〔非公開〕

ＴＲ－Ｃ－００６８

格子点探索法における最近傍底点の周期性

松　田　　光　司

Koji　Matsuda

１９９１．９．１９

ＡＴＲ通信システム研究所

格子点探索法における最近傍底点の周期性

ＡＴＲ通信システム研究所　通信ソフトウェア研究室

松田　光司

目次

１．まえがき

龍谷大学では３回生の必修科目として企業や研究所等で学外実習を行なっているが、この度筆者は1991年8月19日～9月13日までＡＴＲ通信システム研究所（以下ＡＴＲという）においてこの学外実習を行なった。実習テーマは、格子点探索法と言われる素因数分解アルゴリズムの高速化に関するものである。

ＡＴＲでは従来より大きな値の合成数を素因数に高速分解する問題（素因数分解問題）の研究を行っている。この素因数分解問題は、公開鍵方式の一種であるＲＳＡ暗号に必要な鍵の長さの評価を行う為研究されており、鍵がどのくらい長ければ暗号は安全であるかという問題は、個々の素因数アルゴリズムに依存している。

ＡＴＲでも計算量の少ない新しい方式として、格子点探索法と呼ばれるアルゴリズムを提案している。しかしこの格子点探索法は現段階の基本的な方式では、世界レベルの計算量に追いつく事が出来なかった。そこでこの格子点探索法の計算量を更に減らす為に、最近傍底点と言う概念を新たに導入し、よりいっそうのアルゴリズムの高速化を計っている。

今回の実習では格子点探索法の解の探索を行う過程において、この最近傍底点が、周期性を持つかどうか確認する為の計算機実験を行い、更にそれをもとにした解析を行う。そして将来最近傍底点の概念を用いた格子点探索法のアルゴリズムを確立する為に基礎データを得る事を目的としている。

２．暗号

ａ）暗号

重要な情報を第三者に知られないように送る秘密通信のことである。

実際の暗号通信システムは、複雑であるがその本質的なモデルを考えると、図Ａのようになる。もとの通信文は平文といわれ、その意味内容が第三者に分かりにくい形に変換されたものが暗号文といい、逆に、暗号文をもとの平文に復元する過程を復号化という。

これらの変換を定めるものは、アルゴリズムとそのパラメータとなる鍵である。そのアルゴリズムで定まるものを、暗号方式という。

そして、復号化アルゴリズムは、暗号化アルゴリズムの逆変換である。また、暗号化鍵と復号化鍵とは同一であることが多いが、異なった鍵を用いることもある。

図Ａに示す用に、暗号文は、盗聴されることを前提としている。盗聴者は、アルゴリズムや鍵の情報をもたずにもとの平文や鍵を見破ろうとする。これが暗号解読である。

近年の暗号の飛躍的な進歩は、以下の四つが原因となっている。

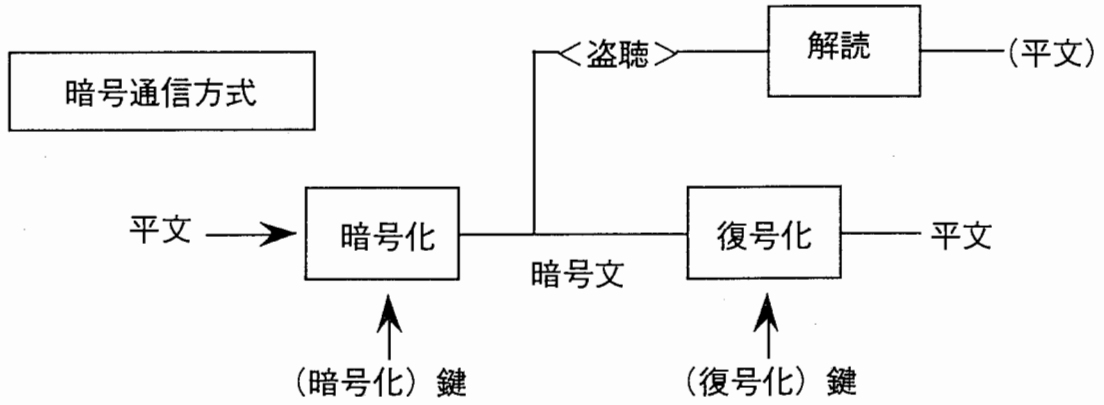
（１）アクセス制御，機密保護，改ざん，偽造防止等の情報セキュリティ面からのニーズ。

（２）エレクトロニクスの進歩により、複雑な暗号化操作が簡単に出来るようになった。

（３）標準暗号ＤＥＳ（データ暗号規格）の出現。

（４）公開暗号方式の発明。

(1)(2)(3)については、本報告書に直接関係がないので、専門書（参考文献(3)）に譲るとして、(4)については以下ｂ）で説明する。



図Ａ　暗号通信システムのモデル

ｂ）公開鍵暗号方式

図Ａの暗号方式において、（復号化）鍵は秘密に保たれるが（暗号化）鍵は公開される場合、これを公開鍵暗号方式と呼ぶ。通常２つの鍵は、受け手が作り、公開鍵である（暗号化）鍵を公開ファイルに登録する。これは、例えると電話帳の様なものと思えばよい。暗号通信を行ないたい送り手は、公開ファイルで受けての公開鍵（暗号化鍵）をひき、それによって暗号化して受け手に送る。受け手は自分の秘密鍵（復号化鍵）でそれを復号して平文を得る。

そして、秘密を保つ為には、公開鍵から暗号鍵を導き出すのを実際上不可能にしなければならない。

また、次に示すＲＳＡ暗号が、公開鍵暗号方式の実現法の一つである。

ｃ）ＲＳＡ暗号

平文は符号化された自然数列になっているものとする。２つの素数ｎ，ｍをとり、Ｎ＝ｎ・ｍとおく。平文を表わす自然数列を適当に区切って、Ｎより小さい整数Ｍの列とみなす。

（ｎ－１）（ｍ－１）と互いに素な数ｒ

ｄ・ｒ≡１（ｍｏｄ（ｎ－１）（ｍ－１））なる数ｄ

をとる。

暗号化の為の公開鍵はＮとｒであり、復号の為の秘密鍵はｄである。

暗号の手続き　Ｅ：Ｍ→Ｃ　は、

$C \equiv M^r \pmod N$

なるＣを作ることである。

復号手続き　Ｄ：Ｃ→Ｍ　は、

$M \equiv C^d \pmod N$

によってＭを得ることである。

Ｅｕｌｅｒの定理やＦｅｒｍａｔの小定理を用いれば、比較的容易に、

Ｄ（Ｅ（Ｍ））＝Ｍ，　Ｅ（Ｄ（Ｍ））＝Ｍ

が得られる。

また、Ｅは、高々$2 \cdot \log_2 r$回、Ｄも高々$2 \cdot \log_2 d$回の乗算で手続きが完了

する。

盗聴した解読者は公開の暗号化鍵Ｎとｒのよって、原理的には、次のようにして復号鍵ｄを求めることができる。

（１）Ｎを素因数分解し、Ｎ＝ｎ・ｍなるｎ，ｍを求める。

（２）ｕ・ｒ≡１（ｍｏｄ ｎ－１），ｖ・ｒ≡１（ｍｏｄ ｍ－１）なるｕ，ｖを求める。

（３）(2)からｕ－ｖはｎ－１，ｍ－１の最大公約数の倍数であるから

ｕ－ｖ＝ｋ（ｎ－１）＋ｌ（ｍ－１）

なるｋ，ｌが存在する。これを求める

（４）ｄ＝ｕ－ｋ（ｎ－１）（＝ｖ＋ｌ（ｍ－１））とおけば、これが復号鍵ｄである。

しかし、上述のアルゴリズムには、(1)の素因数分解が含まれている。この合成数を２つの素因数に高速に分解するのは非常に難しく。例えば、ｎ，ｍを各々１００桁程度にとっておけば、Ｎは２００桁で、現在知られている最も速いアルゴリズムをもってしても、１．２×10^{23}回程度の演算を要する。１つの演算を10^{-6}秒行なっても３．８×10^{9}年かかり、実際上は解読不可能といわれている。

このようにＲＳＡ暗号の解読の難しさの根拠は、大きな合成数を素因数分解することの難しさにある。

３．素因数分解

ａ）素因数分解

素因数分解とは、整数N(>1)が与えられた時、Ｎ＝ｎｍとなる様な１より大きい２整数ｎ，ｍを求めることである。

ｂ）試行割算法

素因数分解のアルゴリズムで最も単純かつ明瞭な方法は、合成数Ｎを素数で順に割る、という試行割算法である。但し、この方法は最悪$N^{1/2}$回の探索となり、１０進２００桁程度の合成数を用いる現在のＲＳＡ暗号は、２のｂ）で触れた様に解読不可能である。

ｃ）現在の世界レベル

素因数分解の現在の世界レベルの計算量は、Ｏ（$N^{1/8}$）である。この様な計算量で素因数分解できるアルゴリズムには、楕円曲線法や２次ふるい法がある。（詳しくは、参考文献(5)参照）

