2001年度
青山学院大学理工学部物理学科
卒業論文

研究題目

価格変動の新しいモデルとそのフラクタル性

正田　真理
羽田野研究室

青山学院大学 理工学部
物理学科 卒業論文

価格変動の新しいモデルと
そのフラクタル性

正田真理 著

価格変動の新しいモデルと
そのフラクタル性

正田真理

羽田野研究室

平成 14 年 3 月 10 日

概 要

高安らによる価格変動のモデルに「量」の概念を加えたところ、価格変動の分布に大きな違いはみられなかったが、各ディーラーの取引間隔の分布がフラクタル性を示す事が分かった。本研究では、このフラクタル性の原因を理論的に解明する事を目的としている。

目 次

1　はじめに

近年、統計物理学や複雑系の研究の観点から市場価格の分布が興味を持たれているが、金融工学の分野では価格変動の分布はガウス分布で近似されてきた。しかし、実際にはガウス分布よりも価格の高騰や暴落の多い分布となっている事が分かっている。そこで、フラクタル分布を価格分布とする価格変動のモデルが、高安らによって提案された [1]。

しかし、このモデルでは各ディーラーは「売ったら次は必ず買う、買ったら次は必ず売る」というように、売りと買いを交互に行なうと仮定されていた。この仮定は明らかに現実とは異なるものである。私はこの点に疑問を抱き、新たに「量」という概念を導入する事によって、売り足し、買い足しを再現する事にした。

2 章では本研究で新たに導入する価格変動の新しいモデルについて説明する。3 章では取引間隔がフラクタルになるということについて、高安らのモデル [1] と比較して説明する。４章ではモデルを単純化することによって得られた、フラクタル性の原因を説明する。

2　新しいモデルでの価格変動の再現

まず、価格変動について検証する。2.1 節ではモデルの概要を説明し、2.2 節では価格変動がすその広い分布となっていることを説明する。

2.1　モデルの概要

いま、N 人のディーラー $(1 \leq i \leq N)$ がそれぞれ手持ち量 $H_i (0 \leq H_i \leq H_{\max})$ と、希望買値 B_i、希望売値 S_i を持っているとする。$H_{\max}$ は最大手持ち量で、各ディーラーに共通とする。売値と買値の差は一定とし、その値を

$$S_i - B_i = \lambda > 0 \tag{1}$$

とする。

各ディーラーは方程式

$$B_i(t+1) = B_i(t) + a_i(t) + c\Delta P_{\mathrm{prev}} \tag{2}$$

に従って、買値（及び売値）を時間発展させていく。ここで、$a_i(t)$ の定義が、高安らのモデル [1] と私の提案するモデルの決定的な違いである。

高安らのモデルでは初期値を与えたのち、取引が行なわれる毎に符号が入れ代わるものであった。それに対し、ここでは $a_i(t)$ は手持ち量 H_i に比例する項で、

$$a_i(t) = c' \left(1 - 2 \times \frac{H_i(t)}{H_{\max}} \right) \tag{3}$$

と定義する。なお、$H_i(t)$ は時刻 t における手持ち量である。手持ち量の範囲は $0 \leq H_i(t) \leq H_{\max}$ なので、手持ち量が少ないほど買値を上げ、多いほど売値を下げる効果を $a_i(t)$ は表わしている。c' はその強さをきめるパラメータである。

また、式 (2) において ΔP_{prev} は 1 回前の取引における価格の増減である。この値は、次の取引が起こるまで同じ値を取り続ける。いわゆる順張りの項である。c はその強さを決めるパラメータである。式 (2) の第 2 項と第 3 項の和が正ならそのディーラーは買手、負なら売手となる。

上のような時間発展の中で取引が成立するのは、全てのディーラーの中で最大の希望売値を持つディーラーと最小の希望売値を持つディーラが

$$\max_i B_i \leq \min_j S_j \tag{4}$$

を満たした時とする。ここで、$\max_i$ 〜、$\min_i$ 〜はそれぞれ i を変化させた時の「〜」の最大値、最小値である。式 (2) の関係があるので、取引条件 (4) は

$$L(t) \equiv \max_i B_i(t) - \min_j B_j(t) \geq \lambda \tag{5}$$

と書き直せる。

市場価格 $P(t)$ は、取引が成立した時の売値と買値の中間の値とする。そして、次の取引が成立するまでは、これを市場価格とする。すなわち、$P(t)$ は

$$P(t) = \begin{cases} \dfrac{\max_i B_i(t) + \min_i S_i(t)}{2} = \dfrac{\max_i B_i(t) + \min_i B_i(t) + \lambda}{2} & (L(t) \geq \lambda \text{ の場合}) \\ P(t-1) & (L(t) < \lambda \text{ の場合}) \end{cases}$$

のようになる。

また、取引成立時には手持ち量の移動を行う。移動量 ΔH は取引成立毎にランダムに決定する。但し、手持ち量 H_i を $0 \leq H_i \leq H_{\max}$ の範囲に抑えるために次のように決める。まず、買手に対する値 $H_{\max} - H_i(t)$ と

