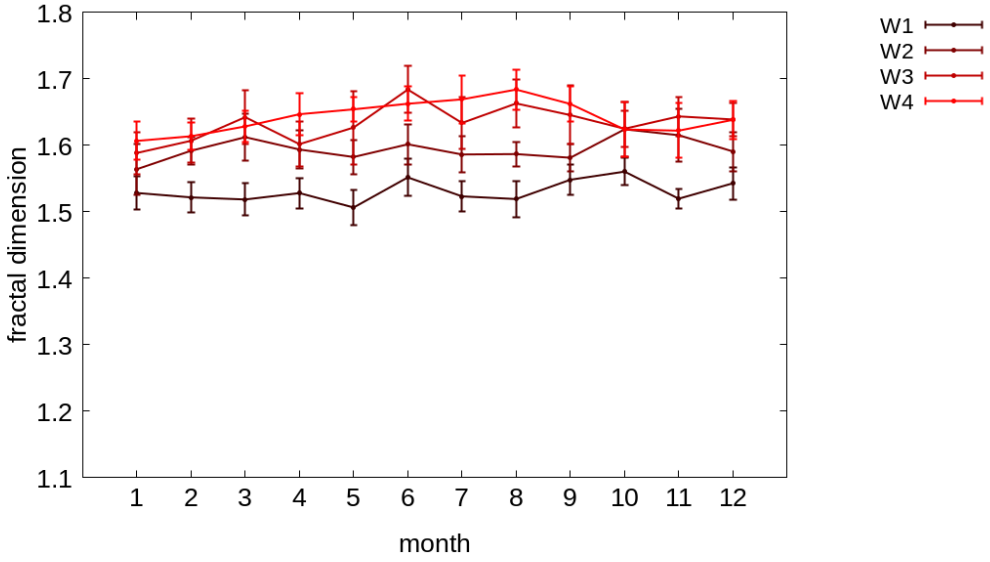
(C)


fractal dimension
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1 2 3 4 5 6 7 8 9 10 11 12
month
W1
W2
W3
W4

(D)



(E)

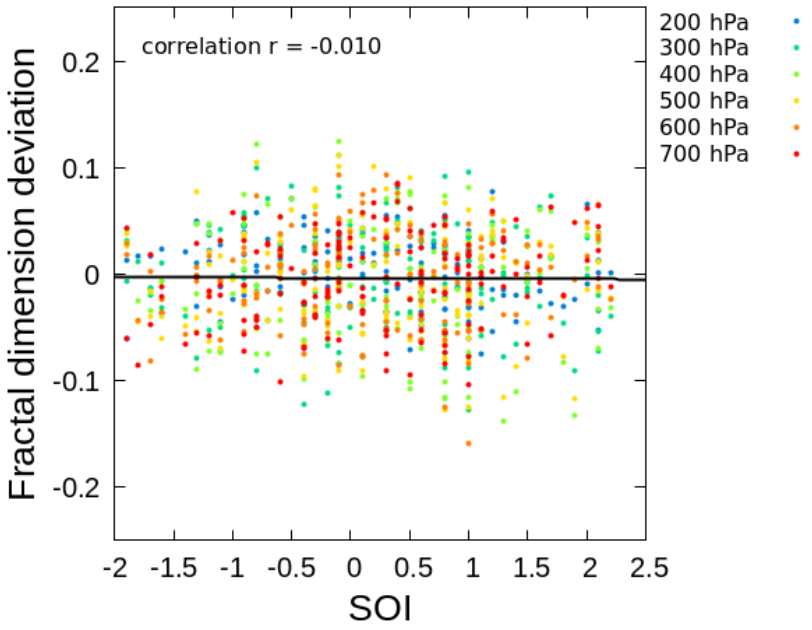


(F)