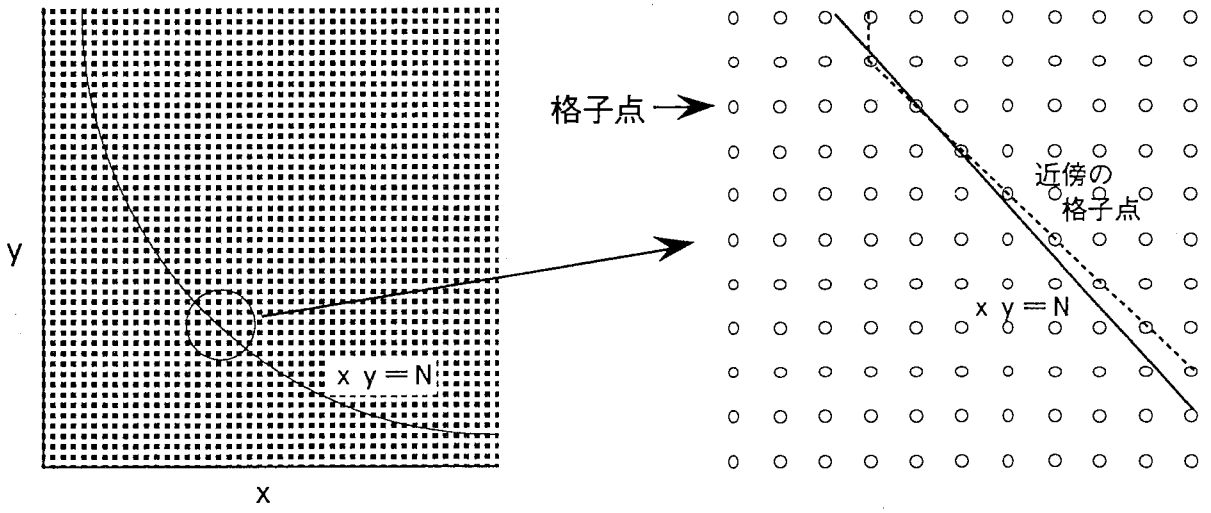
４．格子点探索法

ａ）格子点探索法

ＡＴＲでは、素因数分解を行うアルゴリズムとして、格子点探索法の研究を行っている。素因数分解は幾何学的には図Ｂに示すようにｘ，ｙを座標軸とする解平面上で、双曲線ｘｙ＝Ｎが整数座標値からなる格子点を通過する座標を求めることに相当する。そ

こで格子点探索法では、双曲線ｘｙ＝Ｎの近傍に位置するような格子点だけを探索する。見つかった格子点について、座標値の積を計算して、合成数に一致すれば、それが因数である。この方式では最大、$N^{1/4}$の間隔で因数探索を進めることが出来る。

しかし、このままの方式では、計算量は世界レベルのＯ（$N^{1/8}$）には追いついていない。



図Ｂ　解曲線近傍の格子点

ｂ）最近傍底点

そこで、計算量のよりいっそうの低減を計る為に、最近傍底点という概念を導入する。この最近傍底点とは、探索毎に得られる底点のうち最も双曲線に近いものをさす。

合成数の値が、非常に大きい場合（10^{200}程度）、双曲線は局部的には直線であると見なす事が出来ると考えられ、この時双曲線と底点との距離は図Ｃに示すように一定の割合で減少（または増加）し、これは双曲線と底点との距離が格子点の間隔以上になる（つまり底点が次の探索格子点に移る）迄続く。更にこの性質は一定の周期を持って繰り返されるのではないかと想像する事が出来る。

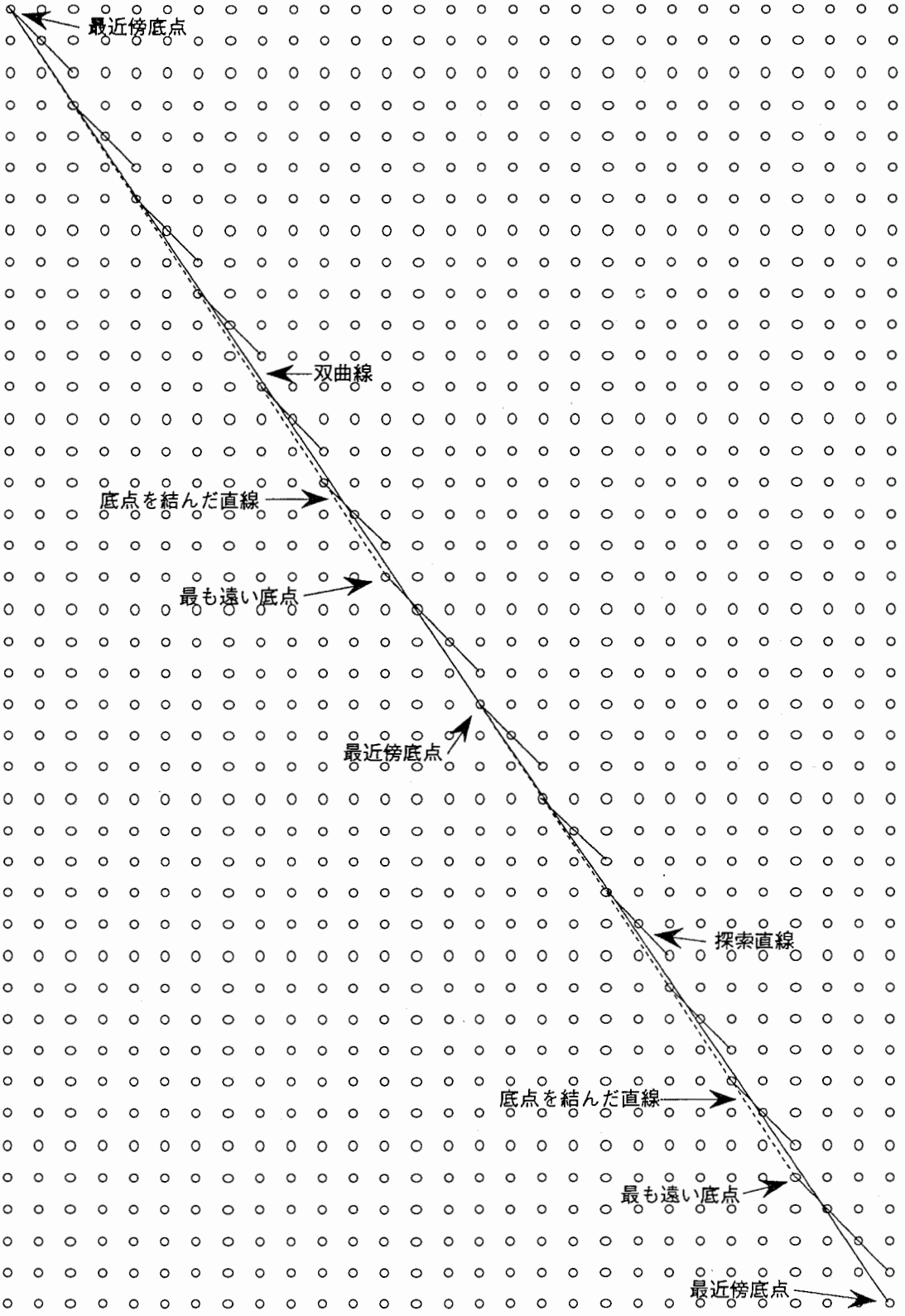
もしそうであるならば、周期的に出現する最近傍底点の部分のみ探索を行うことによって少ない探索回数で双曲線上の格子点の発見（つまり素因数分解）を行う事が出来る。

今回は底点が本当に周期性を持つかを計算機実験により確認し、そして最近傍底点の概念の実現性についての証明を行う。

５．提供プログラムの評価

ＡＴＲでは、最近傍底点とは別に剰余類を用いた格子点探索法の研究が行われている（参考文献(4)参照）。その研究の為に作成されたプログラムを基に今回の実験は行われた。

提供されたプログラムは、合成数を２個の因数の積に分解する際に、複数の剰余類を用いることにより因数の候補を絞り込むという因数分解の高速化の手法を評価する為のものである。