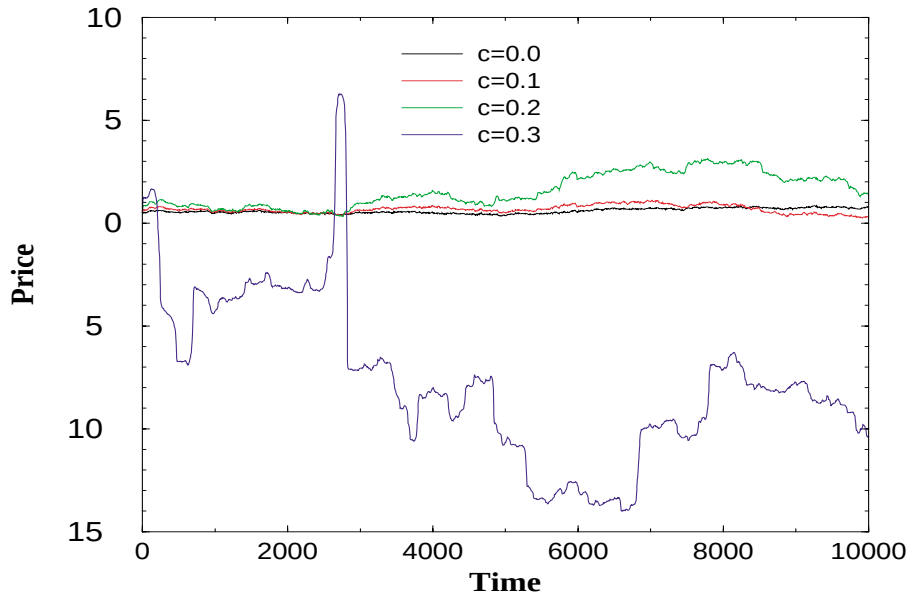

図 1: 高安らのモデル [1] による $N = 100$、$\lambda = 1$ での市場価格の時間変化。パラメーター c は 0 から 0.3 まで変化させた。

売手に対する値 $H_j(t)$ のうち少ない方を、取引の最大値 $T_{\max}$ とする。次に、$[0, T_{\max}]$ の範囲からランダムに移動量 ΔH を選択する。

実際のシミュレーションにあたっては、各ディーラーの初期値 $B_i(0)$ は $[-\lambda/2, \lambda/2]$ の一様乱数で与え、手持ち量は半数を $H_i(0) \equiv 0$、半数を $H_i(0) \equiv H_{\max}$ とした。変化させるパラメータはディーラー数 N、最大手持ち量 $H_{\max}$、閾値 λ、2 つの定数 c, c' である。

2.2 高安らのモデルとの比較

高安らのモデル [1] と私のモデルの振る舞いを比較するため、両モデルに対して、$N = 100$、$H_{\max} = 1000000$、$\lambda = 1$、$c' = 0.01$ で c を変化させてシミュレーションを行なった。結果を図 1 と図 2 に示す。

これらを比べると、高安らのモデル [1] よりも新しいモデルの方が価格の揺らぎが若干大きいが、基本的には同じような振舞をしている。また、これらのグラフから既存のモデルでの範囲 $c = 0.0 \sim 0.3$ での振舞が新しいモデルではちょうど範囲 $c = 0.0 \sim 0.6$ での振舞に似ていることがわかる。

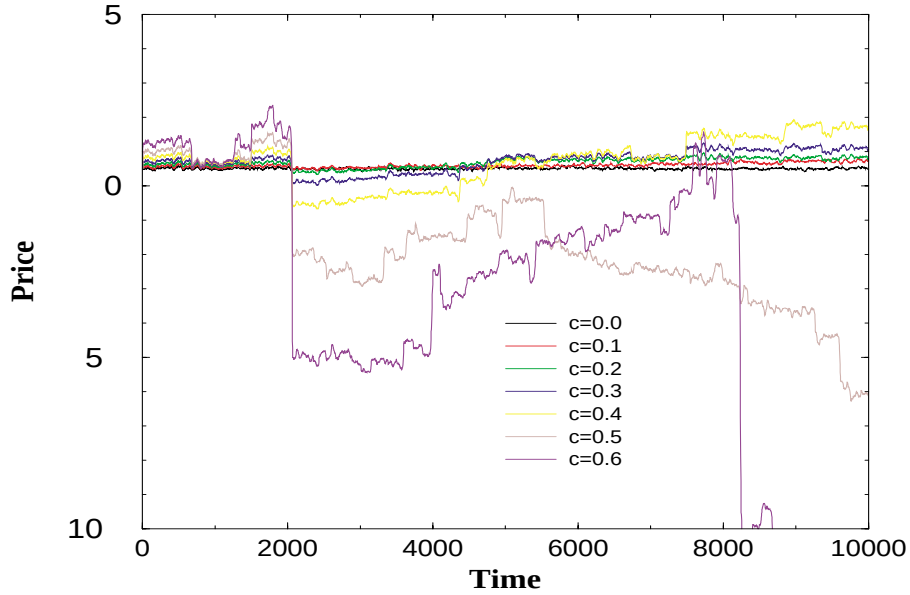

図 2: 新しいモデルによる $N = 100$、$\lambda = 1$ での市場価格の時間変化。パラメーター c は 0 から 0.6 まで変化させた。

次に、同じ条件で高安らのモデル [1] と新しいモデルの価格変動 $\Delta P = P(t) - P(t-1)$ の確率密度分布を比較する。結果を図 3 と図 4 に示す。これらより、c が小さい部分では 0 付近での分布の形状に違いが見られるが、c が大きくなるにつれて共に冪乗のすそを持つフラクタル分布になっていく様子がわかる。