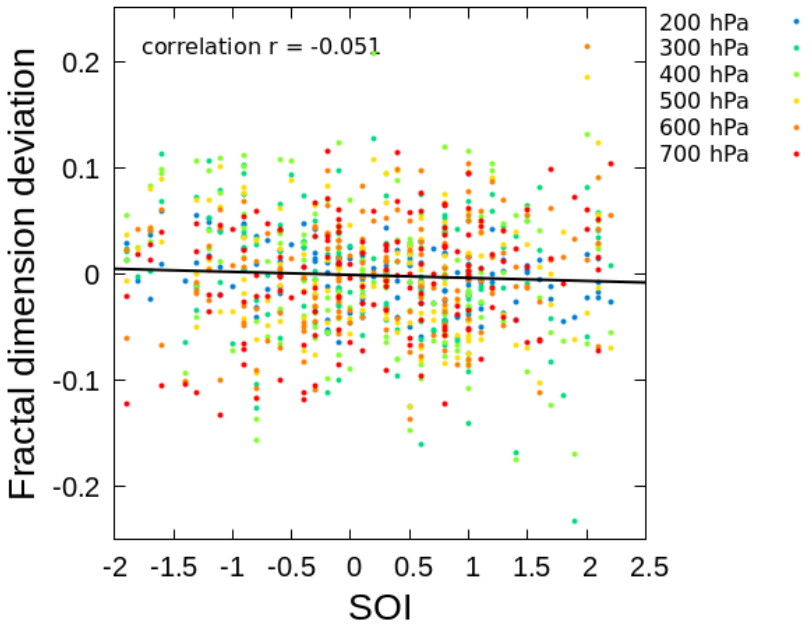
図 3.4: フラクタル次元の季節変化．縦軸がフラクタル次元，横軸が月．(A) 圧力面 = 200hPa，(B) 圧力面 = 300hPa，(C) 圧力面 = 400hPa，(D) 圧力面 = 500hPa，(E) 圧力面 = 600hPa，(F) 圧力面 = 700hPa.

3.4 ENSO

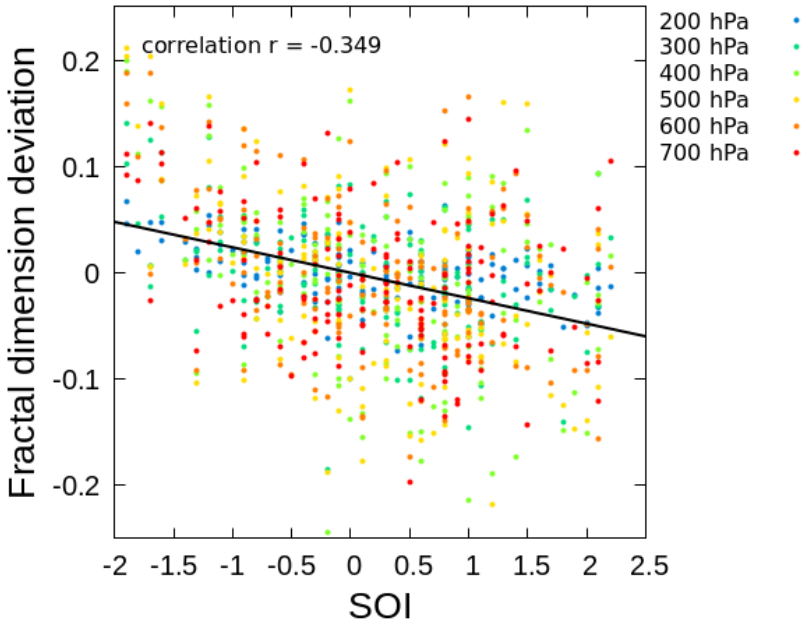
図 3.5 は，解析期間における各月の SOI(南方振動指数) とフラクタル次元の偏差の関係をプロットしたものである．フラクタル次元の偏差は，月平均のフラクタル次元から，月平均の平均 (図 3.3) を引いたものとして定義した．SOI はエルニーニョのフェーズを表す指標で，ここでは気象庁のデータ (`https://www.data.jma.go.jp/cpd/data/elnino/index/soi.html`) を用いた．領域 W3 では負の相関があるように見えるが，その他の領域では明確な相関があるようには見えない．