最近傍底点
双曲線
底点を結んだ直線
最も遠い底点
最近傍底点
探索直線
底点を結んだ直線
最も遠い底点
最近傍底点

図C　最近傍底点の図解

a）実行ファイル

ｄｎｐｄｄｅｃ

＊機能

合成数N、素数P、合成数と因数のPに関する剰余ｈ，ｊ，ｋを与えると、ｍ／ｎ＝ｚからN＝ｎｍとなるｉ ｍｏｄ Ｐ型のｎ、ｊ ｍｏｄ Ｐ型のｍを順方向に探索する。

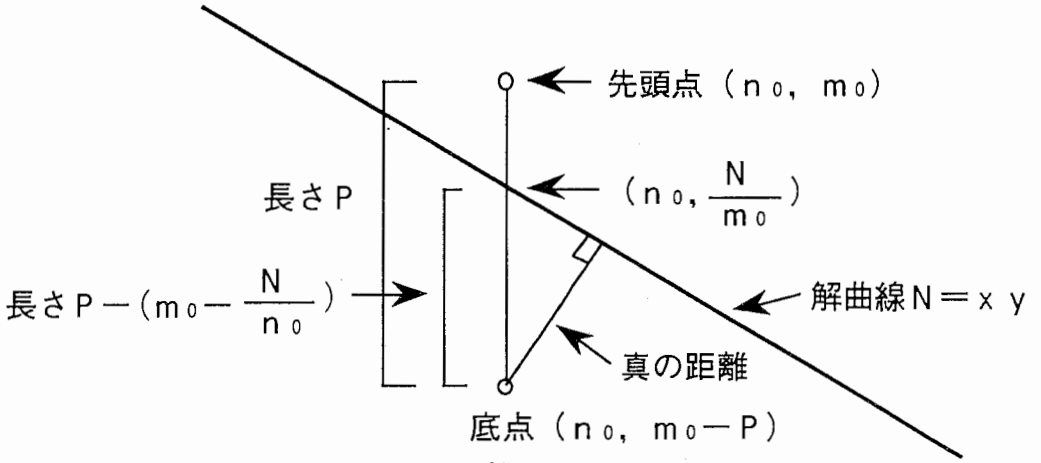
＊入力

N	合成数
P	素数
N__ｍｏｄ	被分解合成数NのPに関する剰余
ｎ__ｍｏｄ	探索する因数ｎのPに関する剰余
ｍ__ｍｏｄ	探索する因数ｍのPに関する剰余
ｚ	探索開始点のｍ／ｎの初期値（１，２，３・・・・）

＊出力

ｔ	探索回数
ｎ	探索先頭点のｎ
ｍ	探索先頭点のｍ
ｂｎ＋ｍ	探索区間のおけるβｎ＋ｍの値（共通因子判定用）
Ｐ－（ｍ－Ｎ／ｎ）	底点と解曲線の距離に比例する値
ｋ	探索速度指数（Ｐｋだけ探索が進む）

何故、上述のＰ－（ｍ－Ｎ／ｎ）の値が底辺との距離に比例するのかは、アルゴリズムと深い関係がある。



図Ｄ　距離とP-(m-N/m)

上図における直角三角形の斜辺と底辺は、直角三角形の性質より比例する。よって、P-(m-N/n)と真の距離は、比例する。

＊起動方法

ａ．マニュアルモード（表示に従ってパラメータを入力する）

ｄｎｐｄｄｅｃ

ｂ．オートモード

ｄｎｐｄｄｅｃ　＜入力パラメータファイル

b）プログラムの評価

合成数Nが１９９桁まで入力できるが計算において、出力結果が小数点１０桁は、表示されない。

・小数点１０桁以下の必要性

このプログラムは、桁数を大きくすると（Ｐ－（ｍ－Ｎ／ｎ））が非常に小さくなり、１０^{-10}以下なると「０」のみを表示して（Ｐ－（ｍ－Ｎ／ｎ））がどの様な値になっているかわからない。よって、大きな桁を扱う時は、改良の必要がある。

また、仮に上述の改良がなされたとしても、非常に大きいな桁の合成数を扱う場合、実行速度が遅い為計算し終えるのに何年も掛かってしまう為、実用的ではない。より高速化の必要がある。

なお、この評価は、ワークステーションの環境下で行われた。

６．実験方法・結果・考察

６－０　グラフの見方

この実験に於けるグラフは、横方向にかなり圧縮されている為、横方向の状況は、余り信憑性がない。だが、分布状況をおおまかに調べるのには、問題はない。

６－１　探索回数と距離の関係

a）実験１の方法

いろいろな合成数について、双曲線と底点との距離を計算しグラフ化する。データ作成は、npddec.cを改良したt_distance.c（プログラム参照）で行なった。アルゴリズムは、図Ｅ参照。

桁数	合成数（素因数＊素因数）	素数発見	距離データ
４桁	5069=137＊37	16回目	out4keta.dat
６桁	636913=137＊4649	311回目	out6keta.dat
８桁	76659109=3541＊21649	2272回目	out8keta.dat
１０桁	9000909991=9901＊909091	38768回目	out10keta.dat
１２桁	410923533521=69857＊5882353		out12keta.dat

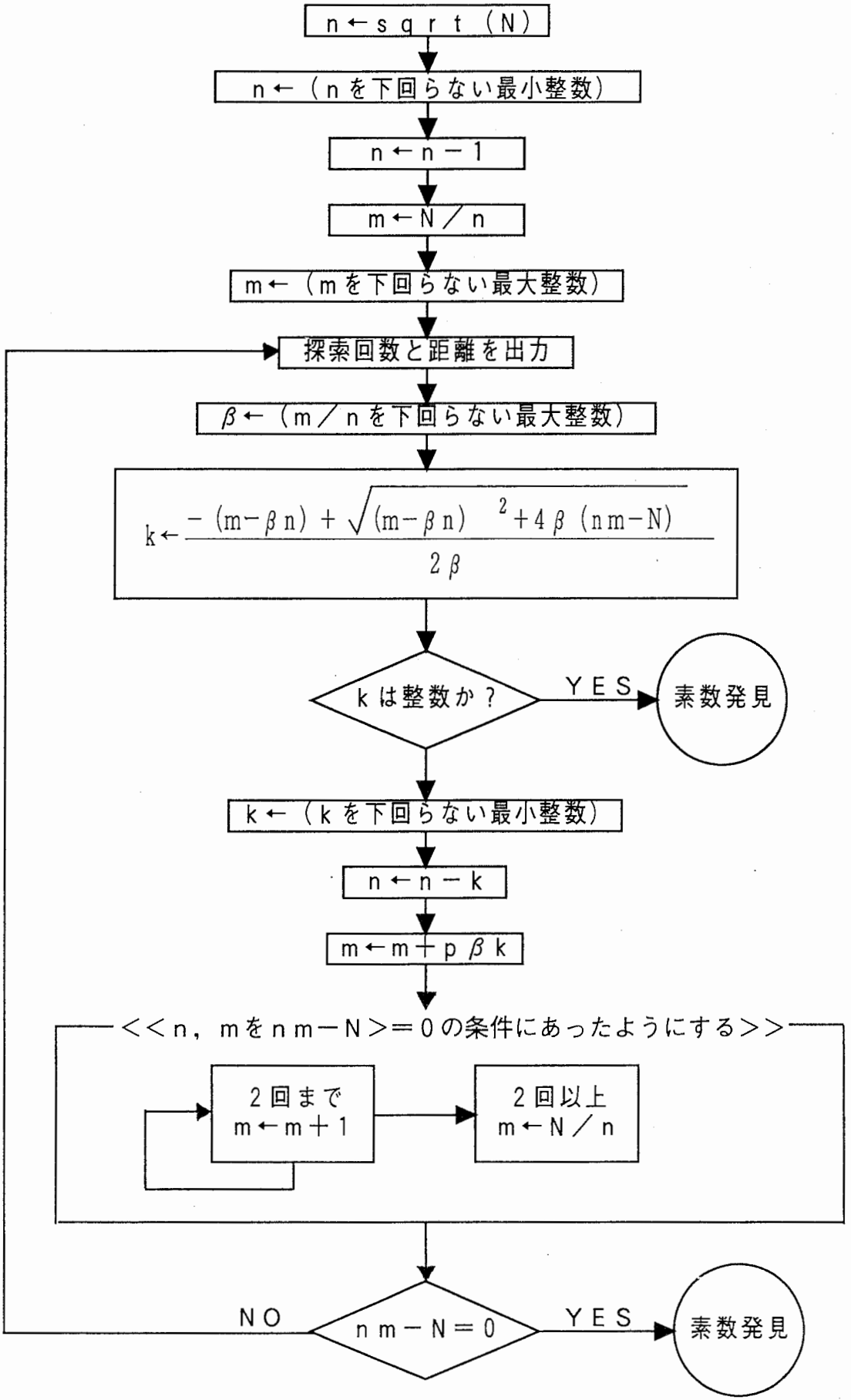
表１　各合成数におけるグラフのデータ

そして、横軸に探索回数、縦軸に距離を取り作成されたデータをグラフ化する。

b）実験１の結果

グラフは、以下の通り

図１－１　　４桁
図１－２　　６桁
図１－３　　８桁
図１－４　１０桁
図１－５　１２桁



図E　t_distance.cのアルゴリズム

余りに多くのデータを扱った為、折れ線グラフで表示することは、困難となった（線で真っ黒になる）為、点だけのグラフにした。

c）実験１の考察

各点グラフの分布は、桁が大きくなればなる程ある種の幾何学的図形に定まっていく。

・分布状況

（１）距離に於ける分布の範囲が初め小さく徐々に大きくなっていく。

（２）(1)の大きくなるのが、ある程度大きくなると急激に小さくなる。

（３）曲がった三角形の様な図形が複数現れる。

（４）(3)の図形は、徐々に横幅が小さくなり、やがて、小さくなりすぎたのかわからなくなる。

（５）(3)で曲がった三角形と表現したが、詳しく表現すると左側の斜辺が膨らんでいる。

（６）探索が進むにつれて、点の分布は、ランダムになる。

・仮説

＜１＞(1)(2)(4)(6)の原因は、βの値の変化と関係がある。

＜２＞(1)(2)(4)(6)の原因は、ｋの値の変化と関係がある。

＜３＞(3)(5)の原因は、解曲線が双曲線である為。

＜４＞(1)～(6)の内ほとんどが、合成数の桁数が小さいので、双曲線を直線に近似できないのが原因となっている。

６－２　探索回数とβ・探索回数とｋの関係

a）実験２の方法

データは、プログラム　t_k.cとt_b.c（プログラム参照）とで作成された。

プログラムのアルゴリズムは、図Ｅと同じで、ただ、出力するデータを変えただけのものである。

桁数	合成数（素因数＊素因数）	kデータ	bデータ
8桁	76659109=3541＊21649	out8k.dat	out8b.dat
10桁	9000909991=9901＊909091	out10k.dat	out10b.dat

