価格変動のフラクタル分布

饗場行洋
羽田野研究室

2000 年 2 月 29 日

概 要

金融市場での価格の変動という一見物理と無関係な事象が、実は物理と馴染み深い方程式で記述されるようなシステムを内在しているという事実を紹介する。

まず、実際の価格変動を解析し、価格差の確率分布がフラクタル分布で近似できるということを現実の市場価格の最大の特徴と位置付ける。そしてその特徴を再現できる決定論モデルを構築し、決定論モデルを確率過程として近似する。そのとき核になる Langevin 方程式が価格変動のシステムにおける本質的な方程式であることを明らかにする。

ここでフラクタル分布とは、分布関数が $P(x) \propto |x|^{-\beta-1}$ に従うような分布のことをいう。

目 次

1　はじめに

昨今の金融市場ではめまぐるしく価格が変化している。ディーラーたちは長年の経験による知識と勘によってその変化を読み取ろうとする努力を怠っていないが、予想は大きく外れている。その最大の原因は現在使われている金融予測モデルにあると考えられる。現在のモデルでは、実際の市場では日常的に起こっている暴騰や暴落という事象を全くといっていいほど無視しているのである。現代の金融モデルの基盤にはポートフォーリオと呼ばれる数学理論がある。この理論に従い標準偏差の 10 倍より大きい変動が起こる確率を計算すると 10^{-24} 程度の確率になる。しかし実際の市場ではその程度の変動の起こる確率は 10^{-2} くらいのオーダーであり、毎月のように起こっているのである [1]。

実際の価格変動の分布は大きな変動の頻度が正規分布よりはるかに高いような分布である（図 1）[2]。しかし、既存の確率モデルではこの事実を無視している。

この論文の目的は、物理の視点から金融市場を見ることによって、実際の市場価格の振舞の特徴を再現するようなモデルを構築し、価格変動のシステムに内在する本質的な方程式を導出することにある。

結論から先に述べると、価格変動のシステムに内在する本質的な方程式とは、離散的な非線形ランジュバン方程式と呼ばれる確率方程式である。

$$x(t+1) = b(t)x(t) + f(t) \tag{1}$$

ここで $f(t)$ は平均が 0 で分散が有限の値をとる確率変数、つまりホワイトノイズであり、$b(t)$ は非負の値をとる無相関な確率変数である。

この方程式は、掛け算できいてくるノイズと、足し算できいてくるノイズによって次のステップの x の値を逐次決定していくという方程式である。一般に式（1）で記述される過程は RMP (Random Multicative Prosess) と呼ばれ最近注目を集めている [3]。式 (1) は物理の様々な分野で顔をだす。例えば高分子の外力による曲げモード [4]、乱流におけるパッシブスカラー [5, 6]、レーザーのノイズ [7, 8] 等がある。

このような物理と馴染深い方程式で価格変動のメカニズムを記述できるということを以下で説明していく。まず 2 節で実際の価格変動を解析しその特徴をつかむ。次にその特徴を再現するような決定論的モデルを 3 節で構築し、4 節ではそこから得られたデータをもとに確率モデルで近似する。これにより、Langevin 方程式を導出する。

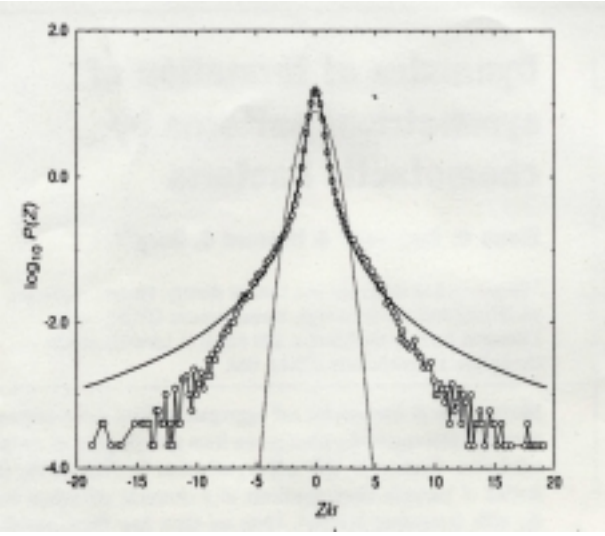

図 1: S&P500 における単位時間当りの価格変動の確率密度分布。実測値は通常よく仮定する正規分布（点線）よりフラクタル分布（実線）に近い。参考文献 [2] による。

2　実際の価格変動の解析

激しく変動する市場価格へのアプローチは様々な方法が考えられるが、ここで注目するのは価格差 (単位時間当たりの価格の変動量) である。つまり、どのような価格変動がどのような確率で発生しているのかに注目する [2]。図 1 は、1984 年から 1989 年まで 6 年分の S&P500 指数 (Standard & Poor's 500 index) のデータの解析から得られた価格差の確率分布である。実際の価格差の分布は正規分布よりはるかに大きな変動が高い頻度で起こっていることを示している。価格差の分布は大きな変動のところでベキ乗法則に従っており、さらに大きな変動のところでは指数関数的に減衰している。

この論文では価格差の分布がベキ乗法則に従う、つまりフラクタル分布に従うということを実際の価格変動の最も重要な特徴であると考える。

3　決定論的モデルでの実際の価格変動の再現

株価や為替の変動には様々な要因が考えられる。しかし、統計物理的視点に立つと、できるだけ単純なモデルの構築が重要であると考えられる。したがって多数のディーラーが自己の利潤のみを追及し、ただ１つの銘柄について売買するという、売買の素過程のみに注目したモデルを考える [9]。

3.1　決定論的モデルの概要

いま、N 人のディーラーがそれぞれ希望買値 B_i と希望売値 S_i を持ち、安く買って高く売ろうとしている系を考える。利潤を得るためには当然、

$$S_i - B_i > 0 \tag{2}$$

が成り立っていなければならない。さらにここでは簡単のために、各々のディーラーの売値と買値の差を一定とし、その値を

$$S_i - B_i = \Lambda \tag{3}$$

とおく。また、取り引きが成立する条件は i 番目と j 番目のディーラーに対して

$$B_i \geq S_j \tag{4}$$

となる場合である。式 (3)(4) により結局取引が成立する条件として

$$B_i - B_j \geq \Lambda \tag{5}$$

を得る。以上の仮定により、全てのディーラーに対して買値 $\{B_i\}$ を考えるだけでよくなる。

市場に N 人のディーラーがいて株の売買を行っているとすると、市場で取引が成立するためには、全てのディーラーのなかで最大の希望買値と最小の希望売値の間で式 (5) が成り立つ必要がある。よって取引成立の条件は

$$\max_i B_i(t) - \min_i B_i(t) \geq \Lambda \tag{6}$$

となる。ここで $\max_i \sim$、$\min_i \sim$ はそれぞれ、i を変化させたときの「$\sim$」の最大値と最小値を表す。

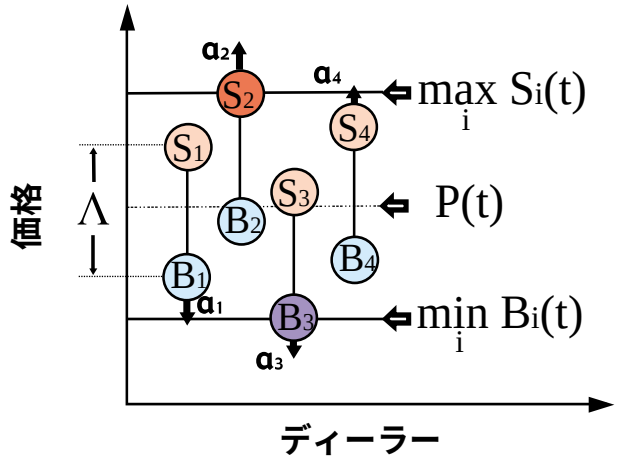

図 2: 取引の概念図。この図では 2 番目のディーラーが売り手、3 番目のディーラーが買い手である。

市場価格 $P(t)$ は売買が成立したときの売値と買値の中間の値とし、売買が起るまで市場価格は変化しないものとする。従って $P(t)$ は次のように定義できる（図 2）：

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

ここで、時刻 t における最大と最小の希望価格の差を

$$L(t) = \max_i B_i(t) - \min_i B_i(t) \tag{7}$$

とおいた。

次に、ディーラーは次のルールに従い買値（売値）を時間発展させていくとする：

$$B_i(t+1) = B_i(t) + a_i(t) + c\Delta P_{\text{prev}} \tag{8}$$

ここで、ΔP_{prev} は寸前に生じた取引による価格差であり（図 3）、その影響の強さをコントロールするパラメーターを c とする。

また $a_i(t)$ は各ディーラーの個性を表す項で、以下のように決める。いま簡単のために取引が起ったときに、売り手は買い手に、買い手は売り

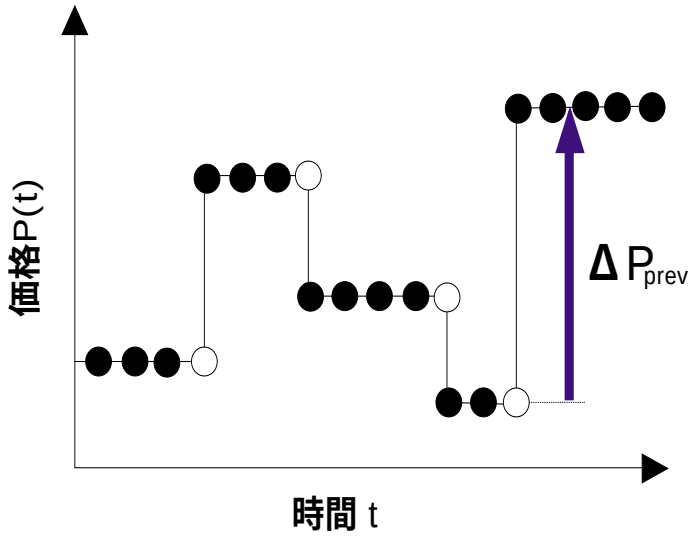

図 3: 価格変動の概念図。ΔP_{prev} は直前に起った取引で生じた価格差である。

手に立場を変えなければならないとすると、$a_i(t)$ の時間発展を次のようにする：

$$a_i(t+1) = \begin{cases} |a_i(t)| \\ \quad \text{（取引が生じたときに売り手のディーラー）} \\ -|a_i(t)| \\ \quad \text{（取引が生じたときに買い手のディーラー）} \\ a_i(t) \\ \quad \text{（取引が生じなかったときの全てのディーラーと} \\ \quad \text{取引が生じたときに売り手と買い手以外のディーラー）} \end{cases}$$

売り手は売れるまで自らの売値を下げ続け、買い手は買えるまで買値を上げ続けると仮定し、$a_i(t) < 0$ のとき売り手、$a_i(t) > 0$ のとき買い手を意味する。

初期値 $B_i(0), a_i(0)$ はそれぞれ、$[-\Lambda/2, \Lambda/2]$、$[-\alpha, \alpha]$ の一様乱数で与える。また、$t = 0$ では $P(0) = 0$、$\Delta P_{\mathrm{prev}} = 0$ とする。

このモデルの推移には確率的な要素は全くなく完全に決定論的である。パラメータはディーラー数 N、$a_i(t)$ の初期ゆらぎを決める α、閾値 Λ、過去の価格変動量 ΔP_{prev} の影響をコントロールする定数 c の 4 つだけである。

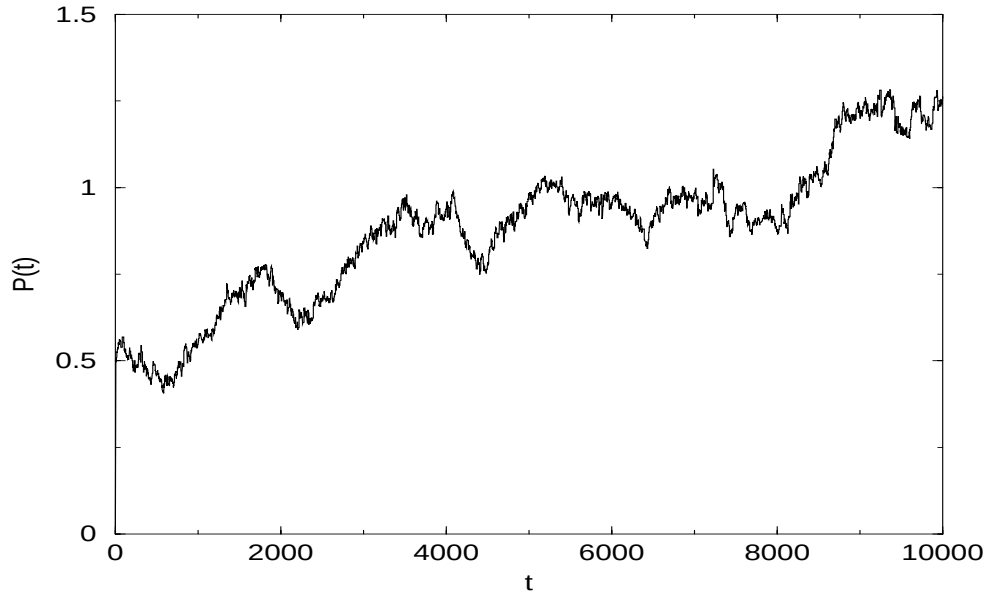

図 4: $c = 0.0$ における市場価格 $P(t)$ の振舞い。

3.2 決定論的モデルによるシミュレーションの結果

$N = 100$、$\Lambda = 1.0$、$\alpha = 0.01$ に固定し、c のみを変化させて計算した。このときの $P(t)$、$\Delta P(t)$ の振舞いをそれぞれ図 4〜7、図 8〜11 に示す。ここで、$\Delta P(t)$ は

$$\Delta P(t) \equiv P(t) - P(t-1) \tag{9}$$

で定義される価格差である。c が大きくなるに従って、急激で大きな変動が現れていることがわかる。

次に、$c = 0.0$、$c = 0.3$ での $\Delta P(t)$ の確率分布を図 12、13 に示す。$c = 0.0$ のときというのは、各々のディーラーが過去の影響を受けずに売買を行っている状態を意味する。近似曲線 $U(\Delta P)$ は以下のように定義される平均 0 のラプラス分布である：

$$U(\Delta P) = \frac{1}{2\sigma} \exp\left(-\frac{|\Delta P|}{\sigma}\right) \tag{10}$$