以上をまとめると、新旧 2 つのモデルは価格変動の特徴に大きな違いはないものと考えられる。

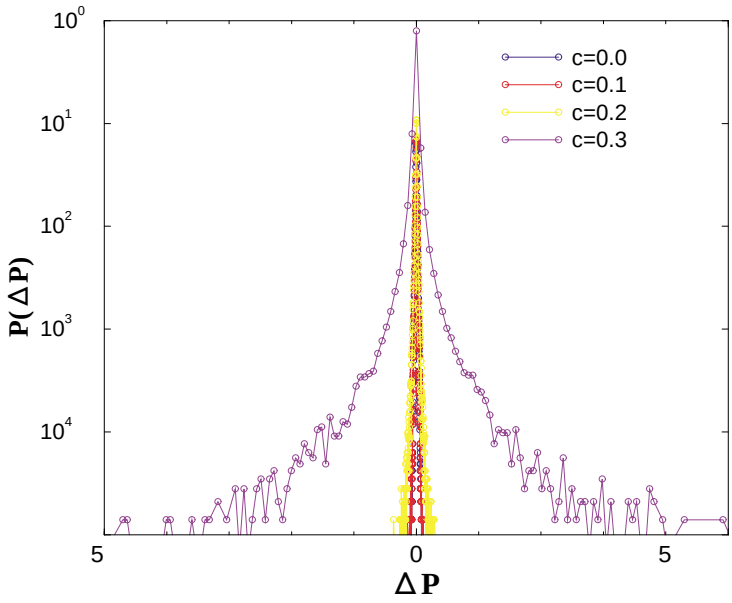

図 3: 高安らのモデル [1] による $N = 100$、$\lambda = 1$ での価格変動 ΔP の確率密度分布。パラメーター c は 0 から 0.3 まで変化させた。

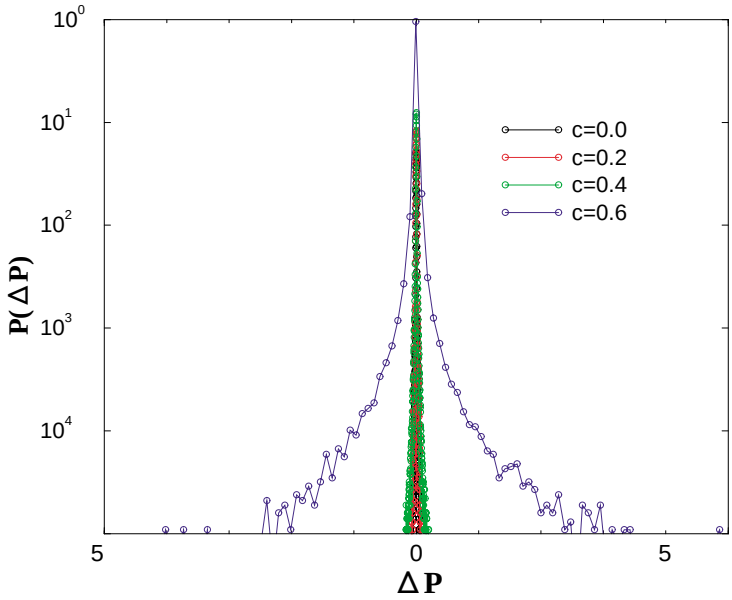

図 4: 新しいモデルによる $N = 100$、$\lambda = 1$ での価格変動 ΔP の確率密度分布。パラメーター c は 0 から 0.6 まで変化させた。

3　各ディーラーの取引間隔

この節では、2つのモデルにおける各ディーラーの取引間隔の分布を調べる。新しいモデルにだけ、分布にフラクタル性があらわれることが分かる。

取引間隔には、

1. 誰かが取引を行ってから次に別の誰かが取引を行うまでの時間間隔
2. 一人のディーラーに注目して、その人が取引を行う時間間隔

の２種類がある。ここでは後者の取引間隔に着目し、シミュレーションにおけるこの分布を調べた。具体的な方法としては、全てのディーラーについて個々の取引間隔を調べ、横軸に取引間隔、縦軸にその出現回数をプロットする、というものである。その結果を図5と図6に示す。

これらを比較すると、新しいモデルでは非常に直線性がよいことがわかる。その直線の傾きは、約 -3 である。つまり新しいモデルでは、

$$P(\tau) \propto \tau^{-3} \tag{6}$$

というフラクタル分布が現れている。

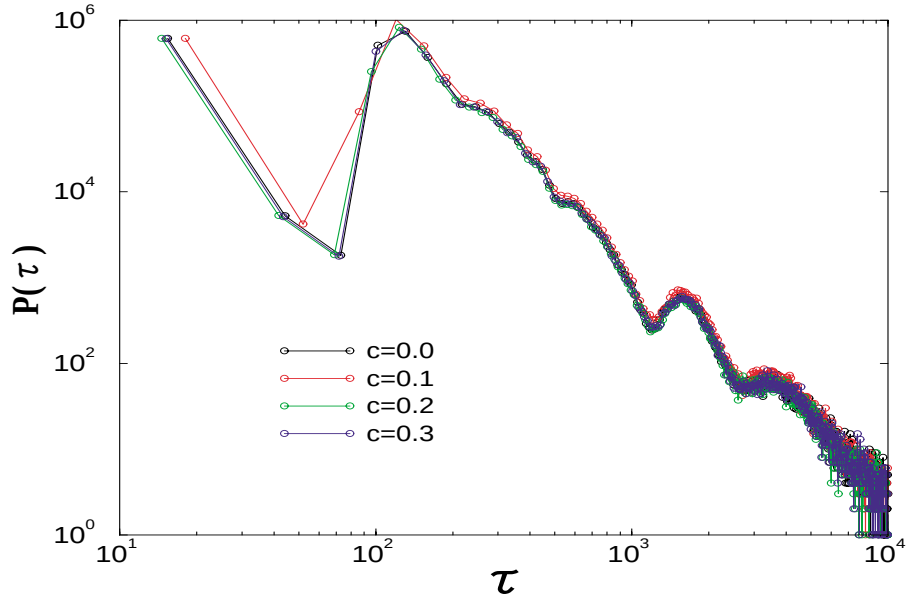

図 5: 高安らのモデル [1] による $N = 100$、$\lambda = 1$ での各ディーラーの取引間隔の分布。パラメーター c は 0 から 0.3 まで変化させた。