表２　各合成数のｋとβのグラフ・データ

b）実験２の結果

グラフは以下の通り。

桁数	βのグラフ	ｋのグラフ
８桁	図２－１	図３－１
１０桁	図２－２	図３－２

表３　各合成数のグラフ図

このグラフは、実験１の仮説の<1><2>を確かめる為、以下の合成数のβとｋを立て軸に取ったものである。

ｃ）実験２の考察

まず、図２－１と図１－３並びに図２－２と図１－４を比べてみるとβの変化が、分布状況(2)(4)(6)の原因になっているのがわかる。その理論的根拠は、以下の３つの傾向より明かである。

＜１－１＞βの変化の点が、分布状況(2)の急激に変化する点付近となる。　→　分布状況(2)

＜１－２＞変形した三角形の幅が、βが変化してから次の変化をするまでの間隔と等しい。　→　分布状況(4)

＜１－３＞探索終わりごろのβの変化は、著しい。　→　分布状況(6)

また、図３－１と図１－３並びに図３－２と図１－４を比べてみるとｋの変化が分布状況(1)(2)(3)(4)の原因になっているのがわかる。

＜２－１＞ｋが急激に変化した点が、分布状況(2)の急激に変化した点付近となる。　→　分布状況(2)

＜２－２＞変形した三角の幅が、ｋが急激に変化してから次の急激な変化をするまでの間隔と等しい。　→　分布状況(4)

＜２－３＞急激な変化の合間の折れ線グラフは、双曲線の様に振舞う。　→　分布状況(1)(3)

<2-3>は、論理的根拠というより、仮説である。

βの傾向とｋの傾向で似かよった部分が多くあるのは、ｋがβに依存する値である。つまり、ｋは、

$k=(-(m-\beta n)+sqrt((m-\beta n)^2+4\beta(nm-N)))/2\beta$

であるからだ。

また、現段階でＰ＝１の時しか距離の分布を調べていないのでＰが他の値の時も上述の様な傾向がでるかを６－３で調べる。

仮説の<4>は、現段階で断定できないが、βの傾向からすると正しくないようだ。

６－３　Ｐ＝３の時の分布状況

ａ）実験３の方法

１３桁の合成数　３０１９０５２９７１３６７

データout13keta.datは、プログラムPt_distance.cより作成。このプログラムのアルゴリズムは、図Ｅとたいして変わらないのだが、今までのプログラムが１つづつ解曲線付近の格子点を探索していくのに対して、このプログラムは、素数パラメータＰの数だけ飛ばし飛ばし格子点を探索していく。

つまり、前のプログラムは、このプログラムのＰ＝１の時のものである。

全てのグラフに表示されているＰの値は、この素数の事である。

b）実験３の結果

グラフは、図６－１である。このグラフは、実験１で得られた傾向は、パラメータＰを変えてもも同じ様な傾向が出るかを確認する為のものである。

c）実験３の考察

図６－１は、実験１の結果と同じ傾向を示した。ということは、実験１の結果は、Ｐ＝１の時の特有の傾向ではない。

６－４　非常に大きな桁の合成数

a）実験４の方法

１５０桁の合成数における双曲線との距離をグラフ化する。

１５０桁の合成数の計算は、膨大な時間がかかるので、最後まで計算しないで、途中までのデータをグラフ化した。

（１）7777.....7777　ou150.dat

（２）1000.....0000 out150keta2.dat

データはt_distance.cより作成

b）実験４の結果

グラフは、以下の通り。

（１）図７－１　7777.....7777　out150.dat

（２）図７－２　1000.....0000 out150keta2.dat

このグラフは、図１系統のグラフと見比べて、理論的根拠<4>（実験１の結果は桁が小さい為におこる）が、正しいかを確認する為のものである。

c）実験４の考察

グラフの傾向は、対数関数の様な曲線内に一様に分布する。

これは、実験１の曲がった三角形の先の部分を横に引き伸ばして拡大したものだと考える事が出来る。

よって、仮説<4>は、Ｐ＝１のおいては、正しいことが証明された。

６－５　Ｐ＝９０９１の時の分布状況

a）実験５の方法

１５０桁の合成数　7777.....777　out150P.dat

データout150P.datは、プログラムPt_distance.cより作成。このプログラムのについては、６－３のａ）を参照。

b）実験５の結果

グラフは、図７－３である。

このグラフは、図１系統のグラフと見比べて、理論的根拠<4>（実験１の結果は桁が小さい為におこる）が、正しいかを確認する為のものであるとともに実験３における考察

が非常に大きい桁の合成数でも成り立つかを確認する為のものある。

c）実験５の考察

図７－１と図７－２と同じように、対数関数の様な曲線内に一様に分布する。

仮説<4>は、間違っており、実験１の分布状況(1)～(6)は、大きい桁でもおこる。

そして、実験３の考察（分布状況はＰ＝１の時のみではない）は、大きい桁の時も成り立つ。

６－６　双曲線と最近傍底点の距離

a）実験６の方法

グラフのデータは、プログラムmin.cで作成。

プログラムmin.cとは、実験１，３，４，５で作成されたデータから極小点のみを抽出するものである。極小点を抽出するという事は、距離の値が減少から増加する境の点を抜き出す事である。

・起動方法

min 入力ファイル　＞出力ファイル

桁数	入力ファイル	出力ファイル
8 桁	out8keta.dat	out8min.dat
１０桁	out10keta.dat	out10min.dat
１２桁	out12keta.dat	out12min.dat
１５０桁１	out150.dat	out150min.dat
１５０桁２	out150keta2.dat	out150_2min.dat
１５０桁３	out150P.dat	out150Pmin.dat

表４ 各合成数のグラフ・データ

b）実験６の結果

グラフは、以下の通り。

実験６のグラフ	合成数の桁	抽出される前のグラフ
図４－１	8 桁	図１－３
図４－２	１０桁	図１－４
図４－３	１２桁	図１－５
図４－３－１	１２桁	図１－５
図８－１	１５０桁１	図７－１
図８－２	１５０桁２	図７－２
図８－３	１５０桁３	図７－３

表５　抽出前と抽出後

このグラフは図１－３，１－４，１－５，７－１，７－２，７－３から極小点だけを

抜き出したものである。

なお、図４－３－１は、図４－３の原点付近を拡大したものである。この図の必要性は、図８系統のグラフは、初期の部分にのみのデータである為図４－１，４－２，４－３と比較しても、小さい桁と大きい桁の違いや同一性を確認することは出来ない為である。

ｃ）実験６の考察

桁の小さい合成数の結果は、幾何学的な模様が出る。幾何学的な図形が出る事より、周期性あるいは、なんらかの法則が見いだされるかも知れない。　これは、実験７より明らかになるだろう。

また、大きい桁の合成数は、幾何学的な模様は、現れなかった。また、図４－３－１との同一性は、見いだされなかった。

ということは、図８系統のグラフの縦軸のメモリを見てもわかるように、図４－３－１より非常に原点に近い部分である。つまり比較の範囲が違うのだ。

拡大部分を同じ範囲にしたいのだが、この傾向を解析するのは、本報告書の目的とする所ではないので、別の機会に譲る。

６－７　最近傍低点の出現間隔

ａ）実験７の方法

グラフのデータは、プログラムkankaku.cで作成。

プログラムkankaku.cとは、実験６で作成されたデータから最近傍底点の出現間隔を計算し出力するものである。最近傍底点の出現間隔とは、抽出された隣合う２つの探索回数の差である。

・起動方法

kankaku 入力ファイル　＞出力ファイル

桁数	入力ファイル	出力ファイル
８桁	out8min.dat	8kan.dat
１０桁	out10min.dat	10kan.dat
１２桁	out12min.dat	12kan.dat
１５０桁１	out150.dat	150kan.dat
１５０桁２	out150_2min.dat	150_2kan.dat
１５０桁３	out150P.dat	150Pkan.dat

表６ 各合成数のグラフ・データ

ｂ）実験７の結果

グラフは、以下の通り。

図５－１　　８桁
図５－２　　１０桁
図５－３　　１２桁

図９－１　　１５０桁１
図９－２　　１５０桁２
図９－３　　１５０桁３

このグラフは、最近傍低点の出現間隔に周期性があるかを調査するためのものである。

ｃ）実験７の考察

小さい桁が合成数の図５系統のグラフは、最近傍底点出現間隔は、時々急激に大きくなるが、全体的に余り大きくならない。

Ｐ＝１で大きな桁が合成数の図９－１，９－２のグラフは、最近傍底点出現間隔が2,3,4,5,6,7,8,9に限定されている。点の数は、最近傍底点出現間隔２が一番多くて次に３、次に４と言う具合に探索間隔が大きくなるにつれて点の数は少なくなる。