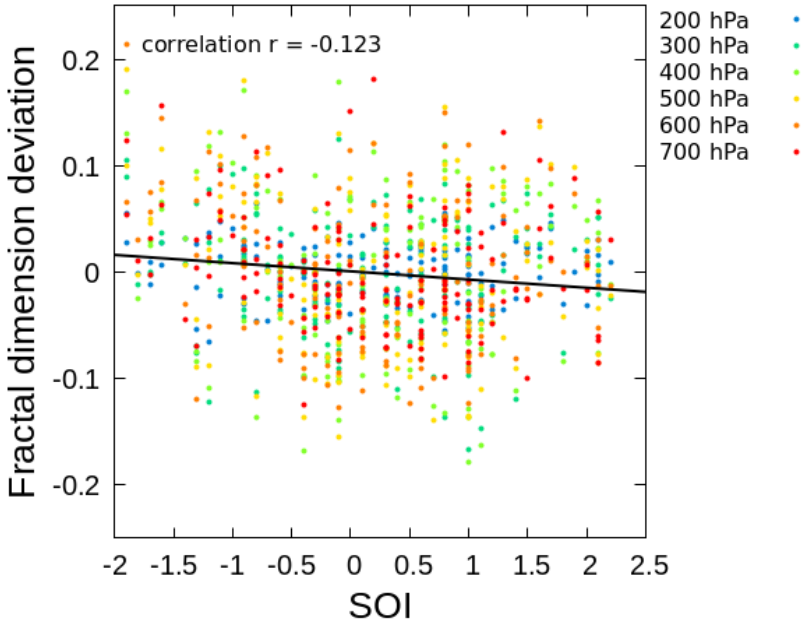
(A)


correlation r = -0.051
Fractal dimension deviation
SOI
200 hPa
300 hPa
400 hPa
500 hPa
600 hPa
700 hPa
0.2
0.1
0
-0.1
-0.2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5

(B)


correlation r = -0.349
Fractal dimension deviation
SOI
200 hPa
300 hPa
400 hPa
500 hPa
600 hPa
700 hPa
0.2
0.1
0
-0.1
-0.2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5

(C)



(D)

図 3.5: 縦軸がフラクタル次元の偏差，横軸が SOI，黒の実線は回帰直線.（A）W1,（B）W2,（C）W3,（D）W4.

第4章　まとめ

雲の水平断面形状のフラクタル次元を求め，その時空間変動を解析した．その結果，雲のフラクタル次元には季節変化が存在した．また，一部の領域ではエルニーニョとの関係が示唆された．

謝辞

本研究を進めるにあたって，ご指導いただきました主指導教官であるはしもとじょーじ教授には心より深く感謝申し上げます.

参考文献

[1] Lovejoy, S., 1982: Area-perimeter relation for rain and cloud area. Science, 216,185 -187.

[2] Yano and Takeuchi 1987: The Self-Similiarity of Horizontal Cloud Pattern in the Intertropical Convergence Zone. Journal of the Meteorological Society of Japan, 65(4),661-667.

[3] 松下 貢. フラクタルの物理 (1). 裳華房. 2002

[4] 高知大学気象情報頁
http://weather.is.kochi-u.ac.jp/

[5] 気象庁，南方振動指数
https://www.data.jma.go.jp/cpd/data/elnino/index/soi.html

付録

雲の境界長さ

雲の境界長さ求めるときに用いたソースコードを載せる.

プログラム 1: conf.f

```
      integer sYEAR, eYEAR
      integer sMONTH, eMONTH
      integer sDAY, eDAY
      integer sHOUR, eHOUR
      parameter(sYEAR = 11, eYEAR = 11)
      parameter(sMONTH = 01, eMONTH = 01)
      parameter(sDAY = 01, eDAY = 01)
      parameter(sHOUR = 00, eHOUR =00)

      integer lower_end
      parameter(lower_end = 0)

      real Pc
      parameter(Pc = 600) !hPa

      real Tc
      parameter(Tc = 278) !K

      integer number
      parameter (number =10)

      real pie
      parameter (pie = 2.0 * asin(1.0))

      character(len=3) wave
      parameter (wave = 'IR1')
```

プログラム 2: Makefile

```
TARGET = do

FC = gfortran

```

```
5  DIR = include/
6
7  FILES = \
8          $(DIR)temp_conv.o\
9          $(DIR)bound.o\
10         $(DIR)average.o\
11         $(DIR)Box.o
12
13 .SUFFIXES: .f90 .o
14
15 .f90.o:
16         $(FC) -c $< -o $@
17
18 $(TARGET): $(FILES)
19         $(FC) -o $@ $^
20
21 clean:
22         rm -f $(DIR)*.o *.mod *~
```