この分布は確率モデルの構築時に必要になってくる。

$c = 0.3$ のときの ΔP の分布は ΔP の大きいところで明らかに正規分布からはずれている。ΔP の分布の裾野の広がりはフラクタル分布に近い

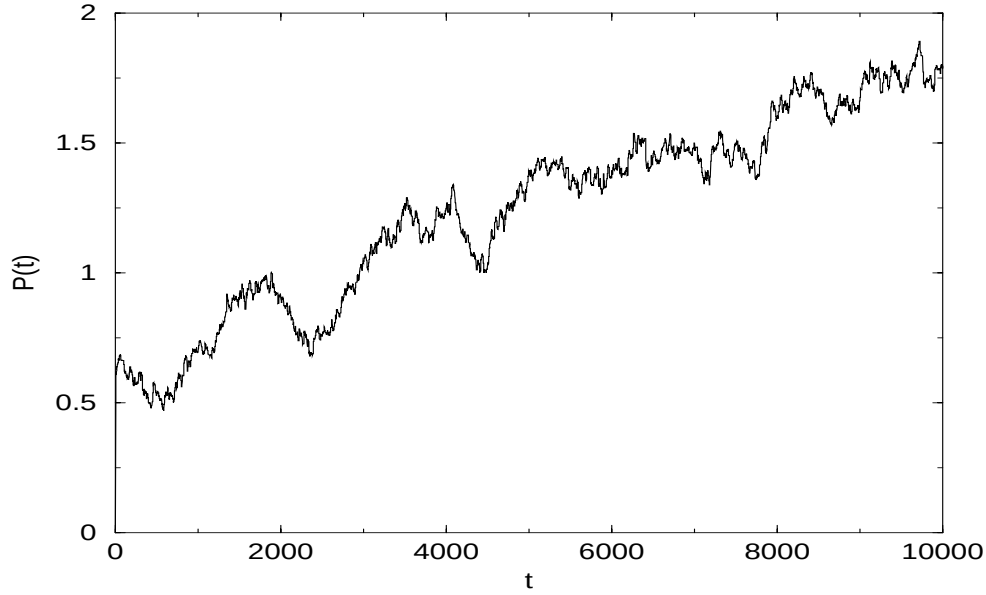

図 5: $c = 0.1$ における市場価格 $P(t)$ の振舞い。

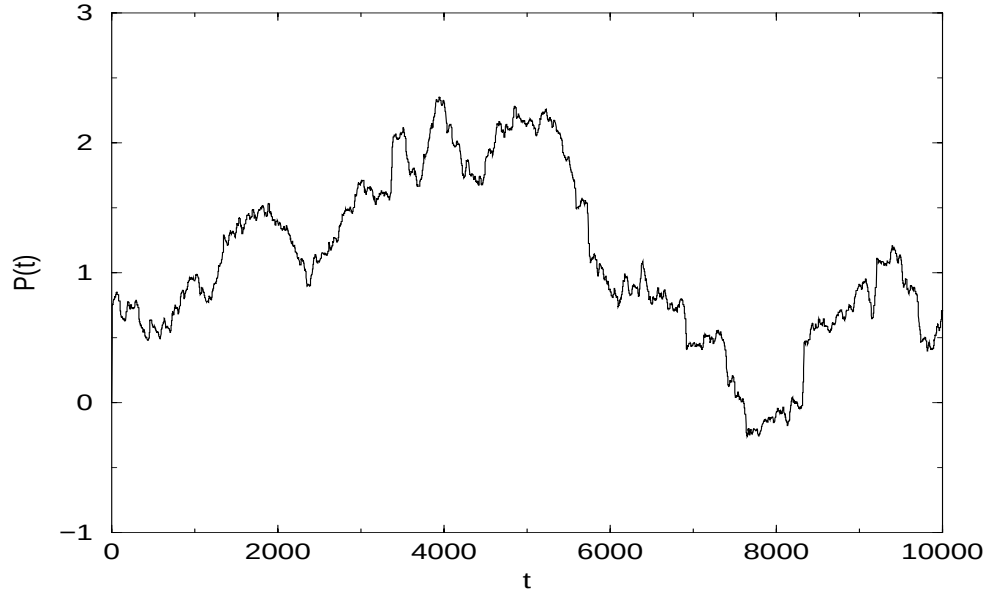

図 6: $c = 0.2$ における市場価格 $P(t)$ の振舞い。

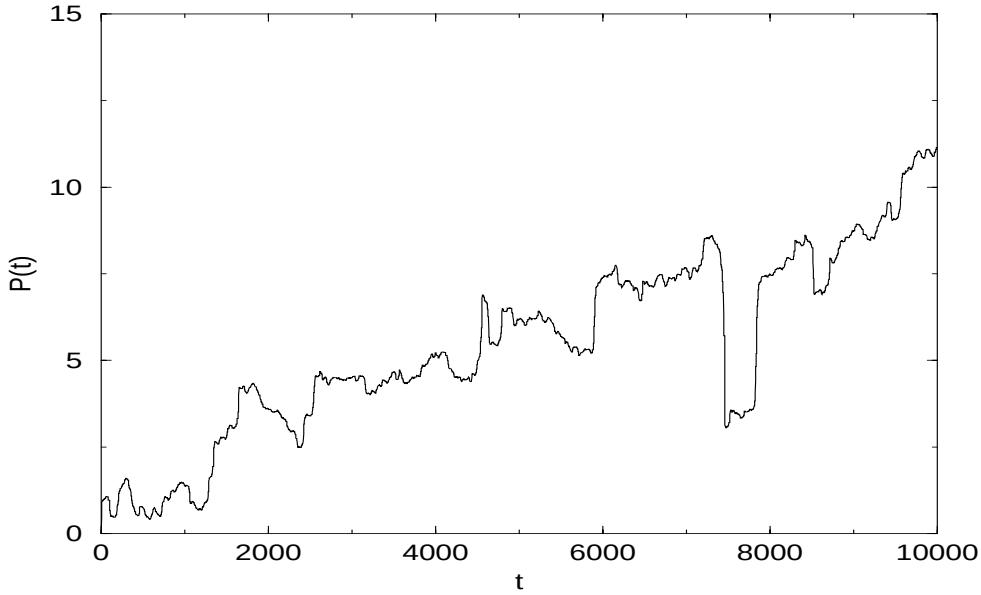

図 7: $c = 0.3$ における市場価格 $P(t)$ の振舞い。

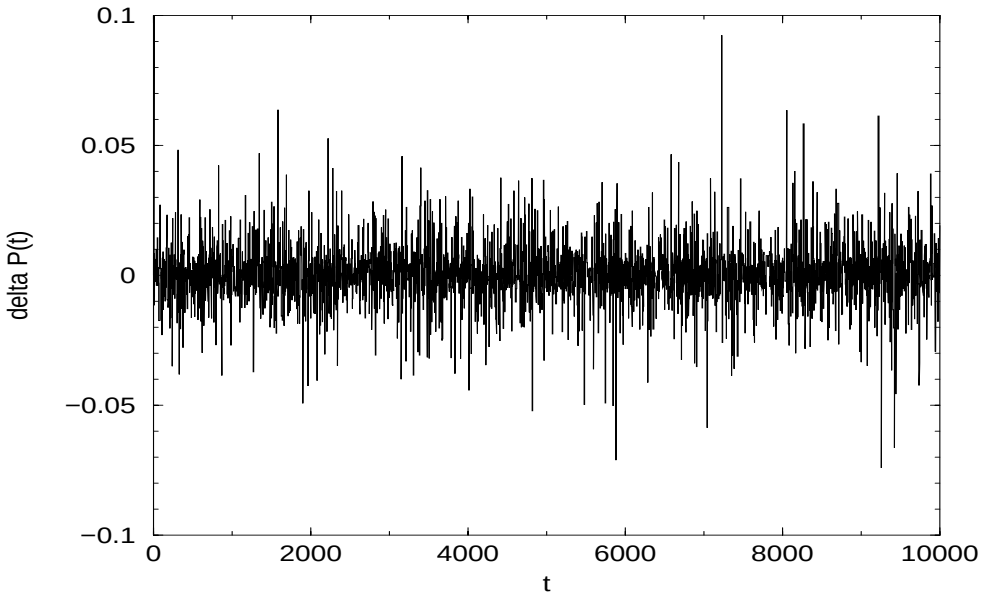

図 8: $c = 0.0$ における価格差 ΔP の振舞い。

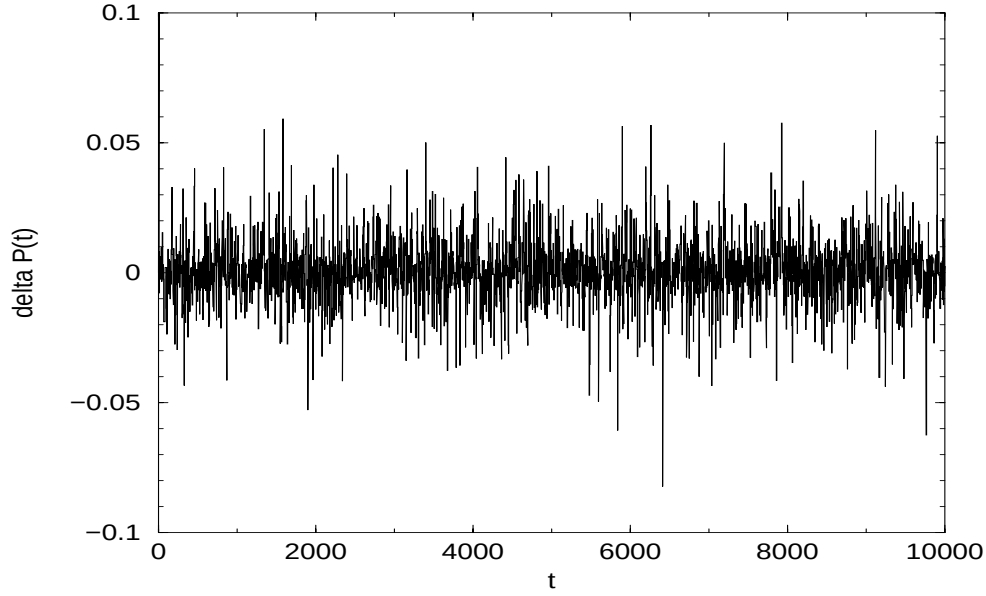

図 9: $c = 0.1$ における価格差 ΔP の振舞い。

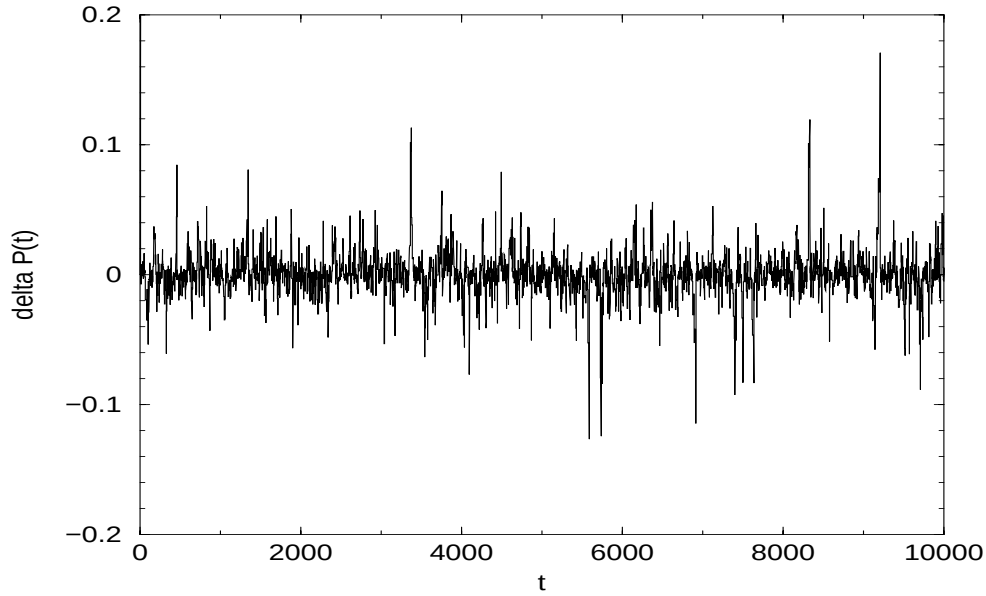

図 10: $c = 0.2$ における価格差 ΔP の振舞い。

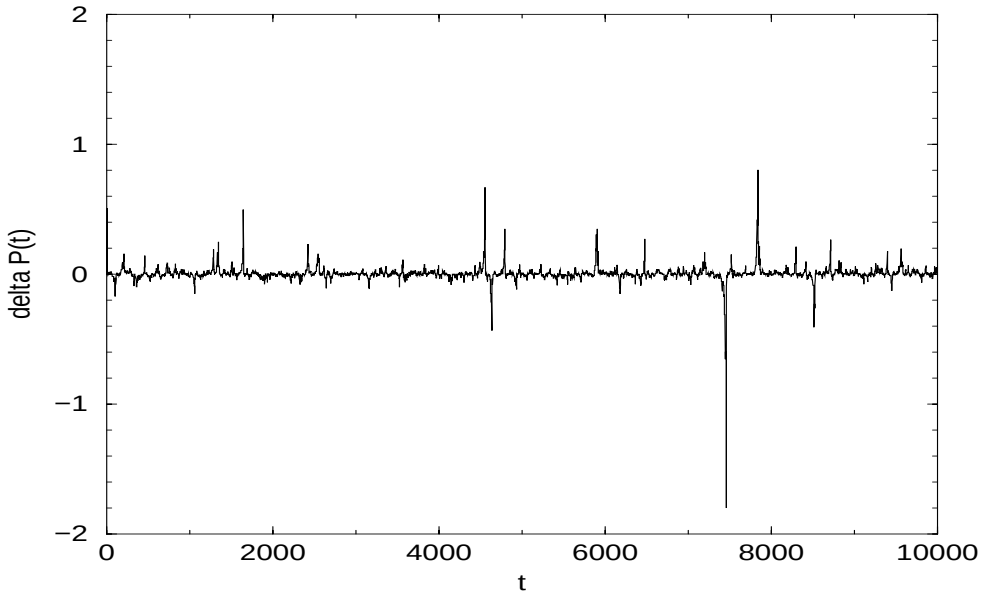

図 11: $c = 0.3$ における価格差 ΔP の振舞い。

ことがわかる。このような分布をする価格差を生み出す確率モデルを次の節で構築する。

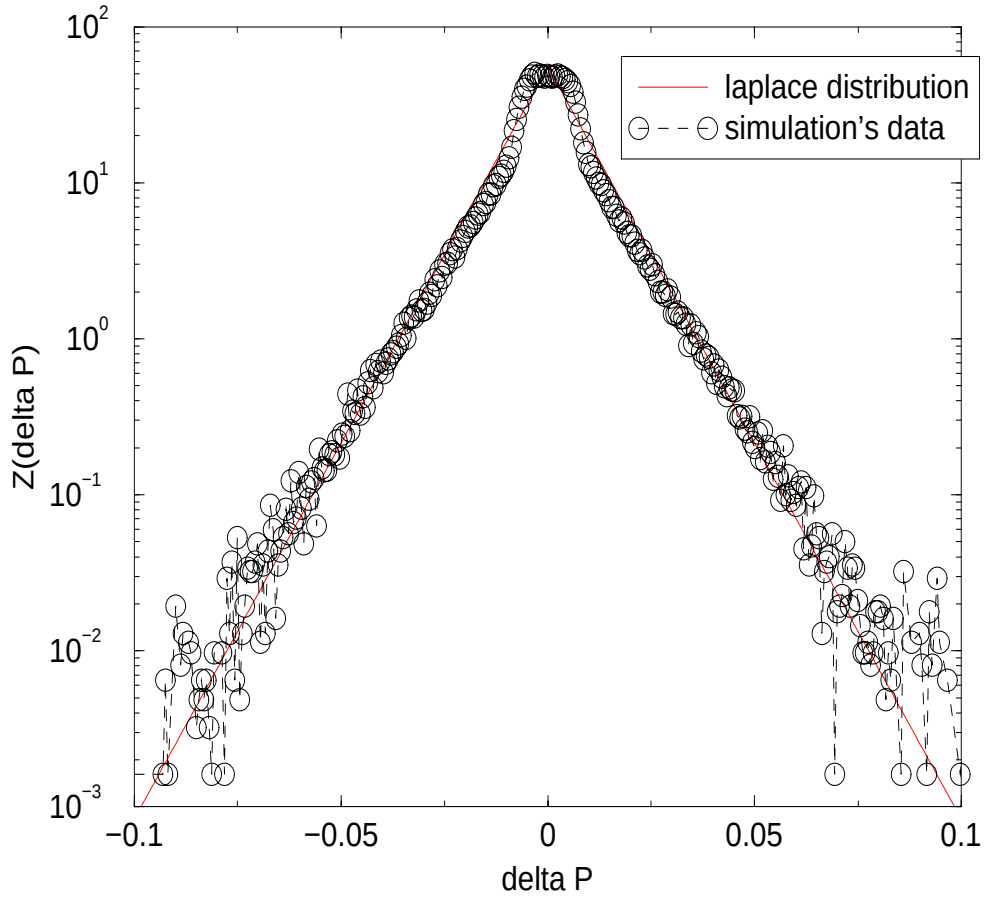

図 12: $c = 0.0$ でのシミュレーションによる ΔP の分布（◯印）。実線は $\sigma = 0.009$ のラプラス分布 $U(\Delta P)$。

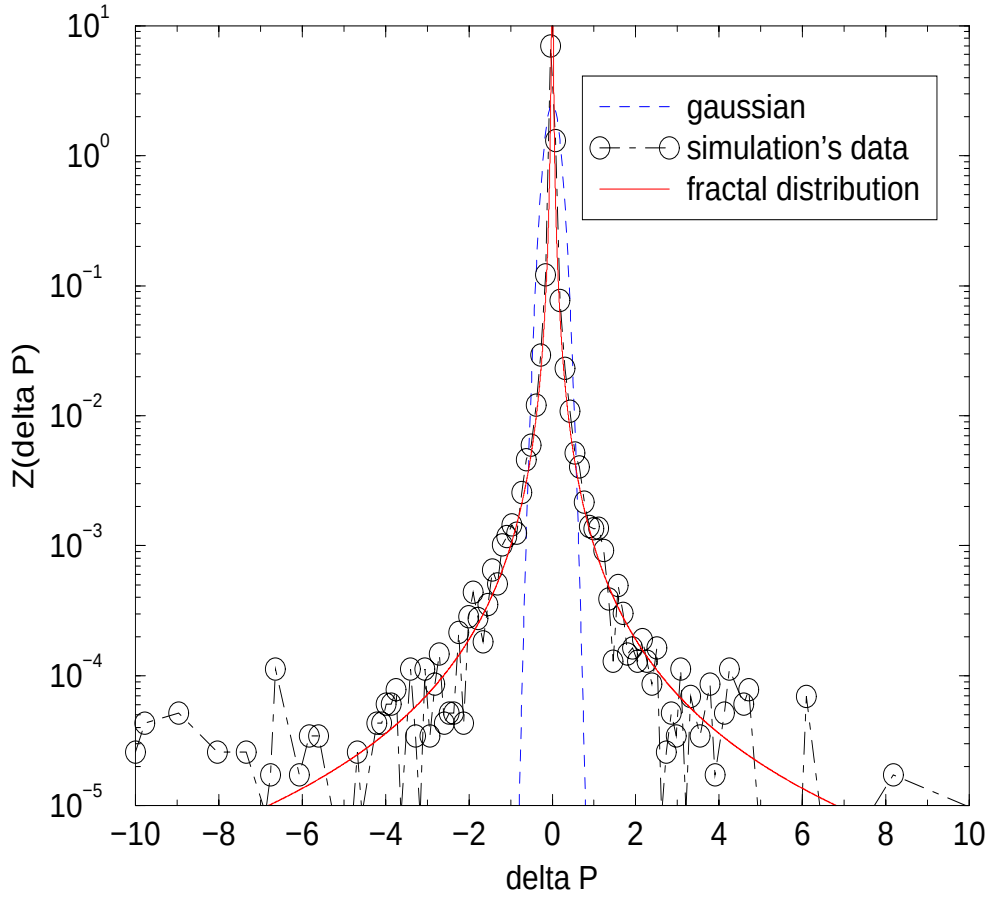

図 13: $c = 0.3$ でのシミュレーションによる ΔP の分布（◯印）。点線は平均と分散をシミュレーションのデータに合せた正規分布、実線は $\beta = 1.4$ のフラクタル分布である。シミュレーションによる ΔP の分布は $|\Delta P|$ の大きいところで明らかに正規分布からはずれている。ΔP の分布の裾野の広がりはフラクタル分布に近いことがわかる。

4　決定論的モデルを再現する確率モデルの構築

4.1　確率方程式の導出

確率モデルでは価格差 Δp を以下のように定義した [9]：

$$\Delta p_s \equiv P(\tau(s)) - P(\tau(s-1)) \tag{11}$$

添字 s は取引回数をあらわす。$\tau(s)$ はその取引の起った時刻である。結論から先にいうと、Δp_s は近似的に次の漸化式を満す：

$$\Delta p_{s+1} = c n_s \Delta p_s + \phi_s \tag{12}$$