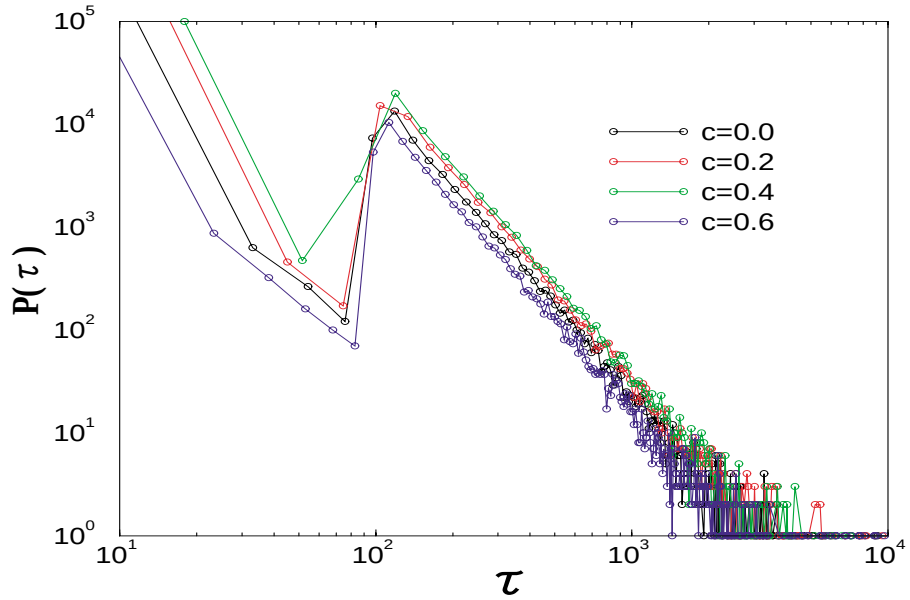

図 6: 新しいモデルによる $N = 100$、$\lambda = 1$ での各ディーラーの取引間隔の分布。パラメーター c は 0 から 0.6 まで変化させた。

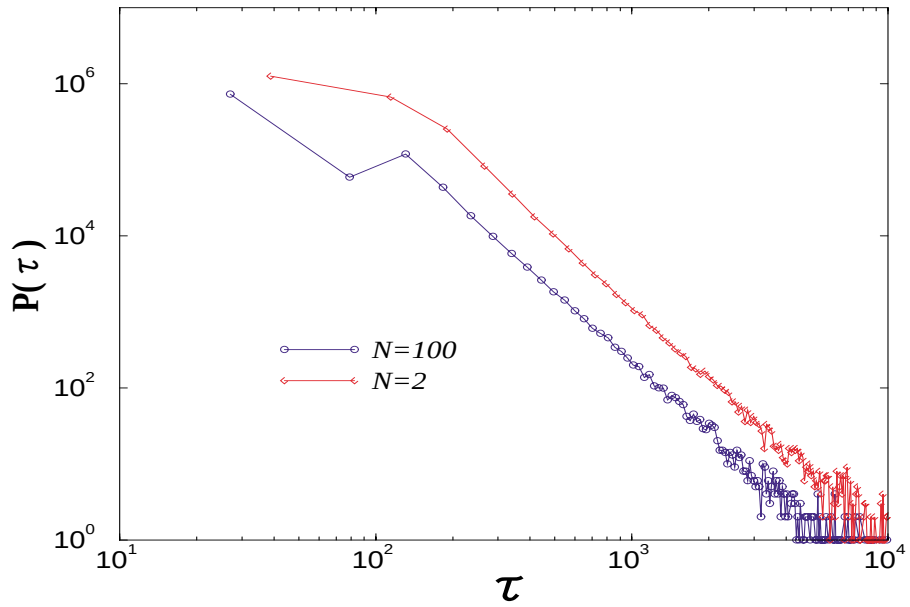

図 7: $\lambda = 1$、$c = 0.0$ で、ディーラー数 N が 2 と 100 での取引間隔の分布。

4 取引間隔の理論的解明

なぜ、新しいモデルのように「量」の概念を加えると、取引間隔がフラクタルになるのであろうか。

幸いなことに、この取引間隔のフラクタル性は単純化しても保たれる。まず、図 6 においては、ΔP_{prev} の係数 c が変化しているにも関わらず、そのフラクタル性は保たれていることがわかる。次に図 7 からわかるように、ディーラー数 N は 2 でも良い。よってこれ以降はディーラー数 $N = 2$、$c = 0$ で考える。

$c = 0$ のとき、買い値の時間発展は

$$B_i(t+1) = B_i(t) + a_i(t) \tag{7}$$

$$a_i(t) = c' \left(1 - 2 \times \frac{H_i(t)}{H_{\max}} \right) \tag{8}$$

となっている。ここで、$a_i(t)$ をそのまま量として捉えると、取引手順は次のようになる。買手を i、売手を j とすると、まず $[0, a_i(t) + c']$ の範囲でランダムな値 x を選ぶ。そして、