プログラム 3: Box.f90

```
1  program fractal
2
3    use temp_conv
4    use bound
5    use average
6
7    implicit none
8    include '../conf.f'
9
10   integer :: istat
11
12   integer :: i, j, n, l
13   integer :: resolution
14   integer :: y, m, d, h, temp
15   integer :: size_C, size_G
16   integer :: i_e, j_e
17   integer,dimension(1800,1800) :: val
18
19   real, dimension(0:255) :: conv
20   real, allocatable :: T(:,:)
21   real, allocatable :: Tin(:,:)
22   real, allocatable :: Tout(:,:)
23   real, allocatable :: Na(:,:)
24
25   character(len=2) :: yy, mm, dd, hh
26   character(len=3) :: sp, st
```

```
27    character(len=200) :: GAME, CAL
28    character(len=200) :: Fractal_D
29    character(len=200) :: chead
30
31    logical :: exist_G, exist_C
32
33    write(st,'(I3.3)')int(Tc)
34    write(sp,'(I3.3)')int(Pc)
35
36    allocate(Na(4,number))
37
38    do y = sYEAR, eYEAR
39       write(yy, '(I2.2)') y
40       do m = sMONTH, eMONTH
41          write(mm, '(I2.2)') m
42          do d = sDAY, eDAY
43             if ((m==2 .and. d==29 .and. mod(y,4)/=0) .or. &
44                  (m==2 .and. d>=30) .or. &
45                  ((m==4 .or. m==6 .or. m==9 .or. m==11) .and. d
                        ==31)) cycle
46             write(dd, '(I2.2)') d
47             do h = sHOUR, eHOUR
48                write(hh, '(I2.2)') h
49
50                GAME = '../data/GAME/'//trim(wave)//'/20' // yy //
                       '/' // mm // '/' // yy // mm // dd // hh //
                       trim(wave) // '.pgm'
51                CAL = '../data/CAL/20' // yy // '/' // mm // '/'
                       // yy // mm // dd // hh // 'CAL.dat'
52
53                if (lower_end == 0)then
54                   Fractal_D='../result/'//trim(wave)//'/20'//yy
                          //'/'// mm //'/Tc'//st//'/B'//yy//mm//dd//hh
                          //'_'//st//'.txt'
55                else
56                   Fractal_D='../result/'//trim(wave)//'/20'//yy
                          //'/'// mm //'/Pc'//sp//'/B'//yy//mm//dd//hh
                          //'_'//sp//'.txt'
57                end if
58
59                inquire(file=GAME, exist=exist_G, size=size_G)
60                inquire(file=CAL, exist=exist_C, size=size_C)
61
62                if (.not. exist_G .or. size_G < 12000000 .or. .not.
                       exist_C .or. size_C == 0) then
63                   open(30, file=Fractal_D)
```

```
64                   do n=1,4
65                      write(30,*) n, (0.0, i=1,number)
66                   end do
67                   close(30)
68
69 ! write(*,*) '20' // yy // '-' // mm // '-' // dd // '-' // hh
       , ' box done'
70                   cycle
71                end if
72
73                open(10, file=GAME)
74                open(20, file=CAL)
75
76                do
77                   read(20,*) chead
78                   if (chead == trim(wave))exit
79                end do
80
81                do i = 0, 255
82                   read(20,*) chead, chead, chead, chead, chead,
                         conv(i)
83                end do
84
85                do i = 1, 5
86                   read(10,*) chead
87                end do
88
89                do j = 1, 1800
90                   read(10,*) (val(i,1801-j), i = 1, 1800)
91                end do
92
93                close(10)
94                close(20)
95
96                allocate(T(1800,1800))
97
98                call temp_conversion(conv, val, lower_end, Pc, Tc, T
                      , y, m, d, h)
99
100 ! open(10,file='t.pgm')
101 ! write(10,'(A)')'P2'
102 ! write(10,*) 1536,1536
103 ! write(10,*) 1
104 ! do j=1,1536
105 ! write(10,*)(int(T(i,j)), i=1,1536)
106 ! end do
```

```
107 ! close(10)
108
109               open(10, file = 'cloud.txt')
110               do i = 1, 1800
111                  do j = 1, 1800
112                     write(10,*)i, j, T(i,j)
113                  end do
114               end do
115               close(10)
116
117               allocate(Tin(1799,1799))
118               call bound_counting(T, Tin)
119
120               deallocate(T)
121               allocate(T(1799,1799))
122
123               T = Tin
124               l = 0
125               Na = 0
126
127               do n = 1, 4
128                  do i = 1, 512
129                     l = l + 1
130                     do j=1, 512
131                        if(T(l,j+144) == -999)then
132                           Na(n,:) = -999
133                           exit
134                        end if
135                        Na(n,1) = Na(n,1) + T(l,j+144)
136                     end do
137                     if(Na(n,1) == -999)exit
138                  end do
139                  if(Na(n,1) == -999)exit
140                  l = l - 83
141               end do
142
143               l = 0
144               do n = 1, 4
145
146                  if(Na(n,1) == -999)exit
147
148                  deallocate(Tin)
149                  allocate(Tin(512,512))
150
151                  do i = 1, 512
152                     l = l + 1
```

```
153                     do j = 1, 512
154                        Tin(i,j) = T(l,j+144)
155                     end do
156                  end do
157                  l = l - 83
158                  do resolution= 1, number-1
159
160                     i_e=int(512/(2**resolution))
161                     j_e=int(512/(2**resolution))
162
163                     allocate(Tout(i_e,j_e))
164
165                     call average_counting(i_e, j_e, Tin, Tout)
166
167                     deallocate(Tin)
168                     allocate(Tin(i_e, j_e))
169
170                     Tin = Tout
171
172                     do i = 1, i_e
173                        do j = 1, j_e
174                           Na(n,resolution+1) = Na(n,resolution+1)
                                 + Tin(i,j)
175                        end do
176                     end do
177
178                     deallocate(Tout)
179
180                  end do
181               end do
182
183               deallocate(Tin)
184               deallocate(T)
185
186               open(30, file=Fractal_D)
187               do n=1,4
188                  write(30,*) n, (Na(n,i), i=1,number)
189               end do
190               close(30)
191
192 ! write(*,*) '20' // yy // '-' // mm // '-' // dd // '-' // hh
      , ' box done!'
193            end do
194         end do
195      end do
196   end do
```

```
197
198 end program fractal
```