式 (12) は掛け算によるノイズと足し算によるノイズによって次の値を決めるという、Langevin 方程式（1）と同じ形をしている。なお、n_s は離散的指数分布 $W(n)$ に従う確率変数であり、ϕ_s はラプラス分布 $U(\phi)$ に従う確率変数である。$W(n)$、$U(\phi)$ はそれぞれ

$$W(n) = \sum_{k=1}^{\infty} \frac{1-e^{-\gamma}}{e^{-\gamma}} \exp(-\gamma k)\, \delta(n-k) \tag{13}$$

$$U(\phi) = \frac{1}{2\sigma} \exp\left(-\frac{|\phi|}{\sigma}\right) \tag{14}$$

である。

Δp がこのような漸化式に従うと設定し、$W(n)$、$U(\phi)$ を上の様に見積もった根拠を以下に述べる [9]。

前述の決定論的モデルは大きく 2 つの部分に分けて考えことができる。前回の取引の影響をコントロールするパラメータ c に依る部分と c に依らない部分である。いいかえると前回の取引に依る部分と依らない部分である。

まず簡単なのは c に依らない部分である。c に依らないということは、前後の取引で起った価格の変動とは独立であるということであり、漸化式（12）においては足し算のノイズとして考えられる。これについては決定論モデルおいて $c = 0.0$ としてシミュレーションを行えばデータが得られる。それにより得られた ΔP の分布が図 12 であり、ラプラス分布 U で近似した。確率モデルでは $\sigma = 0.01$ とした。

つぎに c に依る部分について考える。式（8）より明らかなように全てのディーラーは時間発展とともに一斉に $c\Delta P_{\mathrm{prev}}$ だけ売値（買値）を変

化させる。全てのディーラーが同時に $c\Delta P_{\rm prev}$ だけ売値（買値）を変化させるということは、取引価格が $c\Delta P_{\rm prev}$ だけ変化するということを意味する。つまり、次回の価格は c による項として $cn_s\Delta P_{\rm prev}$ だけ変化している。ここで、n_s は

$$n_s \equiv \tau(s+1) - \tau(s) \tag{15}$$

で定義される取引の時間間隔である。

$\Delta P_{\rm prev}$ は $\Delta p(s)$ に他ならない。次回の価格が $cn_s\Delta P_{\rm prev}$ だけ変化しているのだから、次回の価格差 Δp_{s+1} は $cn_s\Delta p(s)$ という項をもっているはずである。決定論モデルから抽出した n_s の分布は図 14 の様であり、これを $W(n)$ で近似した。確率モデルでは $\gamma = 0.4$ とした。