$$a_i(t+1) = a_i(t) - x \qquad a_j(t+1) = a_j(t) + x \tag{9}$$

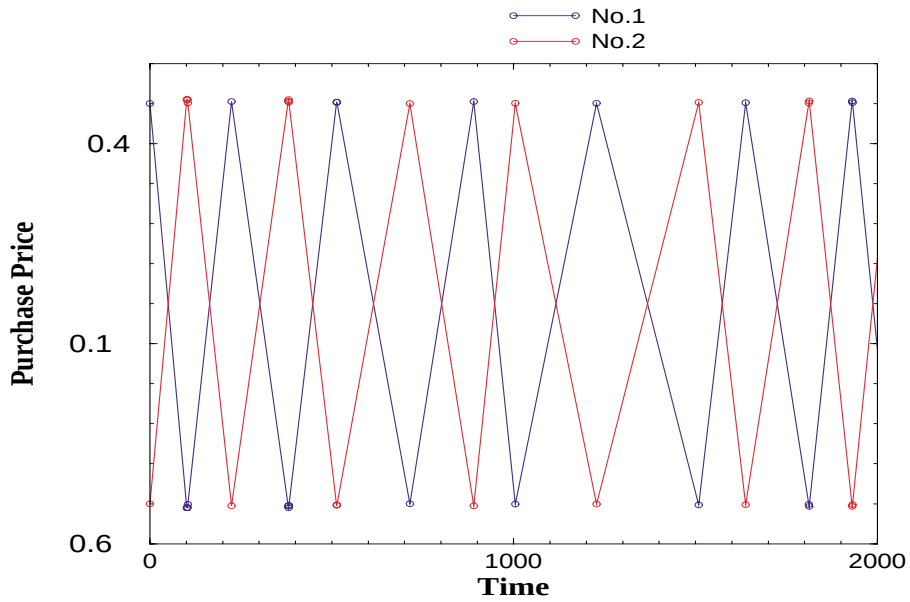

図 8: $\lambda = 1$、$c = 0$、$N = 2$ に対する希望買値の時間変化。

とする。

つぎに $a_i(t)$ に注目してみる。図 8 において、直線の傾きが $a_i(t)$、上（下）で跳ね返ってから下（上）で跳ね返るまでの時間が取引間隔 τ である。つまり、$a_i(t)$ と τ の間には

$$\tau_i(t) = \frac{\lambda}{a_i(t)} \tag{10}$$

という関係式が成り立っている。また、取引間隔の分布 $P(\tau)$ と取引成立時における $a_i(t)$ の分布 $P(a)$ の間には、

$$P(\tau)dx = P(a)da \tag{11}$$

が成立する。よって、式 (6)、(10)、(11) から

$$P(a) \propto |a| \tag{12}$$

が導き出せる。

図 9 がディーラー数 $N = 2$、売値と買値の差 $\lambda = 1$、$c' = 0.01$ として、シミュレーションして得た a の分布である。但し、連続した取引が行なわれた場合、最後の a の値だけを採用した。確かに 1 次の分布 (12) となっていることがわかる。

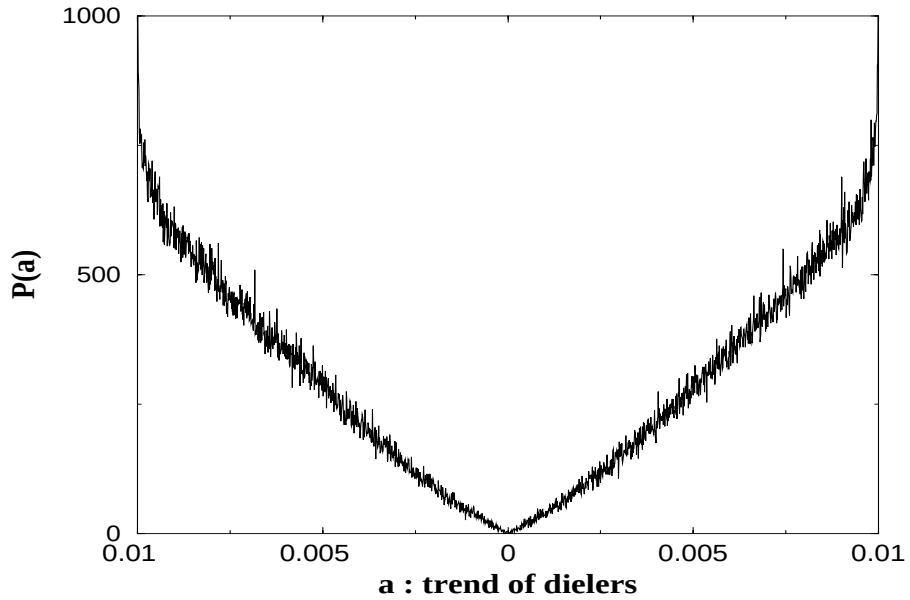

図 9: 単純化シミュレーションから得た a の分布。

5 まとめ

本研究では、価格変動のモデルに「量」の概念を加えてみたところ、価格そのものの振舞には大きな変化は見られなかったのだが、個々のディーラーの取引間隔がフラクタルになった。

今回の新しいモデルでは、各ディーラーは連続して取引を行うことができるが、図 9 においては、上で述べたように連続した取引のうちで最後の a の値だけを取り出している。これが何を意味するのかというと、買い足しや売り足しといった取引を許したことによって取引間隔の分布がフラクタル性を持ったということである。

現実世界ではもちろん買い足しや売り足しが行われている。このことからすぐに取引間隔がフラクタルであると言えるわけではないが、現実世界での各ディーラーの取引間隔がフラクタルである可能性は十分にあると期待される。

6　最後に

本研究を行うにあたって、１年間実に丁寧な指導をして下さった羽田野直道先生に心から感謝いたします。また、忙しい中、研究を手助けして下さった同研究室の大学院生の皆様にお礼を申し上げます。

また、この研究をするにあたって、経済物理という新しい分野に触れることができて良かった。この分野がもっと広がることを期待している。

参考文献

[1] 饗場行洋、「価格変動のフラクタル分布」（青山学院大学卒業論文、2000）

[2] 高安秀樹、「フラクタル」（朝倉書店、1987）

[3] H. Takayasu, M. Takayasu, M. P. Okazaki, K. Marumo and T. Shimizu, in Paradigms of Complexity, ed. M. M. Novak (World Scientific, Singapore 2000) pp.243-258