プログラム 4: temp_conv.f90

```
 1 module temp_conv
 2   implicit none
 3 contains
 4
 5 subroutine temp_conversion(conv, val, lower_end, Pc, Tc, T, y, m,
       d, h)
 6     integer, intent(in) :: val(1800,1800)
 7     real, intent(in) :: conv(0:255)
 8     real, intent(out) :: T(1800,1800)
 9     real, intent(in) :: Pc, Tc
10     integer, intent(in) :: lower_end
11     integer, intent(in) :: y, m, d, h
12
13     real, dimension(37) :: level = (/ 1, 2, 3, 5, 7, 10, 20,
          30, 50, 70, &
14          100,125,150,175,200,225,250,300,350,400,450,500,550,600,650,700,
              &
15          750,775,800,825,850,875,900,925,950,975,1000 /)
16
17     integer, parameter :: nx1 = 72, ny1 = 72
18     real, dimension(nx1+1, ny1+1, 37) :: temp1_sub
19     real, dimension(nx1+1, 1800, 37) :: temp2_sub
20     real, dimension(1800, 1800, 37) :: temp
21
22     character(2) :: yy, mm, dd, hh
23     character(100) :: data
24     integer :: i, j, l, n, a, b
25     real :: pressure
26
27
28     if (lower_end == 1) then
29
30        write(yy, '(I2.2)') y
31        write(mm, '(I2.2)') m
32        write(dd, '(I2.2)') d
33        write(hh, '(I2.2)') h
34        data = '../data/Temp/20'//yy//'/'//mm//'/temp20'//yy//mm
              //dd//hh//'.txt'
35
36        open(10,file=data)
37        do i = 1, nx1+1
38           do j = 1, ny1+1
```

```
39              do l = 1, 37
40                 read(10,*) n, a, b, temp1_sub(i,j,l)
41              end do
42              read(10,*)
43           end do
44        end do
45
46        do i = 1, nx1+1
47           do j = 1, ny1
48              do n = 1, 25
49                 do l = 1, 37
50                    temp2_sub(i, 25*(j-1)+n, l) = temp1_sub(i, j,
                         l) + &
51                       real(n-1)/25.0 * (temp1_sub(i, j+1, l) -
                            temp1_sub(i, j, l))
52                 end do
53              end do
54           end do
55        end do
56
57        do j = 1, 1800
58           do i = 1, nx1
59              do n = 1, 25
60                 do l = 1, 37
61                    temp(25*(i-1)+n, j, l) = temp2_sub(i, j, l) +
                         &
62                       real(n-1)/25.0 * (temp2_sub(i+1, j, l) -
                            temp2_sub(i, j, l))
63                 end do
64              end do
65           end do
66        end do
67
68        do i = 1, 1800
69           do j = 1, 1800
70              if(conv(val(i,j)) == 255)then
71                 T(i,j)= -999
72                 cycle
73              end if
74              pressure = 1
75              do l = 0, 36
76                 if (conv(val(i,j)) > temp(i,j,37-l)) then
77                    pressure = level(37-l)
78                    exit
79                 end if
80              end do
```

```
81              if (pressure < Pc) then
82                 T(i,j) = 100.0
83              else
84                 T(i,j) = -100.0
85              end if
86           end do
87        end do
88
89         else
90
91        do i = 1, 1800
92           do j = 1, 1800
93              if(conv(val(i,j)) == 255)then
94                 T(i,j)= -999
95                 cycle
96              end if
97
98              if (conv(val(i,j)) <= Tc) then
99                 T(i,j) = 100
100             else
101                T(i,j) = -100
102             end if
103
104          end do
105       end do
106    end if
107
108  end subroutine temp_conversion
109 end module temp_conv
```