しかし、その８種類の最近傍底点出現間隔の出現は、周期的ではなかった。　Ｐ＝９０９１で大きな桁の合成数の図９－３のグラフは、最近傍底点出現間隔の種類は、2,3,4,5,6,7,8,9,10と増えたが、Ｐ＝１と同じ様な分布状況だ。

小さい桁の図と大きい桁の図の違いは、出現間隔の最大の値が小さい桁の時の方が、大きい桁の時より大きい。これは、上述の様に大きい桁の場合データの範囲が原点付近に限られている為である。

７．まとめ

実験１～７の考察より、探索回数における周期性がないのは明らかである。現在は実験７で探索回数における最近傍底点の出現間隔しか調べていないが、ｎやｍにおける出現間隔を調べて例え周期性を見いだせても、大幅な計算量の削減は、望めないだろう。なぜなら、実験７の考察より最近傍底点の出現間隔がそれほど大きくなっていないからである。

ということは、格子点探索法における最近傍低点の周期性を扱う素因数分解は、残念ながら高速化は見込めない。

８．参考文献

（１）永瀬，竹中，山下，”解曲線 xy=N の近傍点探索による素因数分解”，SCIS91-10C
（２）池田，小山，”現代暗号理論”電子通信学会，（昭和６１）
（３）”数理科学・新しい暗号－高速処理の技術”，サイエンス社(8/1986)
（４）永瀬，力石，竹中，山下，”複数の剰余類を考慮した素因数分解”，ISEC91-11
（５）和田，”コンピュータと素因子分解”，遊星社（１９８７）

９．感想

ＡＴＲに来て初めて来た時は、机を一つ与えられて一人前の研究員になった様な気になり嬉しかった。実験テーマについては、”格子点探索における最近傍底点の周期性の確認”と言うたいへん興味深いものだったので、たいへんやりがいがあった。