A プログラムリスト

A.1 既存のモデル

A.1.1 価格 $P(t)$、ΔP 作成ルーチン

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 46412679
#define N_trader 1000
struct dealer{
   float purchase;
   float trend;
};
float abs_r(float x);
main()
{
  struct dealer D[N_trader];
  int t,i,imax,imin,counter=0,n,t_max;
  float lambda=1.0,c,alfa=0.01,Pprev=0.0,P=0.0,delta_P=0.0;
  float L,Bmax,Bmin,amax,amin,ran;
  float pre_gap=0.0,gap_P=0.0,xp=0.0,pp=0.0;
  FILE *datafile1,*datafile2;
  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);
  datafile1=fopen("defpri.dat","w");
  datafile2=fopen("defdif.dat","w");
  printf("---Input Parameters---\n");
  printf("c=(0.0-0.3):");scanf("%f",&c);
  printf("Number of 'D':");scanf("%d",&n);
  printf("t_max:");scanf("%d",&t_max);
  for(i=0;i<N_trader;i++){
    D[i].purchase=0.0;
    D[i].trend=0.0;
  }
  for(i=0;i<n;i++){
    ran=(2.0*sprng())-1.0;
    D[i].purchase=ran*lambda/2.0;
    D[i].trend=ran*alfa;
  }
  for(t=0;t<=t_max;t++){
    imax=0;
    imin=0;
    for(i=1;i<n;i++){
      if(D[i].purchase<D[imin].purchase){imin=i;}
      if(D[i].purchase>D[imax].purchase){imax=i;}
```

```
    }
    L=D[imax].purchase-D[imin].purchase;
    if(L>=lambda){
      P=(D[imax].purchase+D[imin].purchase+lambda)/2.;
      xp=P;pp=Pprev;
      delta_P=P-Pprev;
      D[imin].trend=abs_r(D[imin].trend);
      D[imax].trend=-abs_r(D[imax].trend);
    }else{
      P=Pprev;
      xp=P;pp=Pprev;
    }
    for(i=0;i<n;i++){
      D[i].purchase+=D[i].trend+c*delta_P;
    }
    gap_P=xp-pp;
    if(gap_P!=0.0){
      if(gap_P==pre_gap){
printf("t=%d\n",t);
counter++;
      }
      fprintf(datafile1,"%10d %26.16e\n",t,P);
      fprintf(datafile2,"%26.16e\n",gap_P);
    }
    pre_gap=gap_P;
    Pprev=P;
  }
  fclose(datafile1);
  fclose(datafile2);
  printf("counter %d\n",counter);
}
float abs_r(float x)
{
  if(x<0.){
    x=-x;
  }
  return x;
}
```

A.1.2 取引間隔 τ 作成ルーチン

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 46412679
```

```
#define N_trader 1000
#define T_max 1000000
struct dealer{
  float purchase;
  float trend;
};
float abs_r(float x);
main()
{
  struct dealer D[N_trader];
  int B=0,a=1,t,i,imax,imin,counter=0,n,t_max;
  int k,kai,point,point1,point2,all;
  float lambda=1.0,c,alfa=0.01,Pprev=0.0,P=0.0,delta_P=0.0;
  float L,Bmax,Bmin,amax,amin,ran;
  float pre_gap=0.0,gap_P=0.0,xp=0.0,pp=0.0;
  FILE *datafile1;
  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);
  datafile1=fopen("deftra.dat","w");
  printf("---Input Parameters---\n");
  printf("c=(0.0-0.3):");scanf("%f",&c);
  printf("Number of 'D':");scanf("%d",&n);
  printf("t_max:");scanf("%d",&t_max);
  for(k=0;k<n;k++){
    for(i=0;i<N_trader;i++){
      D[i].purchase=0.0;
      D[i].trend=0.0;
    }
    for(i=0;i<n;i++){
      ran=(2.0*sprng())-1.0;
      D[i].purchase=ran*lambda/2.0;
      D[i].trend=ran*alfa;
    }
    kai=0; point1=0; point2=0; point=0;
    for(t=0;t<=t_max;t++){
      imax=0;
      imin=0;
      for(i=1;i<n;i++){
if(D[i].purchase<D[imin].purchase){imin=i;}
if(D[i].purchase>D[imax].purchase){imax=i;}
      }
      L=D[imax].purchase-D[imin].purchase;
      if(L>=lambda){
if(imax==k||imin==k){
  point2=t;
  point=point2-point1;
  point1=point2;
  kai++;
  fprintf(datafile1,"%10d\n",point);
}
```

```
P=(D[imax].purchase+D[imin].purchase+lambda)/2.;
xp=P;pp=Pprev;
delta_P=P-Pprev;
D[imin].trend=abs_r(D[imin].trend);
D[imax].trend=-abs_r(D[imax].trend);
      }else{
P=Pprev;
xp=P;pp=Pprev;
      }
      for(i=0;i<n;i++){
D[i].purchase+=D[i].trend+c*delta_P;
      }
      gap_P=xp-pp;
      if(gap_P!=0.0){
if(gap_P==pre_gap){
  counter++;
}
      }
      pre_gap=gap_P;
      Pprev=P;
    }
    all+=kai;
  }
  fclose(datafile1);
  printf("all= %d\n",all);
  printf("counter %d\n",counter);
}
float abs_r(float x)
{
  if(x<0.){
    x=-x;
  }
  return x;
}
```

A.2 新しいモデル

A.2.1 価格 $P(t)$、ΔP 作成ルーチン

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 46412679
#define N_trader 1000
#define H_MAX 1000000
```