プログラム 5: average.f90

```
1  module bound
2    implicit none
3
4  contains
5    subroutine bound_counting(T, Tout)
6      real, intent(in) :: T(:,:)
7      real, intent(out) :: Tout(:,:)
8      integer :: i, j, l, m
9      real :: count
10
11     do i = 1, 1799
12        do j = 1, 1799
13           count = 0.0
14           do l = 1, 2
15              do m = 1, 2
```

```
16                if(T(i-1+l,j-1+m) == -999)then
17                   count = -999
18                   exit
19                end if
20                count = count + T(i-1+l,j-1+m)
21             end do
22             if(count == -999)exit
23          end do
24          if(count == -999)then
25             Tout(i,j) = -999
26          end if
27          if (count < 400 .and. count > -400)then
28             Tout(i,j) = 1.0
29          else
30             Tout(i,j) = 0.0
31          end if
32       end do
33    end do
34
35  end subroutine bound_counting
36 end module bound
```

プログラム 6: bound.f90

```
1  module average
2    implicit none
3  contains
4
5    subroutine average_counting(i_e, j_e, Tin, Tout)
6      integer, intent(in) :: i_e, j_e
7      real, intent(in) :: Tin(:,:)
8      real, intent(out) :: Tout(:,:)
9
10     integer :: i, j, l, m, x, y
11
12
13     Tout = 0.0
14     do i = 1, i_e
15        do j = 1, j_e
16           do l = 1, 2
17              do m = 1, 2
18                 x = (i-1)*2 + l
19                 y = (j-1)*2 + m
20                    Tout(i,j) = Tout(i,j) + Tin(x,y)
21              end do
22           end do
23           if (Tout(i,j) > 0.0) Tout(i,j) = 1.0
```

```
24         end do
25       end do
26
27     end subroutine average_counting
28 end module average
```