企業実習で学んだことは、以下の４つである。

（１）計画を立てて仕事をすること。

（２）仕事の経過は、記録しておくこと。

（３）データの整理には、惜しみなく文具を使う。

（４）報告書は早めに作成すること。

たいへん有意義な４週間でした。

ただ１つ心残りなのは、期間が終わる頃になると、時間的余裕がなくなって、仕事が雑になってしまった。上述の４項目を実習前に身につけていたらなと思いました。

１０．謝辞

本研究を行なう機会を与えて頂いた当研究所寺島社長、竹中室長、そして本研究に関し、有益な御助言を頂いた力石研究員をはじめとする当研究所の皆様方に感謝します。

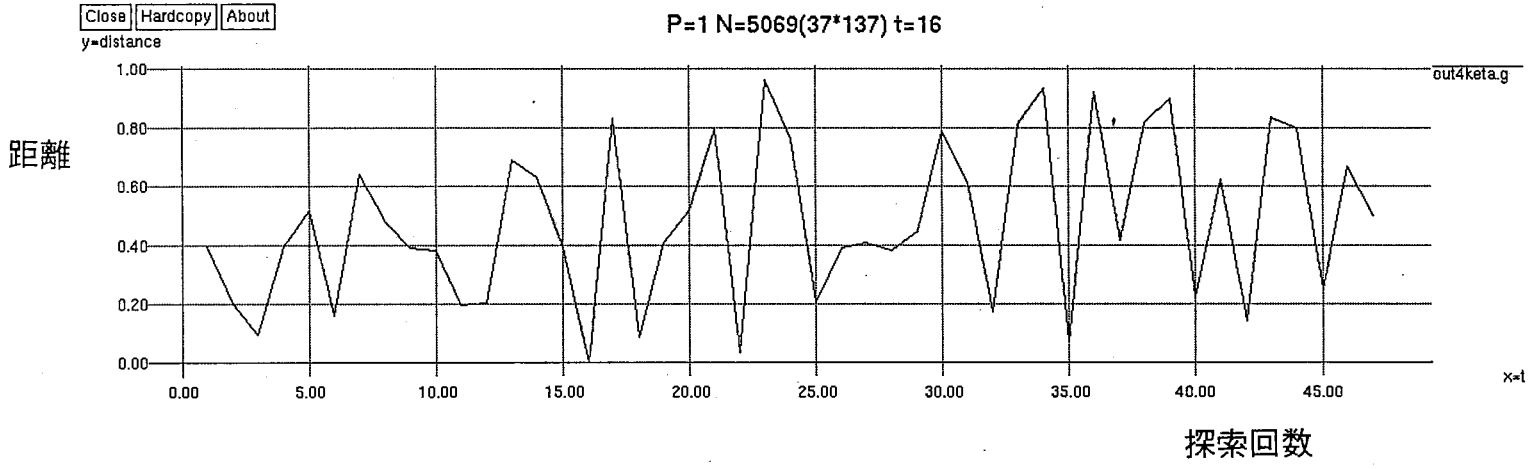


図１-１　双曲線と底点の距離 ： 合成数 5069

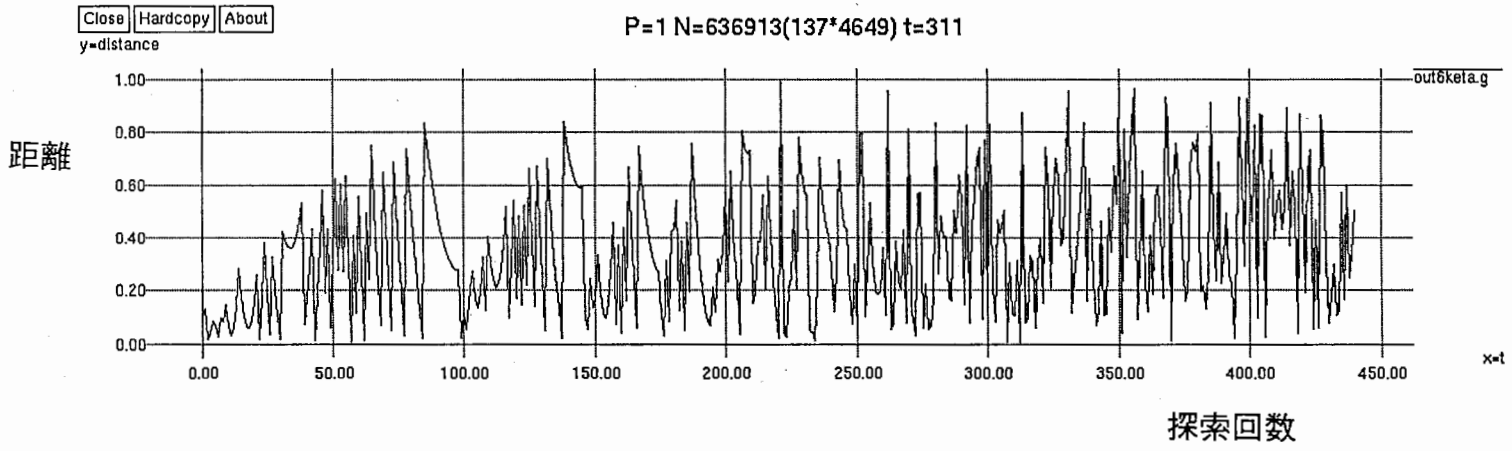


図１-２　双曲線と底点の距離 ： 合成数 636913

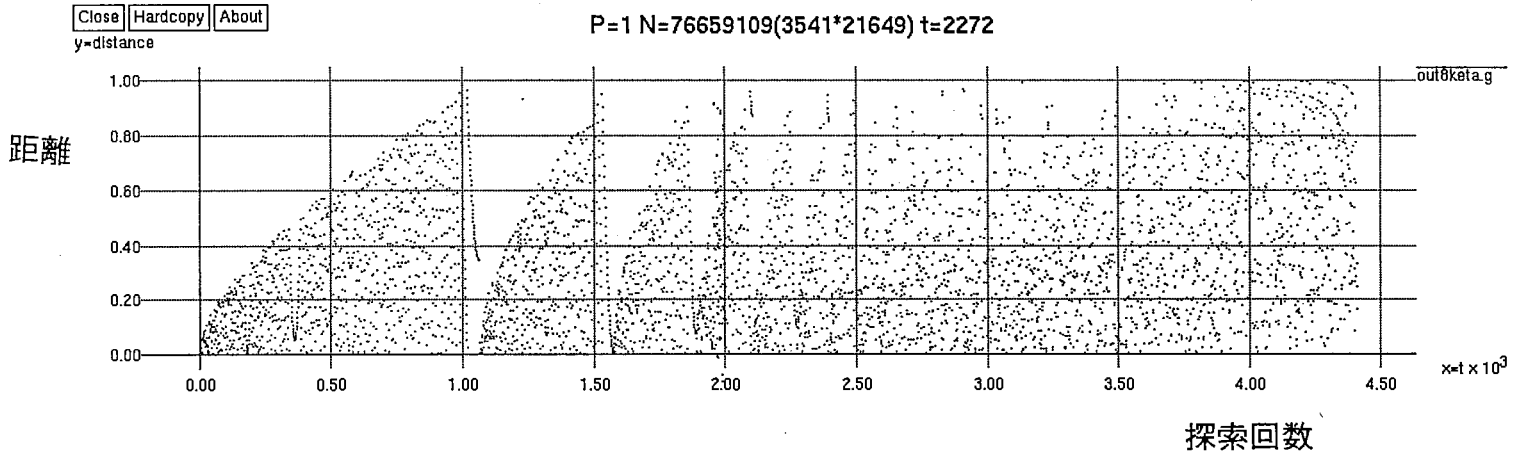