```
struct client{
  float purchase;
  int holding;
  float trend;
};
main()
{
  struct client D[N_trader];
  int t,i,j,imax,n,imin,Hmin,counter=0,t_max;
  float lambda=1.0,c,Pprev=0.0,P=0.0,delta_P=0.0,L,Q,ran;
  float pre_gap,gap_P=0.0,xp=0.0,pp=0.0;
  FILE *datafile1,*datafile2;
  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);
  datafile1=fopen("newpri.dat","w");
  datafile2=fopen("newdif.dat","w");
  printf("---Input The Parameters---\n");
  printf("c=(0.0-1.0)");scanf("%f",&c);
  printf("Number of 'D':");scanf("%d",&n);
  printf("t_max:");scanf("%d",&t_max);
  for(i=0;i<N_trader;i++){
    D[i].purchase=0.0;
    D[i].holding=0;
  }
  Hmin=0;
  for(i=0;i<(n/2);i++){
    D[i].holding=H_MAX;
    D[i].trend=(1.-D[i].holding*2./H_MAX)/100.;
  }
  for(i=(n/2);i<n;i++){
    D[i].holding=0;
    D[i].trend=(1.-D[i].holding*2./H_MAX)/100.;
  }
  for(i=0;i<n;i++){
    ran=(2.0*sprng())-1.0;
    D[i].purchase=ran*lambda/2.0;
  }
  for(t=0;t<=t_max;t++){
    pre_gap=gap_P;
    imax=0;
    imin=0;
    for(i=1;i<n;i++){
      if(D[i].purchase<=D[imin].purchase){imin=i;}
      if(D[i].purchase>=D[imax].purchase){imax=i;}
    }
    L=D[imax].purchase-D[imin].purchase;
    if(L>=lambda){
      P=(D[imax].purchase+D[imin].purchase+lambda)/2.;
      xp=P;pp=Pprev;
      delta_P=P-Pprev;
```

```
        Pprev=P;
        if(H_MAX-D[imax].holding<=D[imin].holding){
         Hmin=H_MAX-D[imax].holding;
         }
        else if(H_MAX-D[imax].holding>D[imin].holding){
        Hmin=D[imin].holding;
        }
        else{printf("It's an error.\n");exit(1);}
        Q=(int)(sprng()*Hmin);
D[imax].holding+=Q;
D[imin].holding-=Q;
        D[imax].trend=(1.0-D[imax].holding*2./H_MAX)/100.;
        D[imin].trend=(1.0-D[imin].holding*2./H_MAX)/100.;
      }else{
        P=Pprev;
        xp=P;pp=Pprev;
      }
      for(i=0;i<n;i++){
        D[i].purchase+=D[i].trend+c*delta_P;
      }
      gap_P=xp-pp;
      if(gap_P!=0.0){
      if(gap_P==pre_gap){counter++;}
      else{
        fprintf(datafile1,"%10d %26.16e\n",t,P);
        fprintf(datafile2,"%26.16e\n",gap_P);
      }}
    }
    fclose(datafile1);
    fclose(datafile2);
    printf("counter %d\n",counter);
}
```

A.2.2 取引間隔 τ 作成ルーチン

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 46412679
#define N_trader 1000
#define H_MAX 1000000
struct client{
  float purchase;
  int holding;
  float trend;
```

```
};
main()
{
  struct client D[N_trader];
  int t,i,j,imax,n,imin,Hmin,counter=0,t_max;
  int k,kai,point,point1,point2,all,x;
  float lambda=1.0,c,Pprev=0.0,P=0.0,delta_P=0.0,L,Q,ran;
  float pre_gap,gap_P=0.0,xp=0.0,pp=0.0;
  FILE *datafile[4];
  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);
  datafile[0]=fopen("newc00.dat","w");
  datafile[1]=fopen("newc02.dat","w");
  datafile[2]=fopen("newc04.dat","w");
  datafile[3]=fopen("newc06.dat","w");
  printf("---Input The Parameters---\n");
  printf("Number of 'D':");scanf("%d",&n);
  printf("t_max:");scanf("%d",&t_max);
  for(x=0;x<4;x++){
    c=0.2*x;
    all=0;kai=0;
    for(k=0;k<n;k++){
      t=0;point=0;point1=0;point2=0;
      Pprev=0.0;P=0.0;gap_P=0.0;xp=0.0;pp=0.0;pre_gap=0.0;
      for(i=0;i<N_trader;i++){
D[i].purchase=0.0;
D[i].holding=0;
      }
      Hmin=0;
      for(i=0;i<(n/2);i++){
D[i].holding=H_MAX;
D[i].trend=(1.-D[i].holding*2./H_MAX)/100.;
      }
      for(i=(n/2);i<n;i++){
D[i].holding=0;
D[i].trend=(1.-D[i].holding*2./H_MAX)/100.;
      }
      for(i=0;i<n;i++){
ran=(2.0*sprng())-1.0;
D[i].purchase=ran*lambda/2.0;
      }
      for(t=0;t<=t_max;t++){
pre_gap=gap_P;
imax=0;
imin=0;
for(i=1;i<n;i++){
  if(D[i].purchase<=D[imin].purchase){imin=i;}
  if(D[i].purchase>=D[imax].purchase){imax=i;}
}
L=D[imax].purchase-D[imin].purchase;
```

```
if(L>=lambda){
  if(imax==k||imin==k){
    point2=t;
    point=point2-point1;
    point1=point2;
    kai++;
    fprintf(datafile[x],"%10d\n",point);
  }
  P=(D[imax].purchase+D[imin].purchase+lambda)/2.;
  xp=P;pp=Pprev;
  delta_P=P-Pprev;
  Pprev=P;
  if(H_MAX-D[imax].holding<=D[imin].holding){
    Hmin=H_MAX-D[imax].holding;
  }
  else if(H_MAX-D[imax].holding>D[imin].holding){
    Hmin=D[imin].holding;
  }
  else{printf("It's an error.\n");exit(1);}
  Q=(int)(sprng()*Hmin);
  D[imax].holding+=Q;
  D[imin].holding-=Q;
  D[imax].trend=(1.0-D[imax].holding*2./H_MAX)/100.;
  D[imin].trend=(1.0-D[imin].holding*2./H_MAX)/100.;
}else{
  P=Pprev;
  xp=P;pp=Pprev;
}
for(i=0;i<n;i++){
  D[i].purchase+=D[i].trend+c*delta_P;
}
gap_P=xp-pp;
if(gap_P!=0.0){
  if(gap_P==pre_gap){counter++;}
}
      }
      all+=kai;
    }
    printf("c=%f\n",c);
    printf("all= %d\n",all);
    printf("counter %d\n",counter);
  }
  for(i=0;i<4;i++){
  fclose(datafile[i]);
  }
}
```