境界長さからフラクタル次元を求める

雲の境界長さからフラクタル次元を求めるときに用いたソースコードを載せる

プログラム 7: least_squares_method.f90

```
1  program least_sqare
2    implicit none
3
4    integer :: sYEAR, eYEAR
5    integer :: sMONTH, eMONTH
6    integer :: sDAY, eDAY
7    integer :: sHOUR, eHOUR
8    integer :: long
9    integer :: start, end
10   real,dimension(15) :: Pc
11   integer :: i, j, d, l, m, n
12   integer :: year, month, day, hour
13   real :: Dimens, sigmaa, r, dimens_all, dimens_sigma, choice
14   real, dimension(9) :: dimens_sub
15   real, dimension(4,10) :: Na
16   real :: a, b, xb, yb, x2b, xyb, sigmay, sse, sst, under
17   real :: paramet
18   character(len=3) :: sp
19   character(len=2) :: yy, mm, dd, hh
20   character(len=200) :: Fractal, Fractal_D
21   logical :: exist_D
22
23   sYEAR = 20
24   eYEAR = 20
25   sMONTH = 06
26   eMONTH = 06
27   sDAY = 09
28   eDAY = 09
29   sHOUR = 23
30   eHOUR = 23
31
32   paramet = 1.5
33
```

```
34   data Pc/100., 150., 200., 250., 300., 350., 400., 450.,
        500., 550., 600., 650., 700., 750., 800./
35
36   do d = 1,1
37
38      write(sp,'(I3.3)') int(Pc(d))
39
40      do year = sYEAR, eYEAR
41         write(yy, '(I2.2)') year
42         do month = sMONTH, eMONTH
43            write(mm, '(I2.2)') month
44            do day = sDAY, eDAY
45               if ((month==2 .and. day==29 .and. mod(year,4)/=0) .
                     or. &
46                   (month==2 .and. day>=30) .or. &
47                   ((month==4 .or. month==6 .or. month==9 .or.
                        month==11) .and. day==31)) cycle
48               write(dd, '(I2.2)') day
49               do hour = sHOUR, eHOUR
50                  write(hh, '(I2.2)') hour
51
52                  Fractal = '../IR1/20'//yy//'/'//mm//'/Pc'//sp
                       //'/B'//yy//mm//dd//hh//'_'//sp//'.txt'
53                  Fractal_D = '../IR1/20'//yy//'/'//mm//'/Pc'//sp
                       //'/B_D'//yy//mm//dd//hh//'_'//sp//'.txt'
54
55                  inquire(file=Fractal, exist=exist_D)
56
57                  if (.not. exist_D) then
58                     open(20, file=Fractal_D)
59                     do n = 1, 4
60                        write(20,*) -999.0, -999.0, -999.0,
                             -999.0, n, -999.0, -999.0
61                     end do
62                     close(20)
63                     cycle
64                  end if
65
66                  open(10, file=Fractal)
67                  do i = 1,4
68                     read(10, *) a, (Na(i, n), n = 1, 10)
69                  end do
70                  close(10)
71
72                  open(20, file=Fractal_D)
73                  do i = 1, 4
```

```
74
75                    Dimens = -999.0
76                    sigmaa = -999.0
77                    r = -999.0
78                    long = 0
79                    dimens_sub = -999.0
80                    choice = 999.0
81
82                    if(Na(i,1) == 0)then
83                       write(20,*) dimens, sigmaa, r, choice, long
84                       cycle
85                    end if
86
87                    do n = 1, 9
88                       dimens_sub(n) = abs(log10(Na(i,n)) - log10(
                            Na(i,n+1)))/log10(2.0)
89                    end do
90
91                    do n = 2, 7
92                       if(long /= 0 .and. (dimens_sub(n) <= 1.0 .
                            or. dimens_sub(n) >= 1.9))then
93                          exit
94                       end if
95                       if(choice > abs(dimens_sub(n) - dimens_sub(
                            n+1)) .and. dimens_sub(n) > 1.0 .and. &
96                             dimens_sub(n) < 1.9 .and. dimens_sub(n
                                +1) > 1.0 .and. dimens_sub(n+1) <
                                1.9)then
97                          choice = abs(dimens_sub(n) - dimens_sub(
                               n+1))
98                          write(*,*) dimens_sub(n), dimens_sub(n
                               +1), n
99                          long = n
100                      end if
101                   end do
102
103
104                   if(long == 0)then
105                      write(20,*) dimens, sigmaa, r, choice, long
106                      cycle
107                   end if
108
109                   xb = 0.0; yb = 0.0; x2b = 0.0; xyb = 0.0
110                   sigmay = 0.0
111                   sse = 0
112                   sst = 0
```

```
113                     under = 0.0
114
115                     do n = long, long + 2
116                        xb = xb + log10(2.0**(n - 1))
117                        yb = yb + log10(Na(i,n))
118                        x2b = x2b + log10(2.0**(n - 1))**2
119                        xyb = xyb + log10(Na(i,n)) * log10(2.0**(n
                               - 1))
120                     end do
121
122                     xb = xb /real(3)
123                     yb = yb / real(3)
124                     x2b = x2b / real(3)
125                     xyb = xyb / real(3)
126
127                     a = (xyb - xb * yb) / (x2b - xb**2)
128                     b = yb - a * xb
129
130                     do n = long, long + 2
131                        sigmay = sigmay + (a*log10(2.0**(n-1)) + b
                               - log10(Na(i,n)))**2
132                        sse = sse + (a*log10(2.0**(n-1)) + b -
                              log10(Na(i,n)))**2
133                        sst = sst + (yb - log10(Na(i,n)))**2
134                     end do
135
136                     sigmay = sqrt(sigmay / real(3))
137
138                     r = 1 - sse / sst
139                     sigmaa = sigmay * sqrt(1.0 / (real(3) * (x2b
                           - xb**2)))
140                     Dimens = -a
141
142                     write(20,*) dimens, sigmaa, r, choice, long
143                  end do
144
145                  close(20)
146               end do
147            end do
148         end do
149      end do
150   end do
151
152 end program least_sqare
```