図１-３　双曲線と底点の距離 ： 合成数 76659109

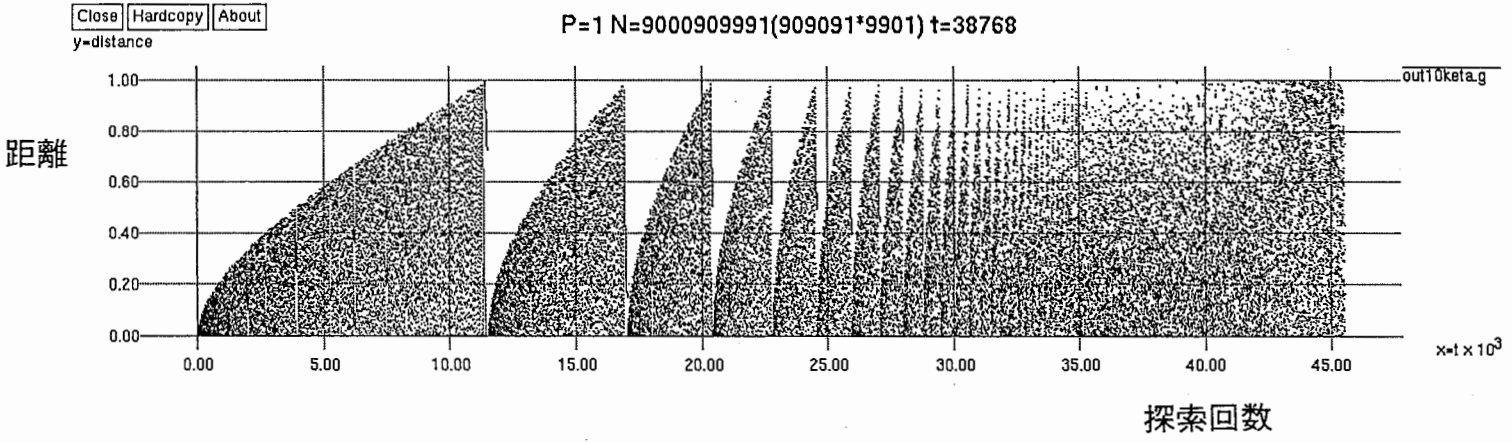


図１-４　双曲線と底点の距離 ： 合成数 9000909991

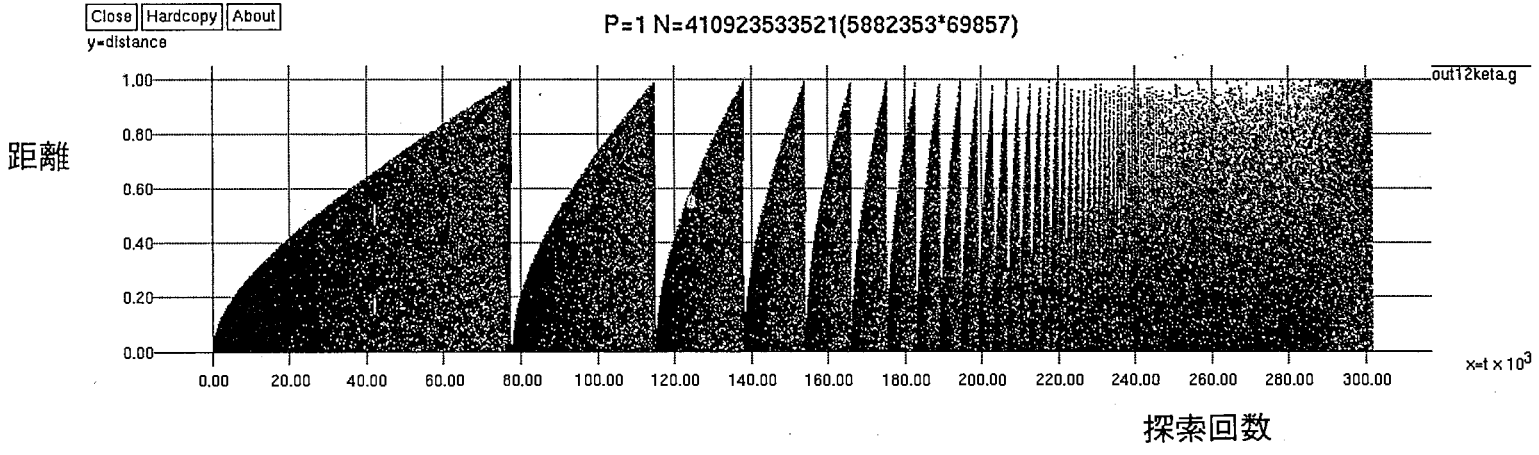


図１-５　双曲線と底点の距離 ： 合成数 410923533521

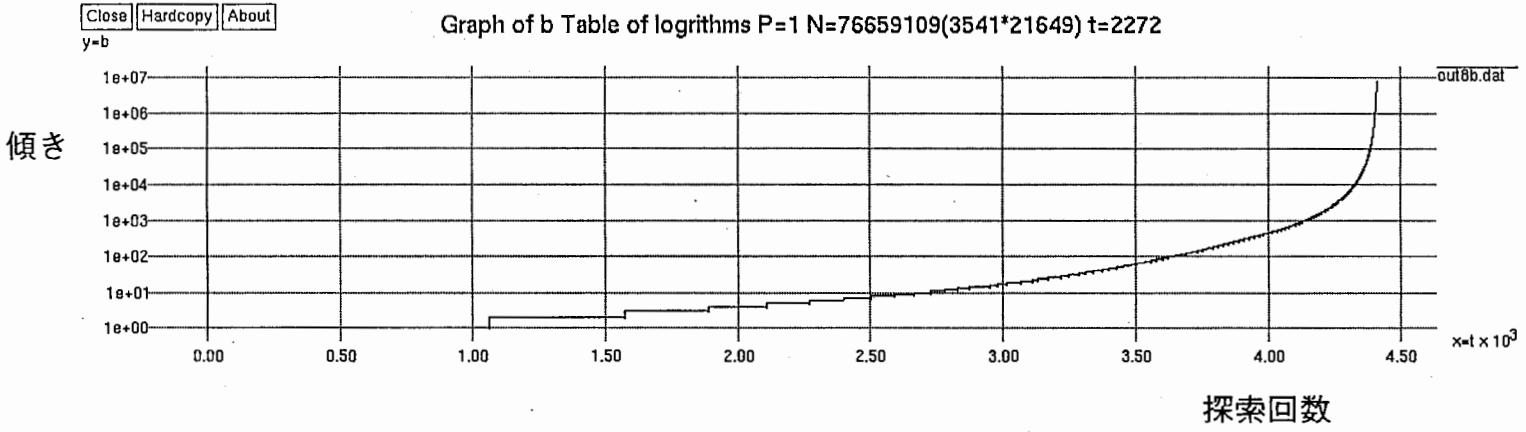


図 2 - 1　探索直線の傾き ： 合成数 76659109

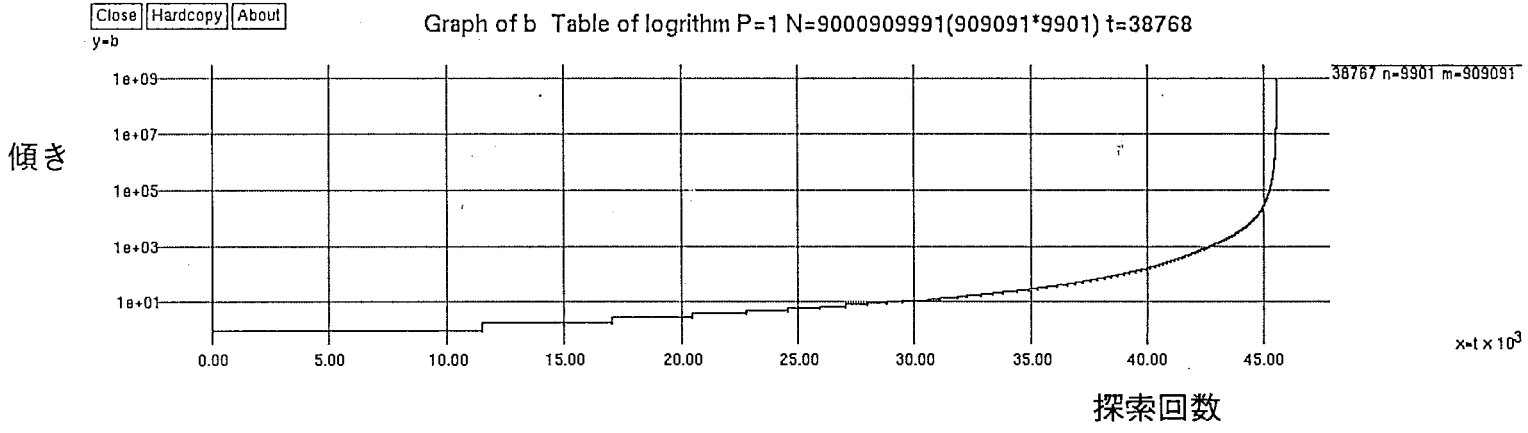


図 2 - 2　探索直線の傾き ： 合成数 9000909991

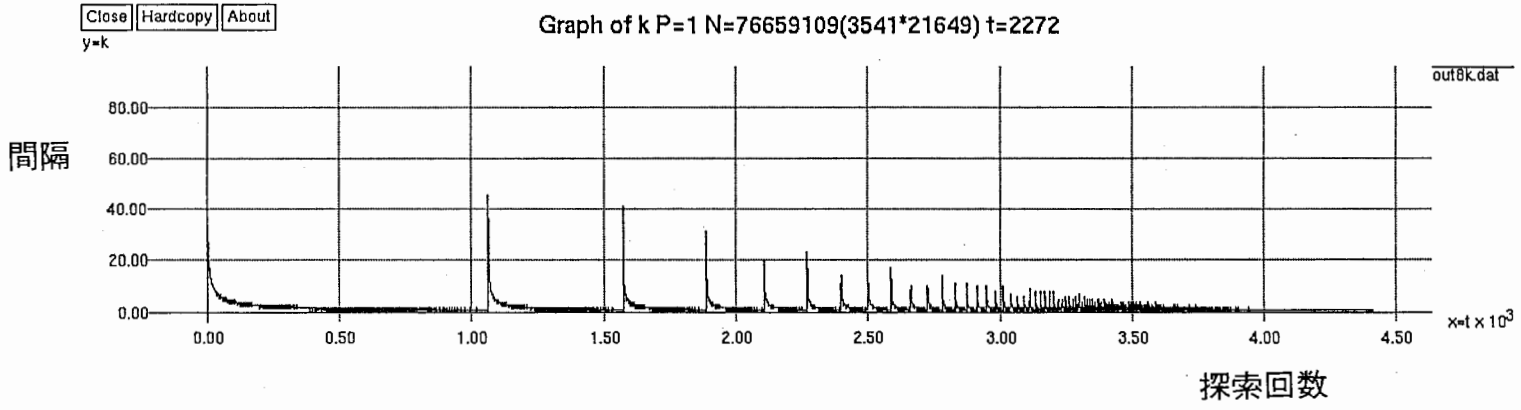


図 3 - 1　探索間隔 ： 合成数 76659109

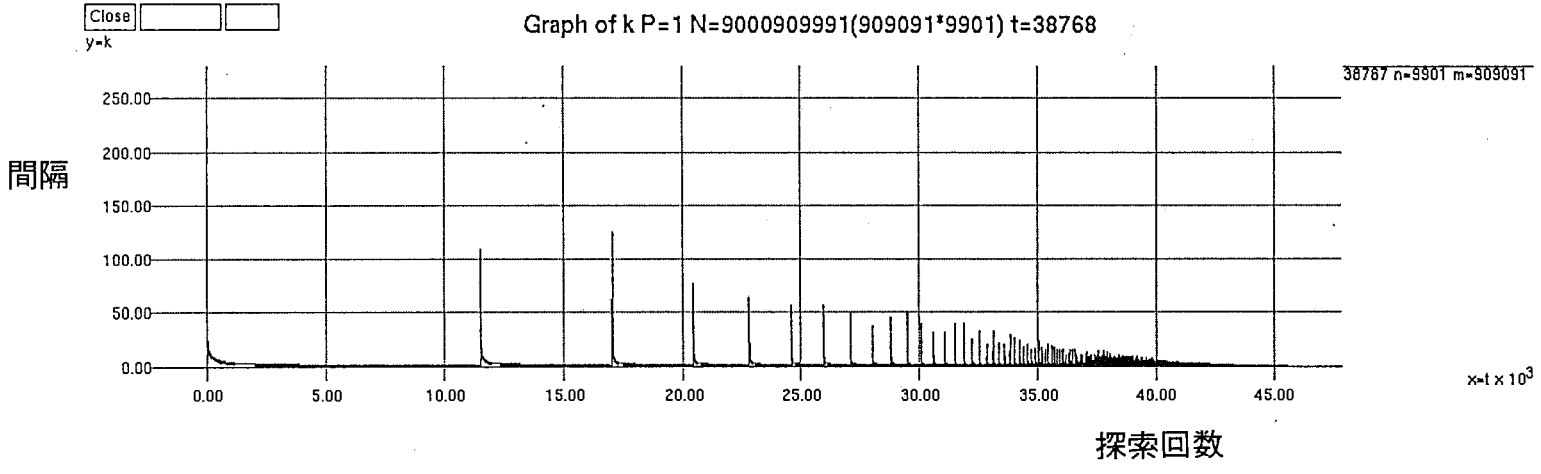


図 3 - 2　探索間隔 ： 合成数 9000909991