以上のことから、Δp が式 (12) のような漸化式に従うと近似しシミュレーションを行った。

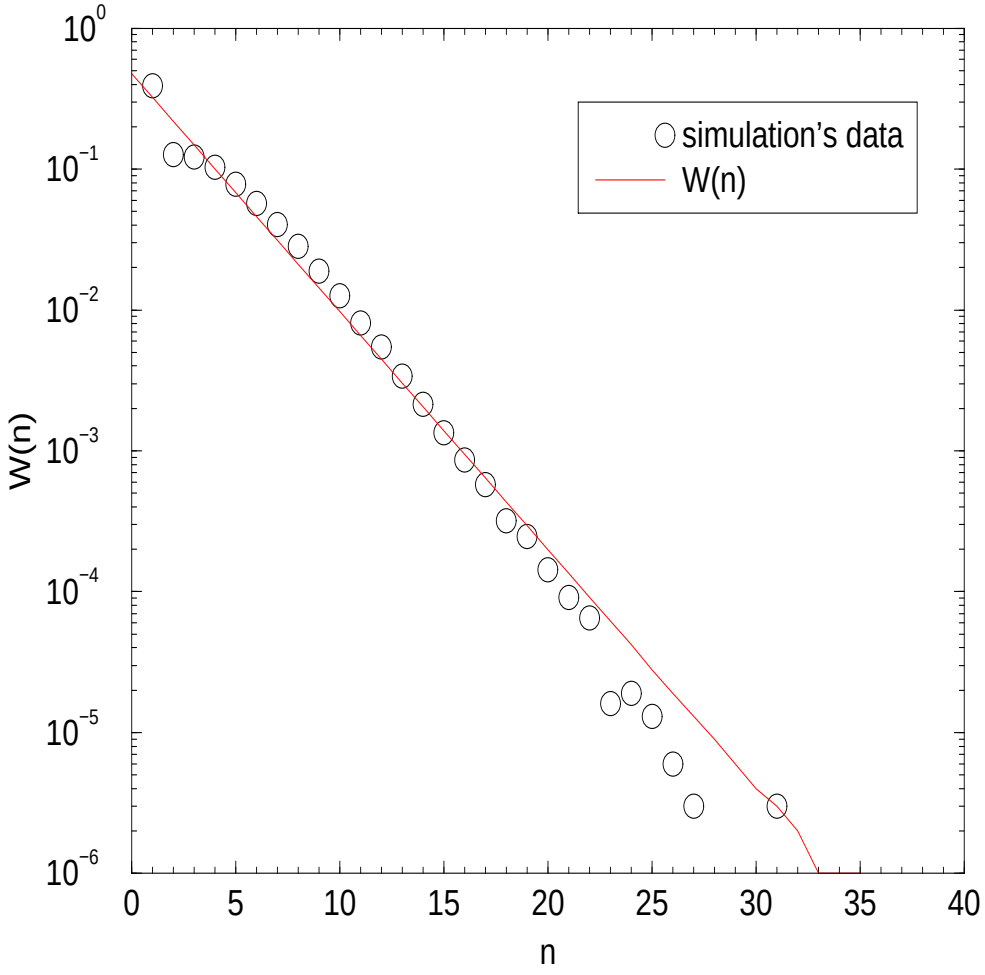

図 14: 決定論モデルによるシミュレーションにより得た n_s の分布（◯印）。実線は $\gamma = 0.4$ の離散的指数分布 $W(n)$。

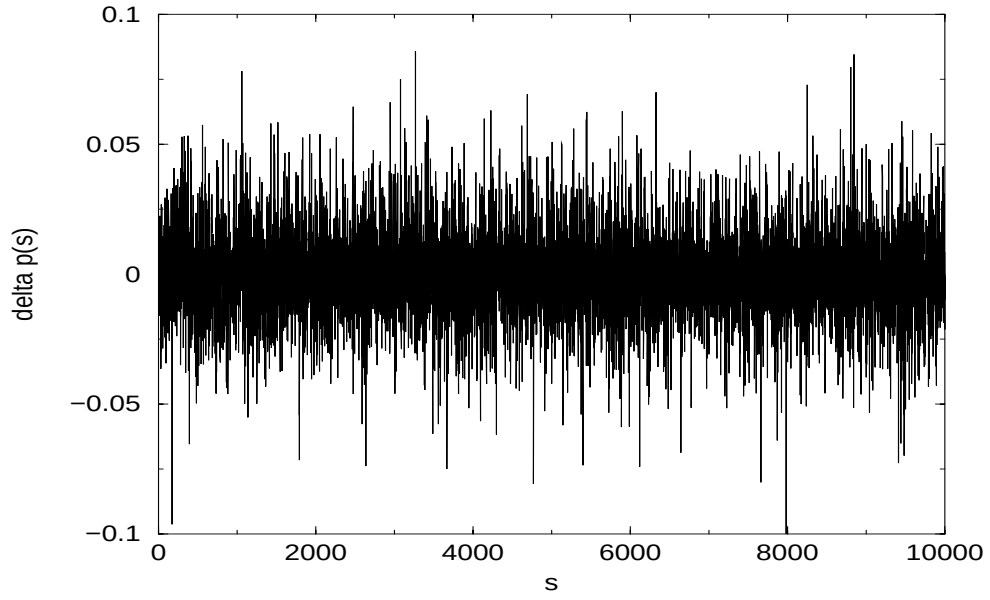

図 15: $c = 0.0$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格差 Δp_s の時系列。

4.2 確率モデルによるシミュレーションの結果

式 (12) によって発生させた Δp を足し込んでいったものを価格とする。つまり価格 p_s は i を正の整数として

$$p_s = \sum_{i=0}^{s} \Delta p_i \tag{16}$$

とかける。

c を 0.0 から 0.3 まで変化させたとき、発生させた価格差 Δp_s を図 15〜18 に、式（16）で定義した価格 p_s を図 19〜22 に示す。横軸が取引回数 s であることから、一概に決定論モデルとの比較はできないが、c が大きくなるにつれて大きな変動が起ってくるという傾向は一致しているといえる。

$c = 0.3$ のときの価格差 Δp の確率分布を図 23 に示す。やはり Δp の分布は $|\Delta p|$ の大きいところで明らかに正規分布からはずれている。実線の $\beta = 1.4$ のフラクタル分布にシミュレーションのデータがきれいにのっていることがわかる。

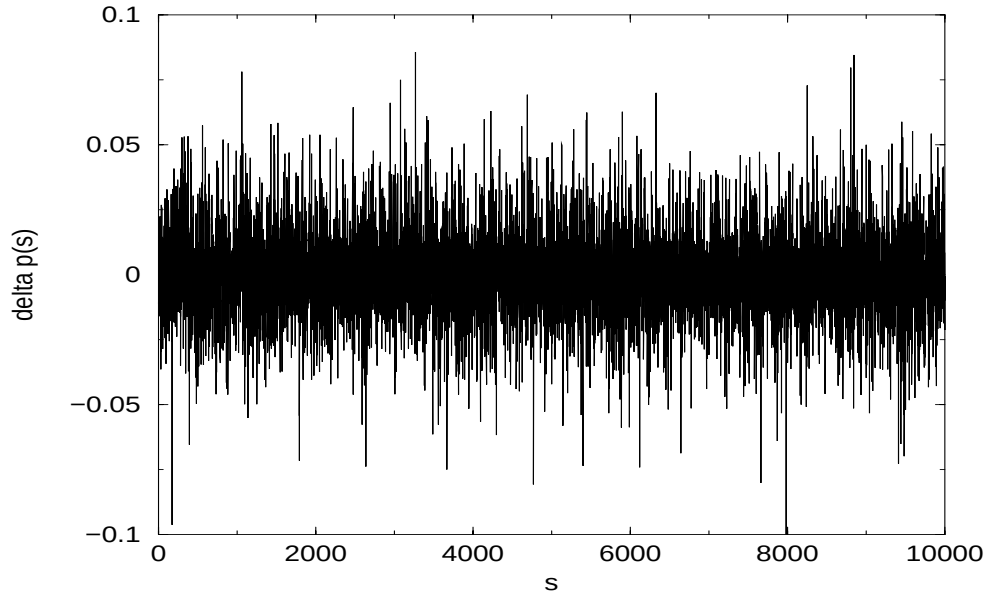

図 16: $c = 0.1$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格差 Δp_s の時系列。

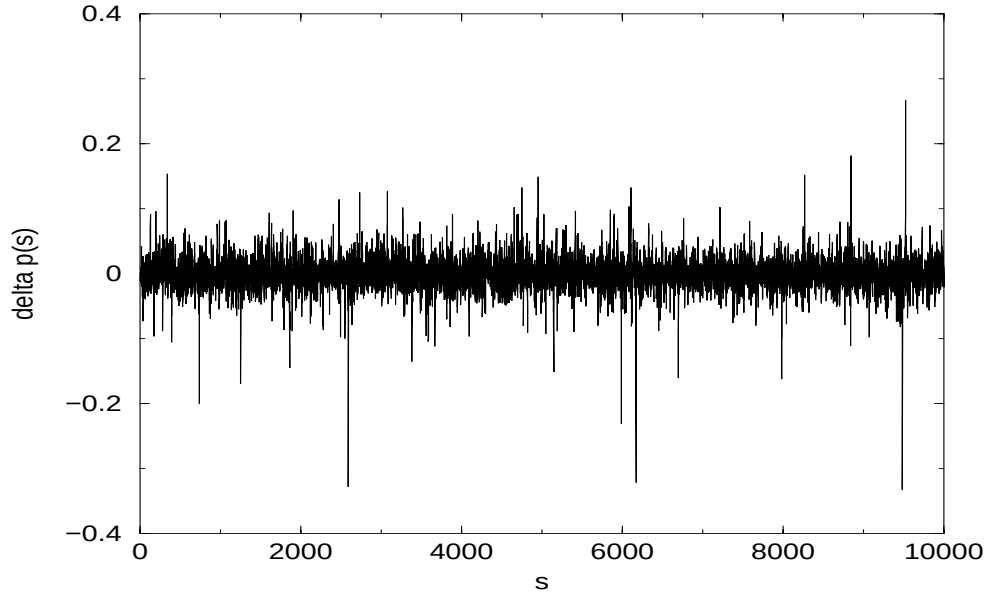

図 17: $c = 0.2$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格差 Δp_s の時系列。

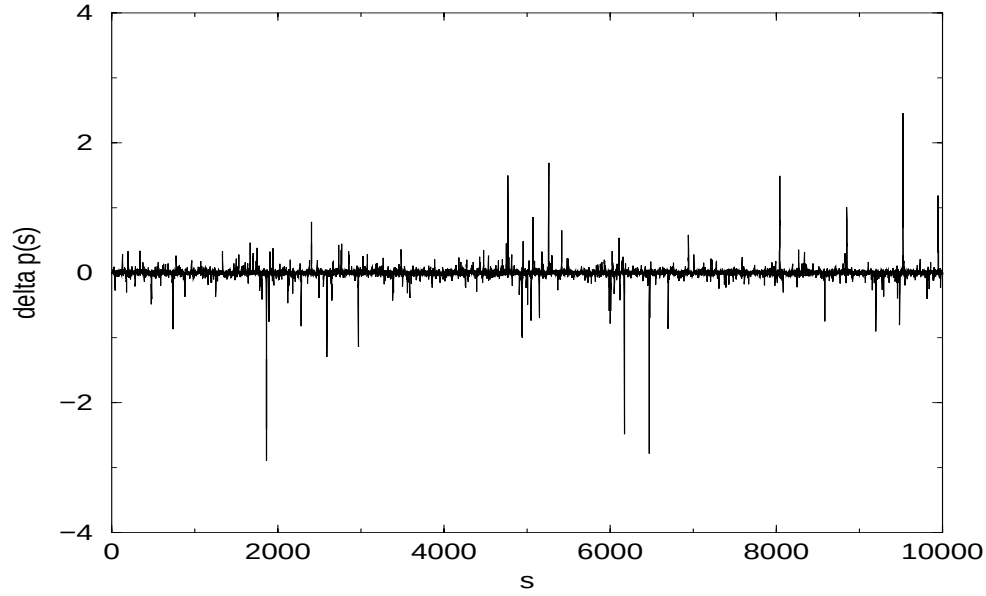

図 18: $c = 0.3$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格差 Δp_s の時系列。