A.3 単純化モデル

A.3.1 取引間隔 τ、$a_i(t)$ 作成ルーチン

```
#include <stdio.h>
#include <stdlib.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 46412679
#define _N 1000
struct client{
  float purchase;
  float trend;
};
float abs_(float x);
main()
{
  struct client D[_N];
  int t,i,imax,imin,kai;
  int point,point1,point2,k,all=0,n=0,t_max=0,Na=0;
  float gyaku,ran;
  float lambda=1.0,vol,L;
  FILE *datafile1,*datafile2,*datafile3,*datafile4,*datafile5;
  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);
  datafile1=fopen("simptra.dat","w");
  datafile2=fopen("simptrend.dat","w");
  datafile3=fopen("simpafttre.dat","w");
  datafile4=fopen("simpa_del.dat","w");
  datafile5=fopen("simpvol.dat","w");
  printf("Number of 'D':");scanf("%d",&n);
  printf("t_max:");scanf("%d",&t_max);
  for(k=0;k<n;k++){
    t=0;
    for(i=0;i<_N;i++){
      D[i].purchase=0.0;
      D[i].trend=0.0;
    }
    for(i=0;i<(n/2);i++){
      D[i].trend=0.01;
    }
    for(i=(n/2);i<n;i++){
      D[i].trend=-0.01;
    }
    for(i=0;i<n;i++){
      ran=(2.0*sprng())-1.0;
      D[i].purchase=ran*lambda/2.0;
    }
    kai=0; point1=0; point2=0; point=0;
    for(t=0;t<=t_max;t++){
```

```
      vol=0; imax=0; imin=0;
      for(i=1;i<n;i++){
if(D[i].purchase<=D[imin].purchase){imin=i;}
if(D[i].purchase>=D[imax].purchase){imax=i;}
      }
      L=D[imax].purchase-D[imin].purchase;
      if(L>=lambda){
if(imax==k||imin==k){
  point2=t;
  point=point2-point1;
  point1=point2;
  kai++;
  fprintf(datafile1,"%10d\n",point);
  }
vol=sprng()*(D[imax].trend+0.01);
fprintf(datafile4,"%f %f\n",D[imax].trend,vol);
if(k==0 && point!=1){
    fprintf(datafile2,"%f\n",D[0].trend);
    fprintf(datafile5,"%f\n",vol);
    Na+=1;
}
D[imax].trend-=vol;
D[imin].trend+=vol;
if(k==0 && point!=1){
  fprintf(datafile3,"%f\n",D[0].trend);
}
      }
      for(i=0;i<n;i++){
D[i].purchase+=D[i].trend;
      }
    }
    all+=kai;
  }
  fclose(datafile1);
  fclose(datafile2);
  fclose(datafile3);
  fclose(datafile4);
  fclose(datafile5);
  printf("all= %d\n",all);
  printf("Number of a(t) is %d.\n",Na);
}
float abs_(float x)
{
  float ans=0.0;
  if(x>=0){ans=x;}
  else{ans=-x;}
  return ans;
}
```

A.4 その他のプログラム

A.4.1 ヒストグラム作成ルーチン

```
#include <stdio.h>
#include <stdlib.h>
#define box 5000000
#define limit 20000
main(int argc, char *argv[])
{
  FILE *rfp, *wfp;
  int i,n=0,m=0;
  static int hist[limit];
  static float X[box];
  float d_X,Xmin,Xmax;
  if(argc!=3){
    printf("Input error.\n");
    exit(1);
  }
  if((rfp=fopen(argv[1],"r"))==NULL){
    printf("The read file can't open.\n");
    exit(1);
  }
  if((wfp=fopen(argv[2],"w"))==NULL){
    printf("The write file can't open.\n");
    exit(1);
  }
  for(i=0;i<box;i++){
    X[i]=0.;
  }
  for(i=0;i<=limit;i++){
    hist[i]=0;
  }
  while(fscanf(rfp,"%e",&X[n])!=EOF){
    n++;
  }
  m=n/300;
  Xmin=X[0];
  Xmax=X[0];
  for(i=1;i<n;i++){
    if(X[i]<=Xmin){Xmin=X[i];}
    if(X[i]>=Xmax){Xmax=X[i];}
  }
  printf("Xmax=%f\n",Xmax);
  printf("Xmin=%f\n",Xmin);
  d_X=(float)((Xmax-Xmin)/(m-1));
  for(i=0;i<n;i++){
    hist[(int)((X[i]-Xmin)/d_X)]++;
  }
```

```
  for(i=0;i<=m;i++){
    if(hist[i]!=0){
  fprintf(wfp,"%15f %d\n",Xmin+d_X*(0.5+i),hist[i]);
    }}
  fclose(rfp);
  fclose(wfp);
  printf("n=%d\n",n);
}
```