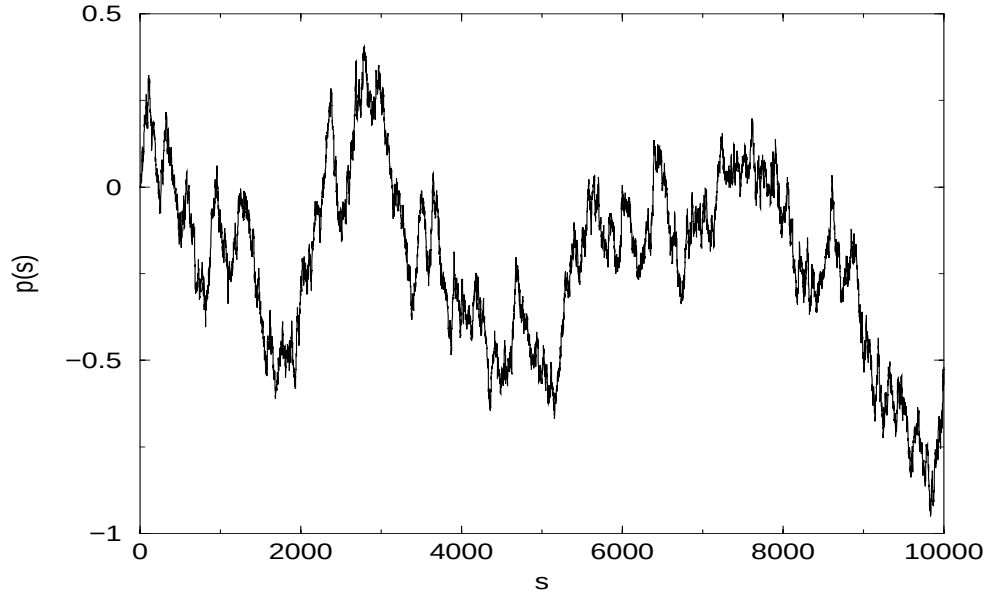

図 19: $c = 0.0$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格 p_s の時系列。

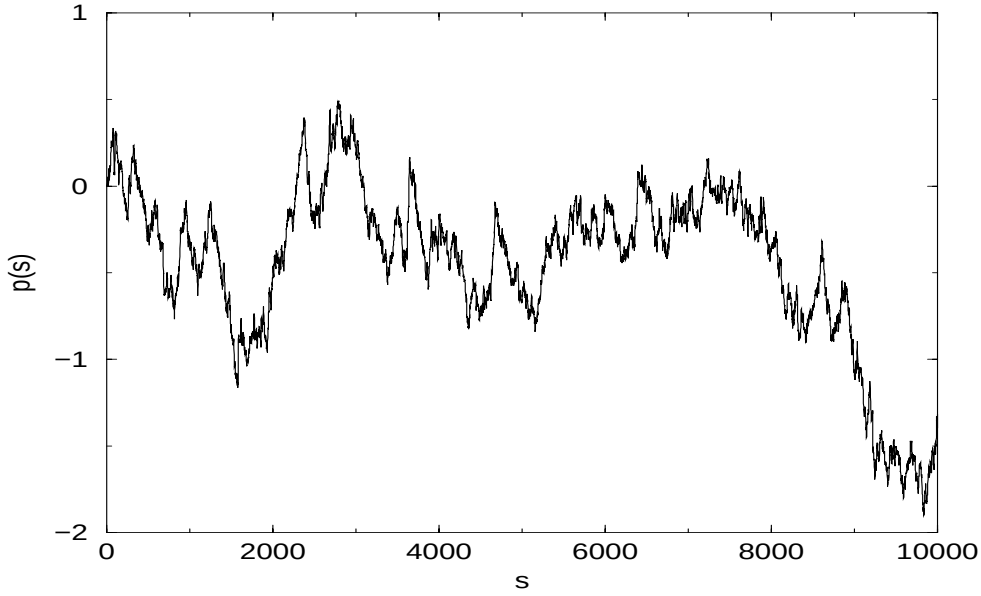

図 20: $c = 0.1$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格 p_s の時系列。

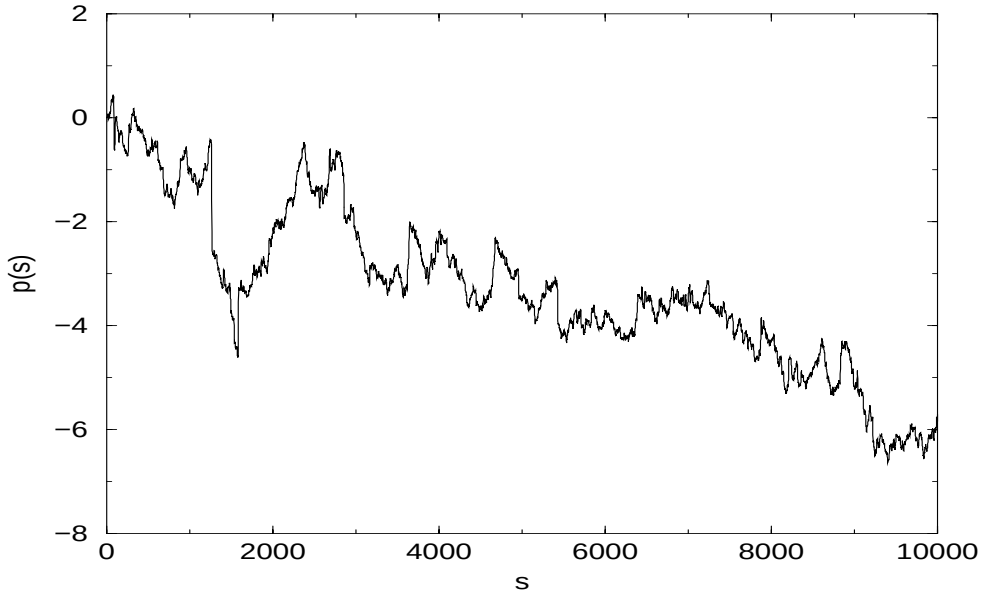

図 21: $c = 0.2$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格 p_s の時系列。

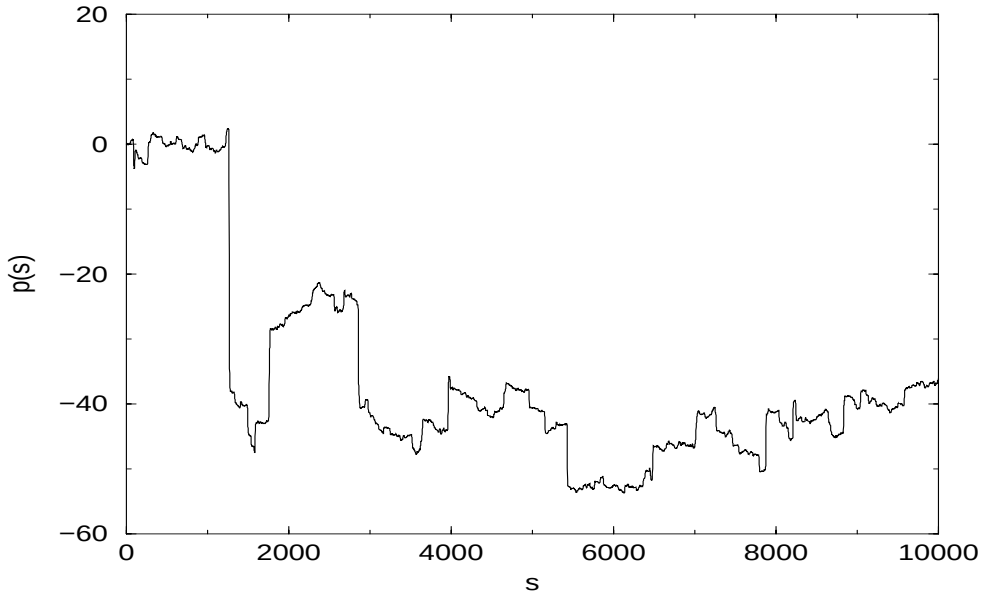

図 22: $c = 0.3$ における確率モデルによるシミュレーションでの、取引回数 s に対する価格 p_s の時系列。

実際の価格変動の最大の特徴と位置付けたベキ乗則が良く再現されていることから、Langevin 方程式が価格変動のメカニズムの本質をよく記述しているといえる。

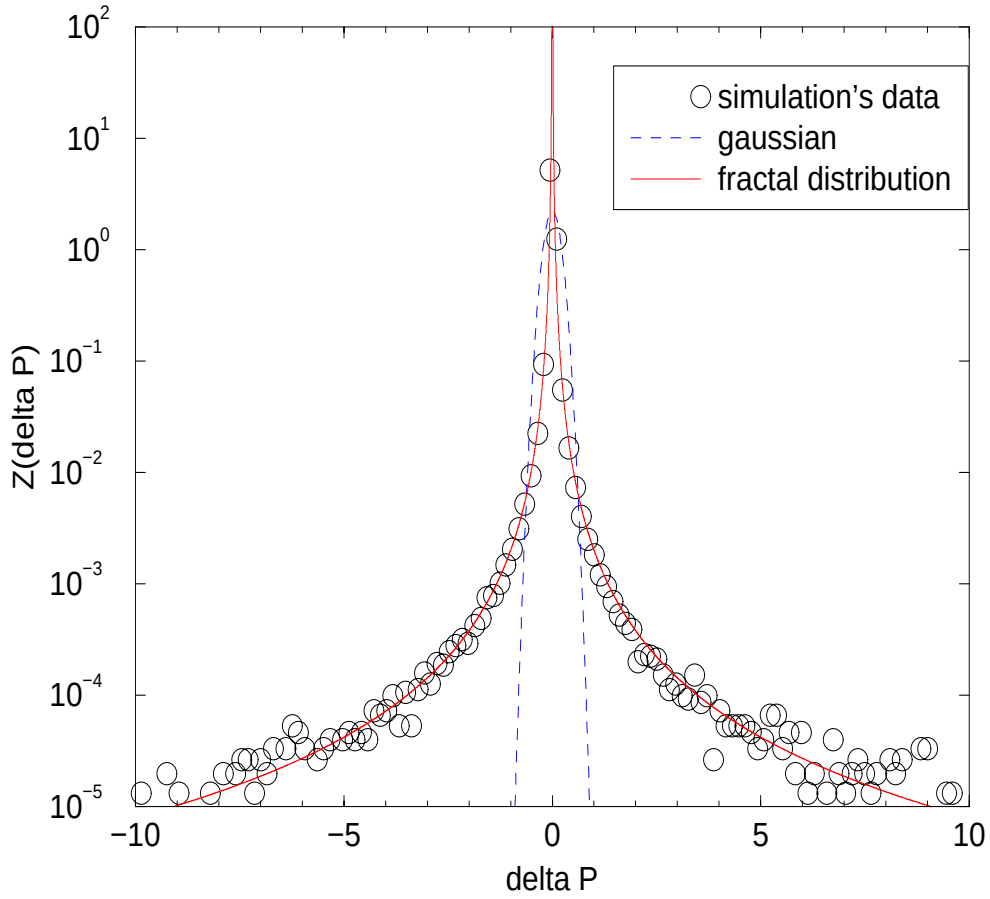

図 23: $c = 0.3$ での確率モデルでのシミュレーションによる ΔP の分布（◯印）。点線は平均と分散をシミュレーションのデータに合せた正規分布、実線は $\beta = 1.4$ のフラクタル分布である。シミュレーションによる Δp の分布は $|\Delta p|$ の大きいところで明らかに正規分布からはずれている。Δp の分布の裾野の広がりはフラクタル分布に近いことがわかる。

4.3 c による価格差 Δp の確率分布の変化

上で述べたように価格差 Δp の確率分布の形は c に依って変化する。$c = 0.0$ のときの価格差 Δp の確率分布は式（12）より明らかなようにラプラス分布 $U(\phi)$ である（図 24）。一方、$c = 0.3$ のときの Δp の確率分布はフラクタル分布で近似できる。つまり、$c = 0.0$〜0.3 の間で Δp の確率分布はラプラス分布からフラクタル分布へと変化していると考えられる。その変化の様子を観測するために以下の量を定義した：

$$\theta(c) = \{\max_{\Delta p} Z(\Delta p)\}\sigma \tag{17}$$

ここで $Z(\Delta p)$ は Δp の確率分布関数である。また σ は

$$\sigma = \sqrt{\langle |\Delta p|^2 \rangle} \tag{18}$$

で定義する Δp の標準偏差である。図 25 に $\max_{\Delta p} Z(\Delta p)$ と σ の概念図を、図 26 には c、Δp と $Z(\Delta p)$ の関係を示す。

$c = 0.0$ のときの $Z(\Delta p)$ はラプラス分布 $U(\phi)$ なので、$\theta(0)$ は解析的に求めることができる：

$$\theta(0) = \frac{1}{\sqrt{2}} \tag{19}$$

一方、$Z(\Delta p)$ がフラクタル分布の場合は σ が ∞ なので、θ は ∞ である。よって c が増大するに従って $\theta(c)$ が $1/\sqrt{2}$ からどのように離れてゆくか調べれば、どのように分布の形が変ってゆくかがわかる。

σ の変化を図 27 に示す。c の大きいところで σ が急激に増加していく様子がわかる。

また、$\max_{\Delta p} Z(\Delta p)$ は $Z(\Delta p)$ がフラクタル分布ならば ∞ に発散すると期待されるが、図 28 を見ると $\max_{\Delta p} Z(\Delta p)$ が、c の増大に伴い減少してゆくのがわかる。これは有限サイズ効果によるものであると考えられる。しかしここで重要なことは、図 24 と図 23 を比べれば解るように c の増大に伴い $\max_{\Delta p} Z(\Delta p)$ が減少しても分布の形はフラクタル分布に近づいてゆくということである。

$\theta(c)$ の振舞いを $1/\sqrt{2}$ と比較して図 29 に示す。c が大きくなると $\theta(c)$ は $1/\sqrt{2}$ から急激に離れてゆくのがわかる。さらに、図 29 の拡大図の図 30 を見ると、$\theta(c)$ は c が小さいところでは $\max_{\Delta p} Z(\Delta p)$ の減少の影響で減少し、c が大きくなると σ の増大の効果で増大してゆくことがわかる。

以上のことからおよそ $c \simeq 0.2$ 付近でラプラス分布からフラクタル分布への変化が起ったということができる。

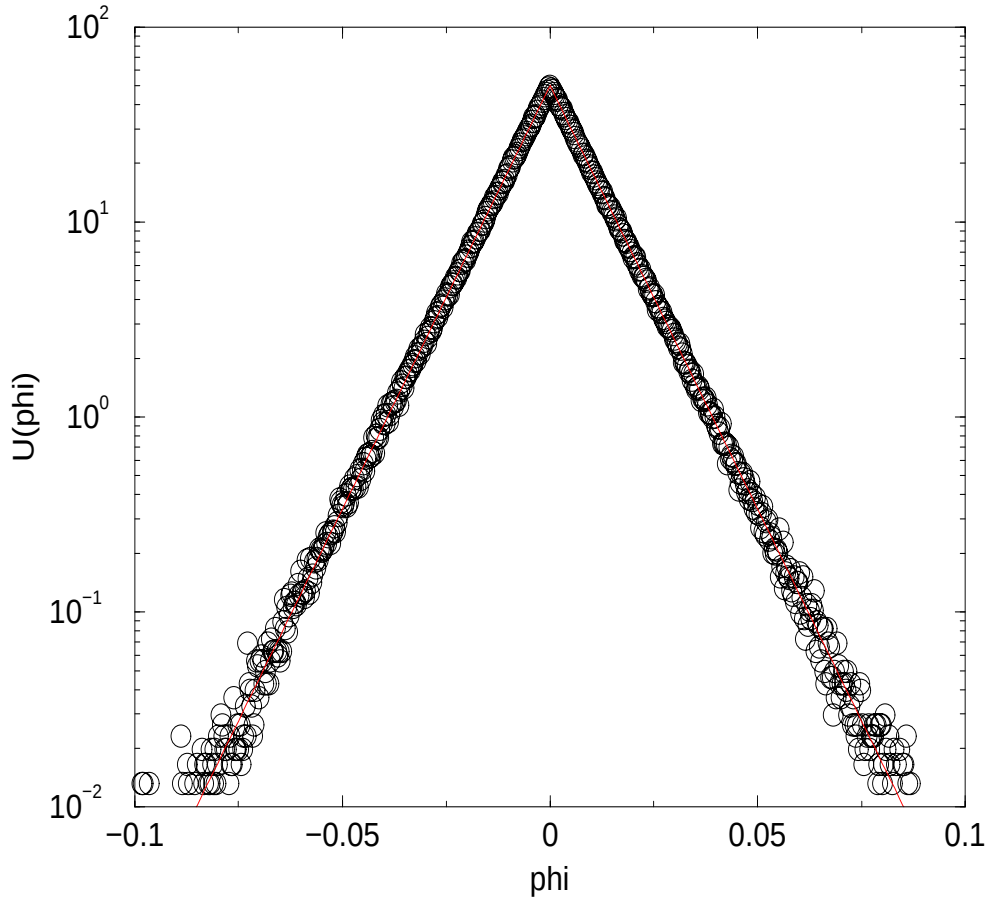

図 24: $c = 0.0$ のときの Δp の確率分布（◯印）。実線は $\gamma = 0.4$ の $U(\phi)$。式（12）より明らかなように $c = 0.0$ のときの Δp は ϕ そのものである。Δp の確率分布は $U(\phi)$ できれいに近似できている。

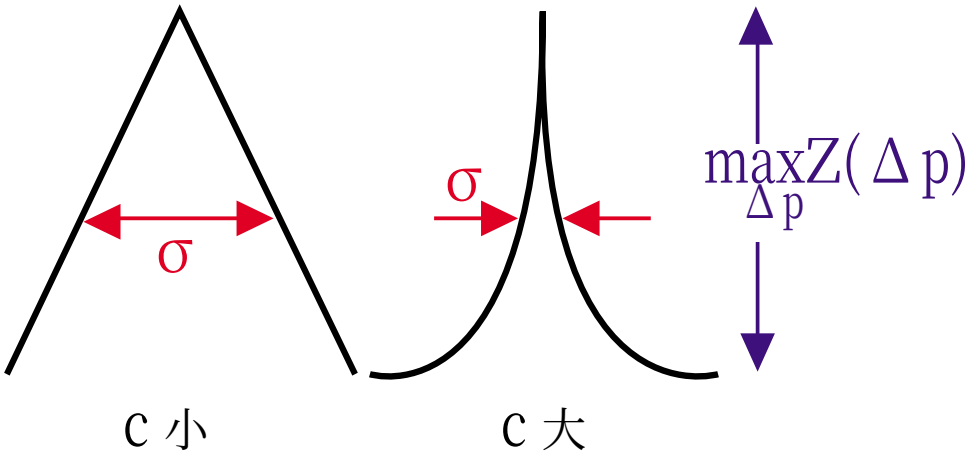

図 25: σ と $\max_{\Delta p} Z(\Delta p)$ の概念図。

5　まとめ

私はまず、実際の価格変動を解析し、価格差の確率分布がフラクタル分布で近似できるということを現実の市場価格の最大の特徴と位置付けた。そしてその特徴を再現できる決定論モデルを構築し、決定論モデルを確率過程として近似することができた。そのとき核になる式（12）であらわされる Langevin 方程式が価格変動のシステムにおける本質的な方程式であることを明らかにした。

ここで式（12）における $U(\phi)$、$W(n)$ についてすこし触れたい。この確率モデルでは $U(\phi)$ をラプラス分布、$W(n)$ を離散的指数分布とおいた。その理由は決定論モデルと対応付けるためであったが、実は $U(\phi)$ として正規分布を選んでも、$W(n)$ として連続的な指数分布を選んでも同様の結果が得られる（図 31〜33）。このことからも、1 節で紹介した Langevin 方程式が価格変動のシステムの本質をよく捉えているといえる。

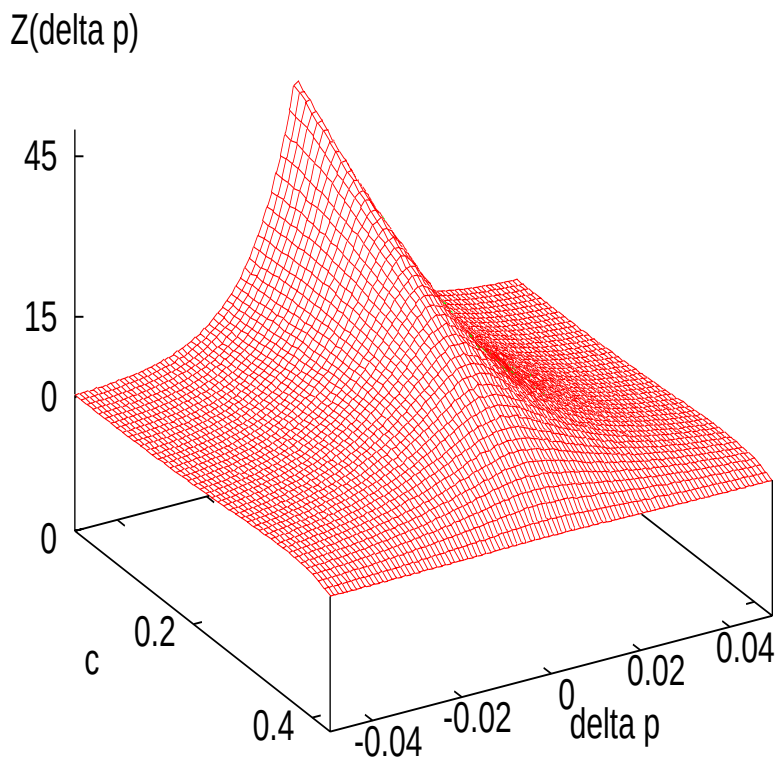

図 26: c、Δp と $Z(\Delta p)$ の 3D グラフ。

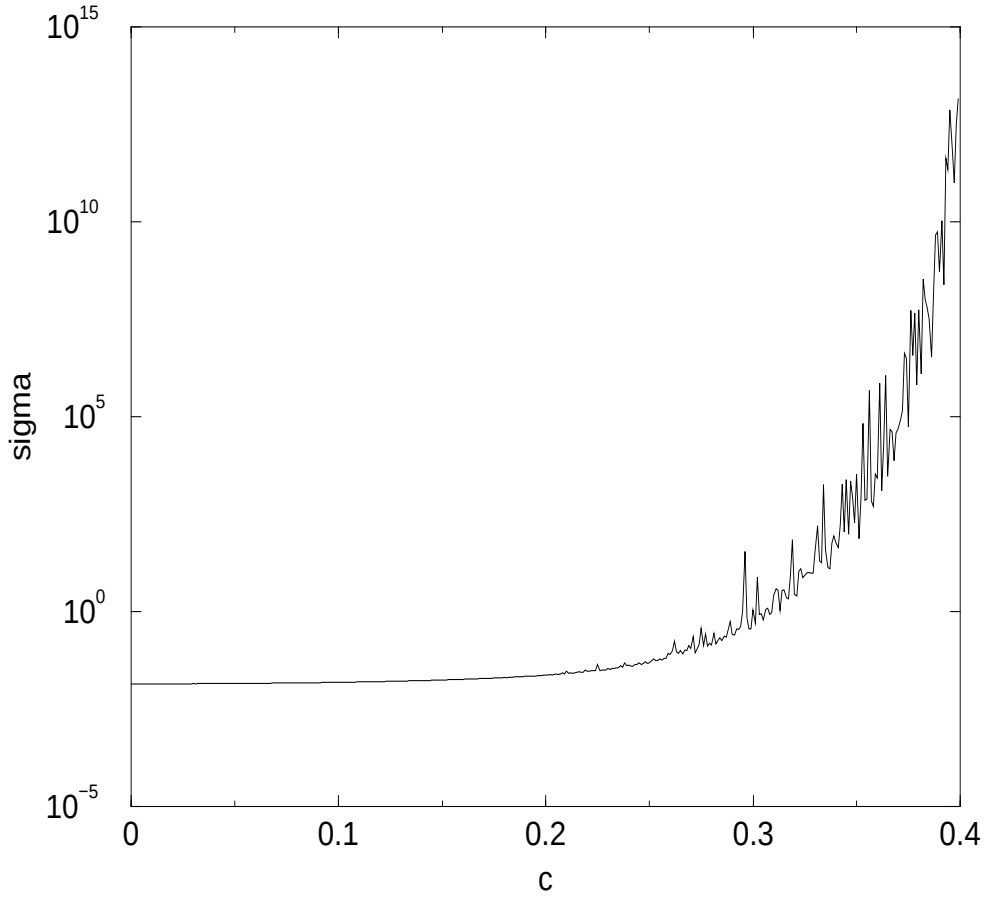

図 27: c に対する σ の変化。c の大きいところで σ が急激に増加する様子がわかる。

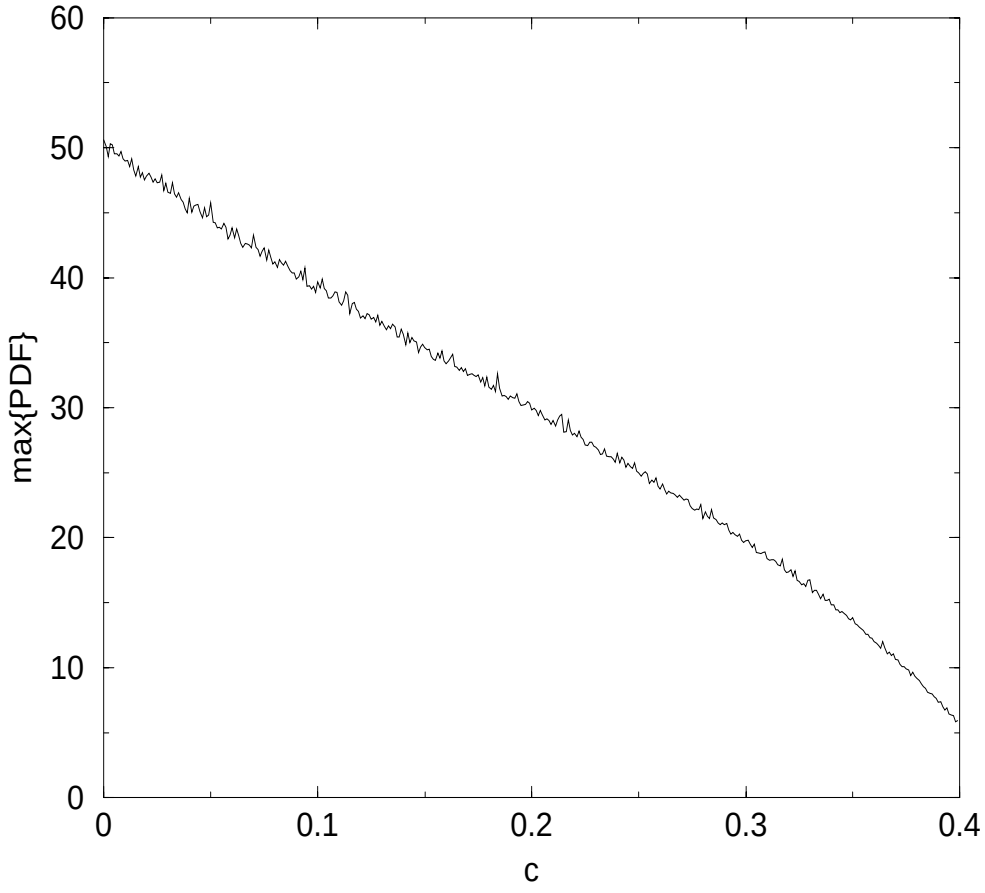

図 28: c に対する $\max_{\Delta p} Z(\Delta p)$ の変化。$\max_{\Delta p} Z(\Delta p)$ が c の増大に伴い減少してゆく様子がわかる。

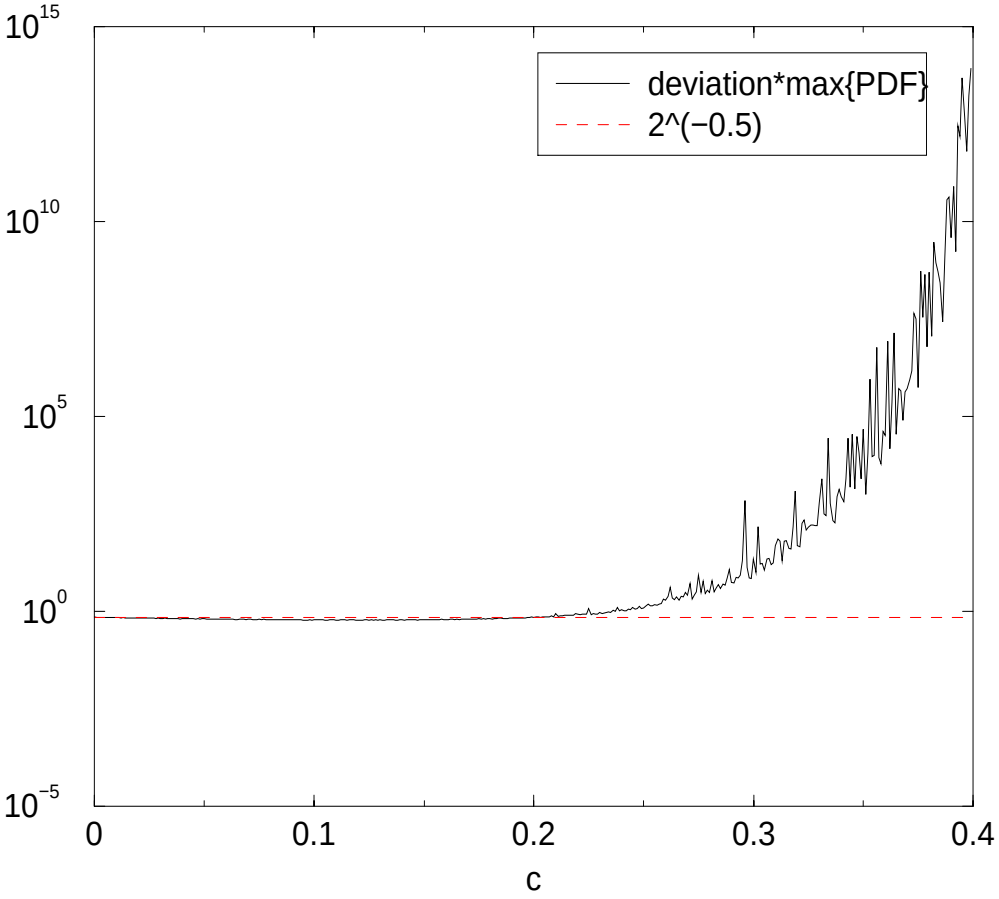

図 29: c を変化させたときの $\theta(c)$（実線）と、$\frac{1}{\sqrt{2}}$（点線）。c が大きくなると $\theta(c)$ が $\frac{1}{\sqrt{2}}$ から急激に離れ、増大する様子がわかる。

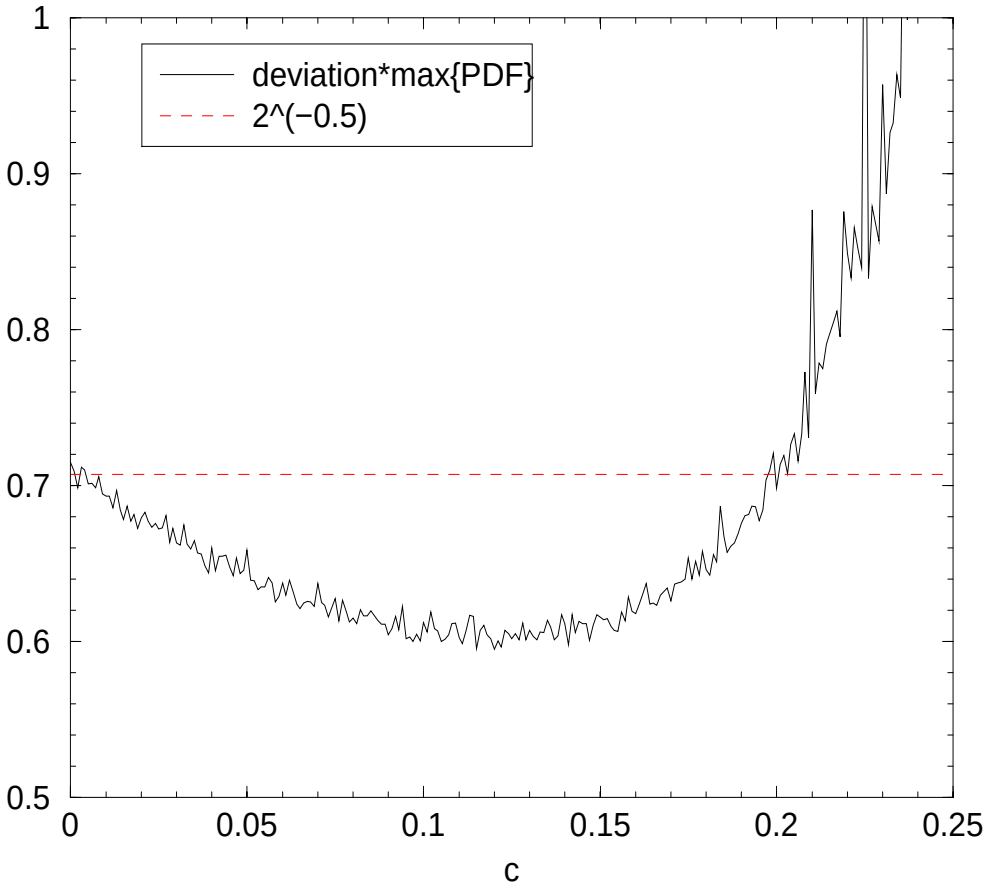

図 30: 図 29 の拡大図。$\theta(c)$ が c の小さいところでは $\max_{\Delta p} Z(\Delta p)$ の減少の効果で減少し、c が大きくなると σ の増大の効果で増大してゆくことがわかる。

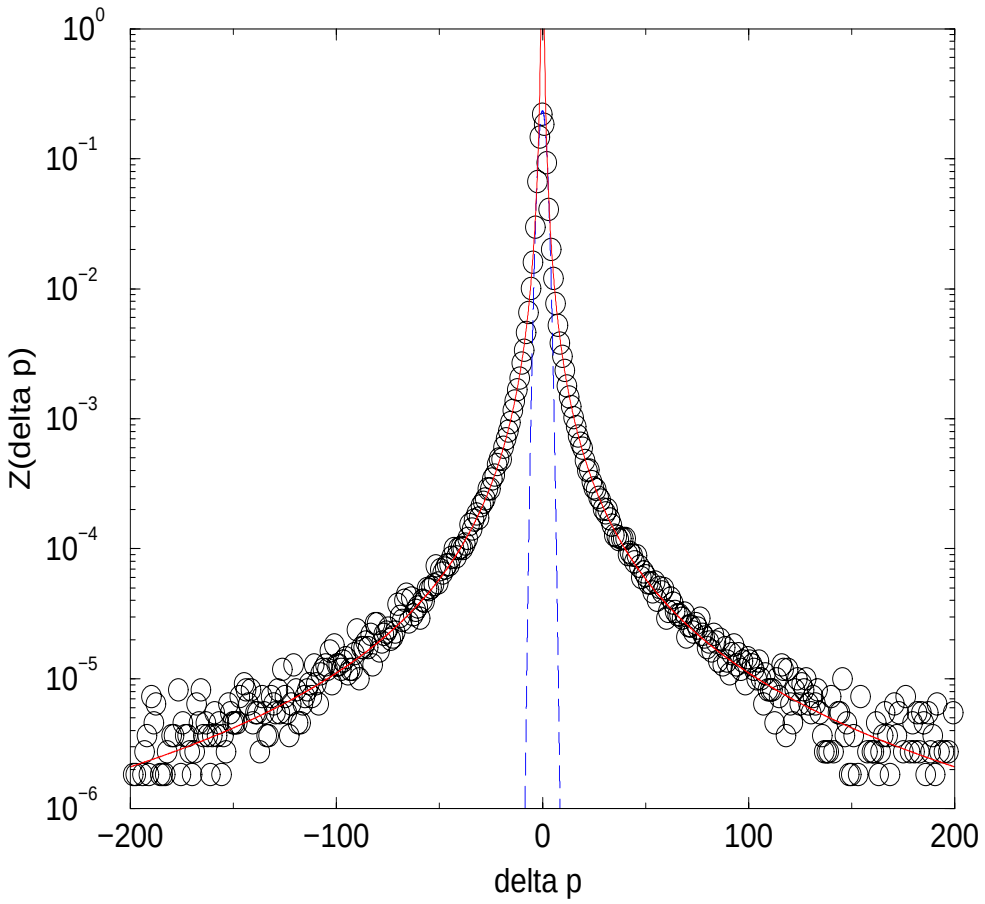

図 31: 式（12）において、ラプラス分布 $U(\phi)$ を正規分布 $u(\phi) = \frac{1}{\sqrt{2\pi}}\exp\left(\frac{-\phi^2}{2}\right)$ で置き換えたときの Δp の確率分布（◯印）。実線は $\beta = 1.4$ のフラクタル分布。点線は平均と分散を合せた正規分布。$U(\phi)$ を正規分で布置き換えても価格差はフラクタル分布で近似できることがわかる。

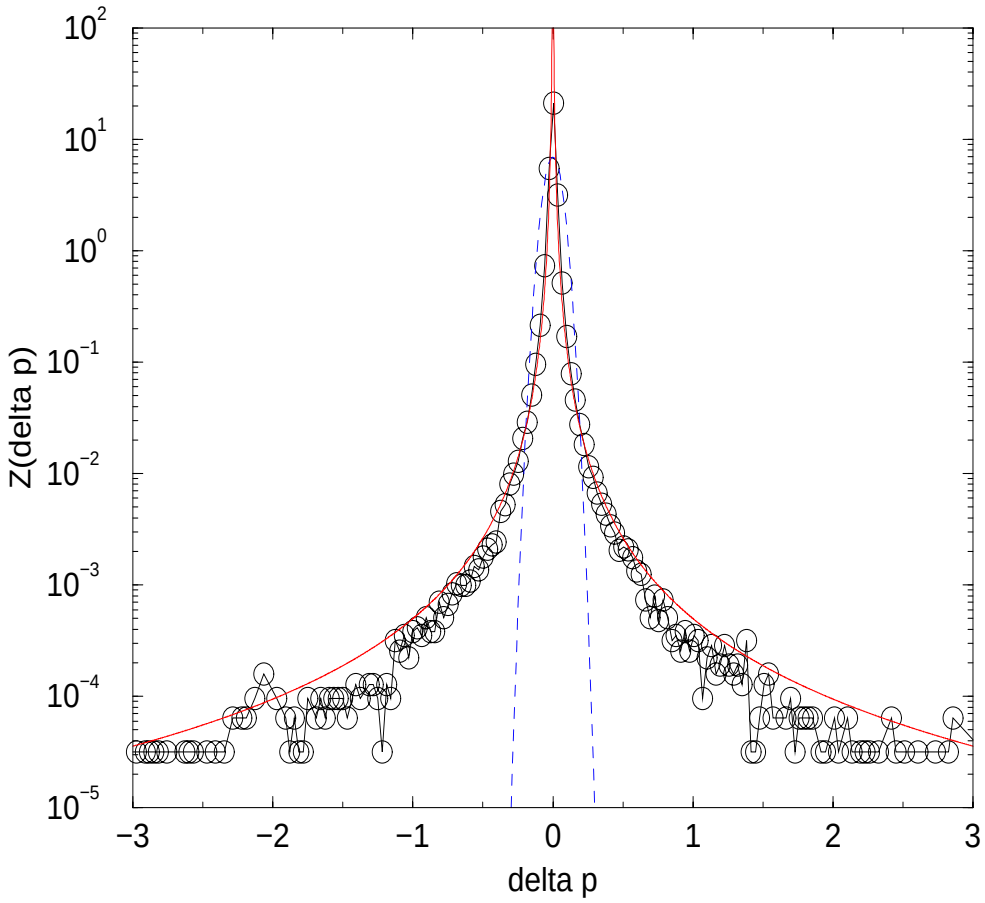

図 32: 式（12）において、離散的指数分布 $W(n)$ を連続的な指数分布 $w(n) = \dfrac{1-e^{-\gamma}}{e^{-\gamma}} \exp(-\gamma n)$ で置き換えたときの Δp の確率分布（◯印）。実線は $\beta = 1.4$ のフラクタル分布。点線は平均と分散を合せた正規分布。$W(n)$ を連続的な指数分布に置き換えても価格差はフラクタル分布で近似できることがわかる。

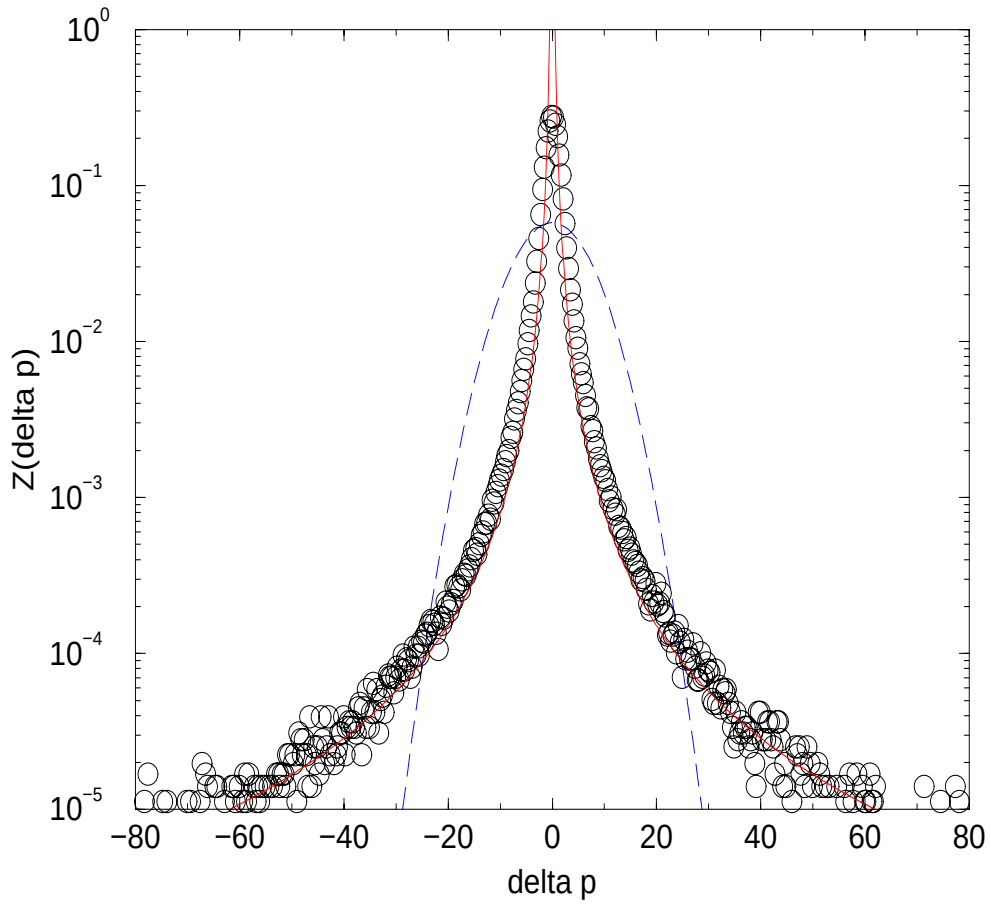

図 33: 式（12）において、ラプラス分布 $U(\phi)$ を正規分布 $u(\phi) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-\phi^2}{2}\right)$ で、離散的指数分布 $W(n)$ を連続的な指数分布 $w(n) = \frac{1-e^{-\gamma}}{e^{-\gamma}} \exp(-\gamma n)$ で置き換えたときの Δp の確率分布（◯印）。実線は $\beta = 1.4$ のフラクタル分布。点線は平均と分散を合せた正規分布。$U(\phi)$ を正規分布で、$W(n)$ を連続的な指数布に置き換えても価格差はフラクタル分布で近似できることがわかる。

6　謝辞

一年間を通して様々な方面でお世話になった羽田野直道先生に心から感謝します。

また、私の質問に快くこたえて頂いたソニーコンピュータサイエンス研究所の高安秀樹先生、東北大学大学院情報科学研究科の佐藤彰宏氏に御礼申し上げます。

参考文献

[1] B.B. Mandelbrot, 日経サイエンス 5 月号 (1999).

[2] R.N. Mantegna, H.E. Stanley, Nature 376(1995), p.46.

[3] H. Takayasu, A.-H. Sato and M. Takayasu, Phys. Rev. Lett. 79(1997)966.

[4] V.A. fock, Bull. Acad. Sci. USSR Ser. Phys. 14(1950)70.

[5] S.B. Pope, Energy Combust. Sci. 11(1985)119.

[6] V. Eswaran, S.B. Pope, Phys. Fluids 31(1988)506.

[7] S.Q. Zhu, Phys. Rev. A 41(1990)1689.

[8] S.Q. Zhu, Phys. Rev. A 45(1992)8148.

[9] A.-H Sato, H. Takayasu, Physica A 250(1998)231.

A プログラムリスト

A.1 決定論モデル

A.1.1 価格 $P(t)$、$\Delta P(t)$ 生成ルーチン

```
#include<stdio.h>
#include<stdlib.h>
#define N_TRADER 100
#define t_max 100000
float abs_r(float x);
main(int argc, char *argv[])
{
  int B=0, a=1, t, i, imax, imin;
  float lambda=1.0, c, alfa=0.01,
    Pprev = 0, P = 0, delta_P = 0;
  float D[N_TRADER][2], BB[N_TRADER],
    L, Bmax, Bmin, amax, amin, ran;
  FILE *fp;
  if ( argc != 2) {
    printf("   Write the data file's name.\n");
    exit(1);
  }
  if ((fp = fopen(argv[1], "w")) == NULL) {
    printf("   The file dosen't open.\n");
    exit(1);
  }
  printf("     Input The Parameters (default) <
         N_TRADER=%d , t_max=%d >\n", N_TRADER,
         t_max);
  printf("     c    =(0.0 - 0.3) "); scanf("%f",&c);
  for( i = 0; i < N_TRADER; i++){
    ran   = (2.0 * (float)rand()/RAND_MAX )- 1.0;
    D[i][B] = ran * lambda / 2.0;
    D[i][a] = ran * alfa;
  }
  for( t=0 ; t <= t_max ; t++ ){
    imax = 0;
    imin = 0;
    for( i=1 ; i < N_TRADER ; i++ ){
      if( D[i][B] < D[imin][B] ){ imin = i; }
      if( D[i][B] > D[imax][B] ){ imax = i; }
    }
    L = D[imax][B] - D[imin][B];
    if( L >= lambda ){
      P = ( D[imax][B] + D[imin][B] + lambda )/ 2.;
      delta_P = P - Pprev;
      Pprev = P;
```

```
      D[imin][a] =  abs_r( D[imin][a] );
      D[imax][a] =  -abs_r( D[imax][a] );
    }else{
      P = Pprev;
    }
    for( i=0 ; i < N_TRADER ; i++ ){
      D[i][B] += D[i][a] + c * delta_P;
    }
    fprintf(fp,"%26.16e%26.16e\n" , P, delta_P);
  }
  fclose(fp);
}
float
abs_r(float x)
{
  if( x < 0. ){
    x = -x;
  }
  return x;
}
```

A.2 確率モデル

A.2.1 価格 p_s 生成ルーチン

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define T_max 10000
#define Gamma 0.4
#define Sigma 0.01
double Laplace_0();
int Risan_Teki_Sisuu();
double delta_P(double *c, double *dP_beforep,
  double *dP_afterp);
main(int argc, char *argv[])
{
  int t;
  double e, P = 0.0, dP_before = 0.0, dP_after = 0.0;
  FILE *fp;

  if ( argc != 2 ){
    printf("   Write the data file's name.\n");
    exit(1);
  }
  if ((fp = fopen(argv[1], "w")) == NULL) {
    printf("   The file dosen't open.\n");
```

```
    exit(1);
  }
  printf("input c\n?")  ;scanf("%lf",&e);
  for( t = 0; t < T_max; t++ ){
    P += delta_P( &e, &dP_before, &dP_after);
    fprintf(fp,"%25.16e\n", P);
  }
  fclose(fp);
}
```

A.2.2 価格差 Δp_s 生成ルーチン

```
double
delta_P(double *cp, double *dP_beforep,
        double *dP_afterp)
{
  *dP_afterp = (*cp) * (double)Risan_Teki_Sisuu() *
    (*dP_beforep) + Laplace_0();
  *dP_beforep = *dP_afterp;
  return *dP_afterp;
}
```

A.2.3 離散的指数分布に従う乱数 n_s 生成ルーチン

```
int
Risan_Teki_Sisuu()
{
  double RTS;
  RTS = -( 1.0 / Gamma )* log( 1.0 - sprng() );
  return (int)RTS + 1;
}
```

A.2.4 ラプラス分布に従う乱数 ϕ_s 生成ルーチン

```
double
Laplace_0()
{
  double ran;
  ran =sprng()  ;
  if( ran < 0.5){
    return ( double) Sigma *  log( 2.0 * ran ) ;
  }else {
    if( ran >= 0.5 ){
      return ( double) -Sigma *
        log( 2.0 * ( 1.0 - ran) );
```

```
    }
  }
}
```

A.3 その他のプログラム

A.3.1 ヒストグラム作成ルーチン

```
#include<stdio.h>
#define N_data 1000000
#define histg_range 10000
main(int argc, char *argv[])
{
  FILE *fp_read,*fp_write;
  static int N_limit=N_data,histg[histg_range],
    i, j, k, l, m, n;
  double data[N_data],d_min,d_max,delta_d, myu, vari;
  if ( argc != 3 ) {
    printf("Write the r_data and the w_data
           file's name.\n");
    exit(1);
  }
  if((fp_read = fopen(argv[1], "r")) == NULL ||
     (fp_write = fopen(argv[2], "w")) == NULL){
    printf("   The file dosen't open.\n");
    exit(1);
  }
  for( i = 0 ; i < histg_range ; i++ ){
    histg[i]=0;
  }
  for( j = 0 ; j < N_data ; j++ ){
    data[j]=0.;
  }
  for( k = 0 ; k < N_data ; k++ ){
    if(fscanf( fp_read ,"%lf*[^\n]",&data[k]) == EOF ){
      N_limit = k;
      break;
    }
  }
  d_min = data[0];
  d_max = data[0];
  for( l = 0 ; l < N_limit ; l++ ){
    if( data[l] < d_min ){
      d_min = data[l];
    }
  }
  for( l = 0 ; l < N_limit ; l++ ){
```

```
    if( data[l] > d_max ){
      d_max = data[l];
    }
  }
  delta_d = (double)( d_max - d_min )/histg_range;
  for( m = 0 ; m < N_limit ; m++ ){
    histg[(int)(( data[m] - d_min )/ delta_d)]++ ;
  }
  for( i = 1 ; i <= N_limit ; i++ ){
    myu += ( data[i] - myu )/ (double)i;
    vari += ( (data[i] * data[i]) - vari )/i;
  }
  printf( " moment1=%26.16e \n moment2=%26.16e \n
         N_limit %d\n", myu, vari, N_limit);
  fprintf( fp_write, "#moment1=%26.16e \n
          #moment2=%26.16e \n#N_limit %d\n",
          myu, vari, N_limit);
  for( n = 1 ; n < histg_range ; n++ ){
    if( histg[n] != 0 ){
    fprintf( fp_write,"%26.16e %26.16e\n",
      d_min + delta_d *(n + 0.5),
      (double)histg[n]/( N_limit * delta_d ));
    }
  }
  fclose(fp_read);
  fclose(fp_write);
}
```

A.3.2 σ と $Z(\Delta p)$ の最大値を出力するルーチン

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define Gamma 0.4
#define Sigma 0.01
#define hist_range 1000
double Laplace_0();
int Risan_Teki_Sisuu();
double abs_r(double x);
main(int argc, char *argv[])
{
  int t=0, T_max ,i = 0, h = 0,
    hist[hist_range],hist_max;
  double delta_P_before, delta_P_after,
    multi, delta_x, dx,
```

```
  vari, c, c_min = 0.0,
  c_max = 0.4, delta_c = 0.001, myu, lap,
  m_vari, prob[hist_range], prob_max, m_devi;
FILE *fp;
if ( argc != 2 ) {
  printf("Write  the w_data  file's name.\n");
  exit(1);
}
if((fp = fopen(argv[1], "w")) == NULL){
  printf("   The file dosen't open.\n");
  exit(1);
}
printf("input T_max\n?")      ;scanf("%d",&T_max);
printf( "#%7.6s%26.2e\n","T_max=",(double)T_max);
fprintf(fp, "#%7.6s%26.2e\n","T_max=",(double)T_max);
printf( "#%7.1s%26.16s%26.16s%26.16s%26.16s\n",
        "c","sqrt(vari-m^2)",
        "max_PDF","multi","1.0 / sqrt( 2.0 )");
printf( "#%7.1s%26.16s%26.16s%26.16s%26.16s\n",
        "-","--------------",
        "-------","-----","-----------------");
fprintf(fp, "#%7.1s%26.16s%26.16s%26.16s%26.16s\n",
        "c","sqrt(vari-m^2)",
        "max_PDF","multi","1.0 / sqrt( 2.0 )");
fprintf(fp, "#%7.1s%26.16s%26.16s%26.16s%26.16s\n",
        "-","--------------",
        "-------","-----","-----------------");
delta_x = 0.1;
dx = delta_x /(double)hist_range;
for( c = c_min; c <= c_max; c += delta_c){
  for( h = 0; h < hist_range; h++ ){
    hist[h] = 0;
    prob[h] = 0.0;
  }
  delta_P_after = 0.0;
  delta_P_before = 0.0;
  vari = 0.0;
  myu = 0.0;
  for( t = 1; t <= T_max; t++ ){
    delta_P_after = c * (double)Risan_Teki_Sisuu()
      * delta_P_before + Laplace_0();
    myu += ( delta_P_after - myu ) / t;
    vari += (( delta_P_after * delta_P_after )
      - vari ) / t;
    delta_P_before = delta_P_after;

    if( abs_r( delta_P_after ) < ( delta_x / 2.0)){
      hist[(int)(( delta_P_after -
        ( - delta_x / 2.0 ) )/ dx)] ++;
```

```
      }
    }
    for( h = 0; h < hist_range; h++ ){
      prob[h] = (double)hist[h]/( (double)T_max * dx);
    }
    prob_max = -1;
    for( h = 0; h < hist_range; h++ ){
      if( prob[h] > prob_max ){
        prob_max = prob[h];
      }
    }
    m_vari = vari - ( myu * myu );
    m_devi = sqrt( m_vari );
    multi = m_devi * prob_max;
    lap = 1.0 / sqrt( 2.0 );
    printf( "%8.2e%26.16e%26.16e%26.16e%26.16e\n",
            c , m_devi, prob_max, multi, lap);
    fprintf(fp, "%8.2e%26.16e%26.16e%26.16e%26.16e\n",
            c, m_devi, prob_max, multi, lap);
  }
  fclose(fp);
}
```

A.3.3 フラクタル分布作成ルーチン

```
#include<stdio.h>
#include<math.h>
#define X_range 20.
double abs_r(double x);
double levy(double x, double beta);
/**** return   f(x) = a * | x - myu |^( -beta) ****/
main(int argc, char *argv[])
{
  FILE *fp;
  double a, beta, myu, delta_X, X, Y;

  if( (fp = fopen( argv[1] , "w" )) == NULL ){
    printf("error : can't open file\n");
    exit(1);
  }
  printf( " f(x) = a * x^( -beta) \n");
  printf( "input beta\n?")      ;scanf( "%lf",&beta);
  printf( "input a\n?")         ;scanf( "%lf",&a);
  printf( "input myu\n?")       ;scanf( "%lf",&myu);
  delta_X = (X_range )/10000.;
  for( X= (-X_range + myu ) ;
       X<= (X_range + myu ) ; X+=delta_X ){
```

```
    if( (X - myu ) != 0.0 ){
      Y = a * levy((X - myu) ,beta);
      if( Y >= 0.0 ){
        fprintf( fp, "%26.16e%26.16e\n", X, Y);
      }
    }
  }
  fclose(fp);
}
double
levy(double x, double beta)
{
  double F;
  F = -beta * log( abs_r( x ) );
  return exp( F );
}
```