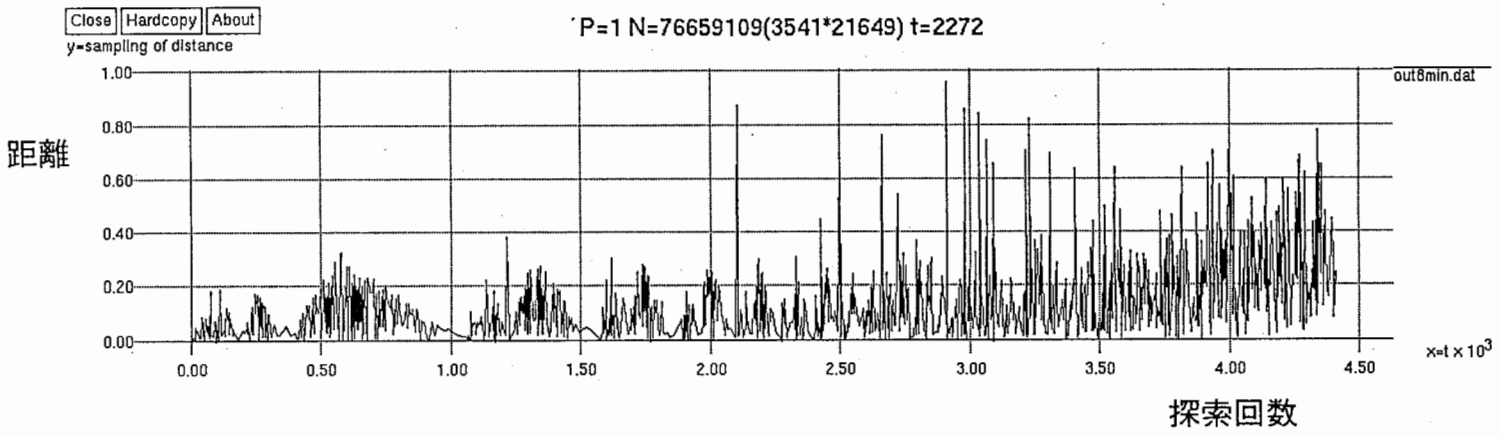


図 4 - 1　双曲線と最近傍底点の距離 ： 合成数 76659109

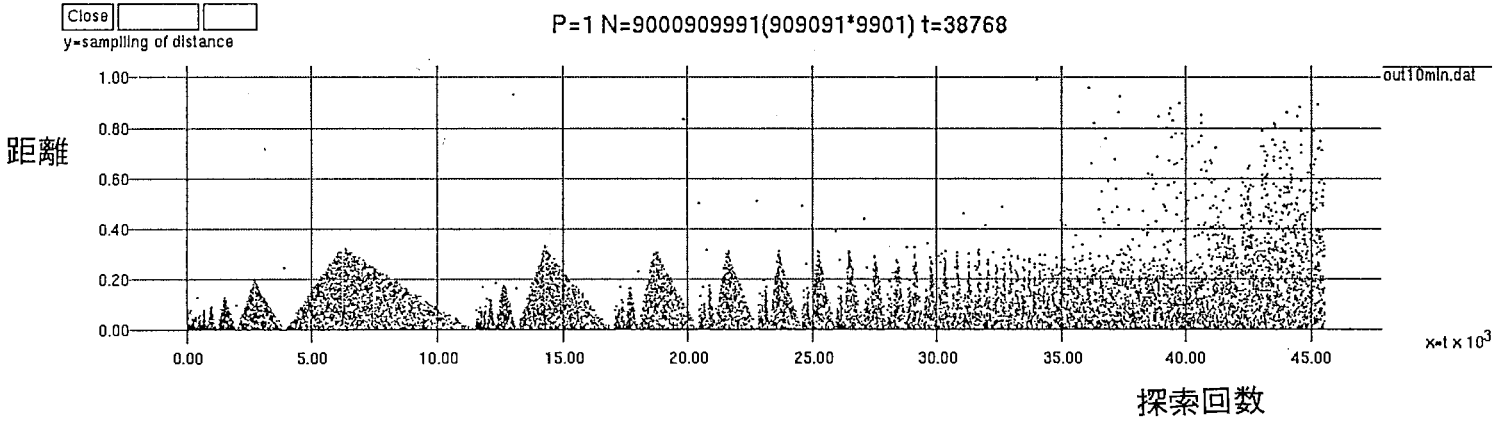


図 4 - 2　双曲線と最近傍底点の距離 ： 合成数 9000909991

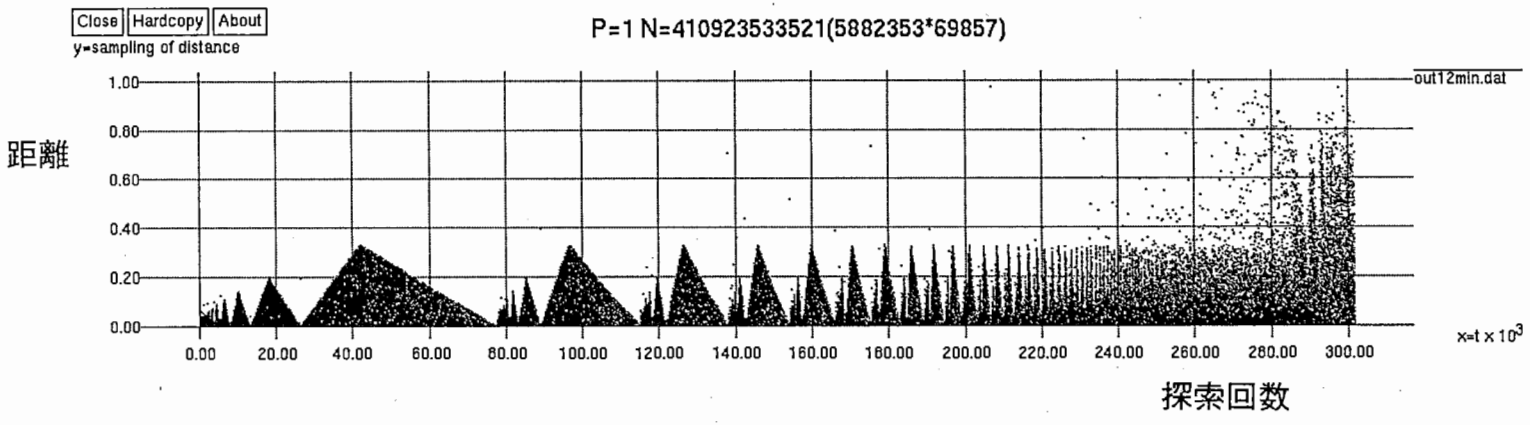


図 4 - 3　双曲線と最近傍底点の距離 ： 合成数 410923533521

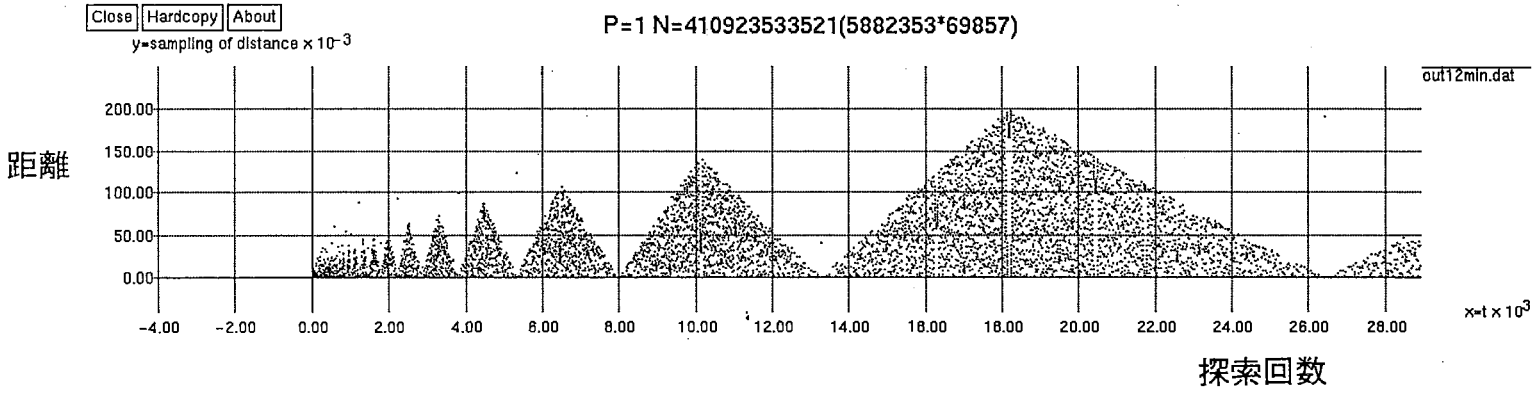


図 4 - 3 - 1　双曲線と最近傍底点の距離 （拡大図） ： 合成数 410923533521

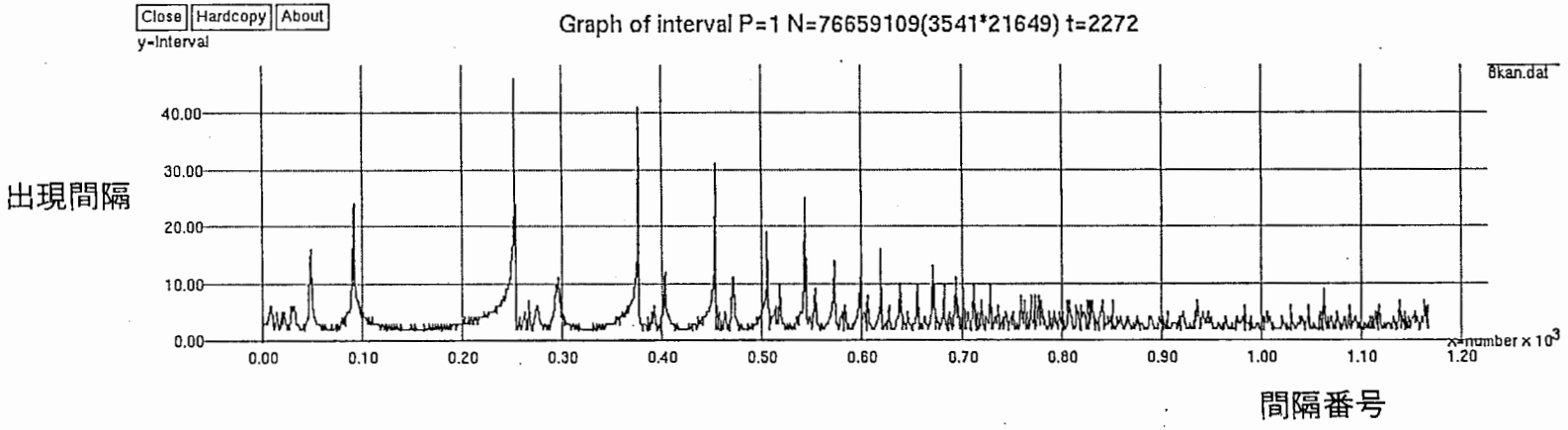


図 5 - 1　最近傍底点の出現間隔 ： 合成数 7659109

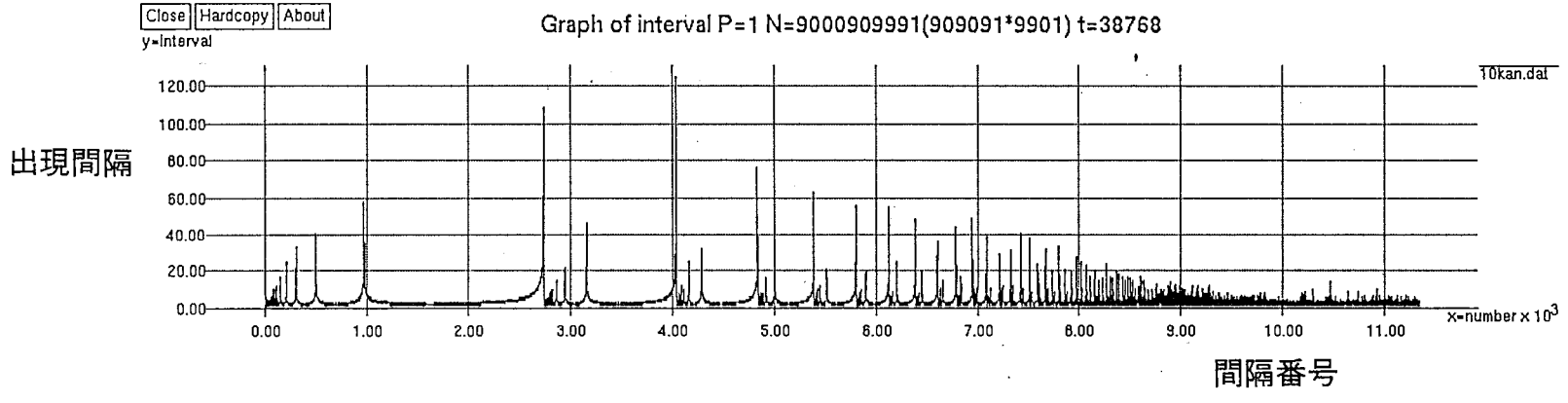


図 5 - 2　最近傍底点の出現間隔 ： 合成数 9000909991

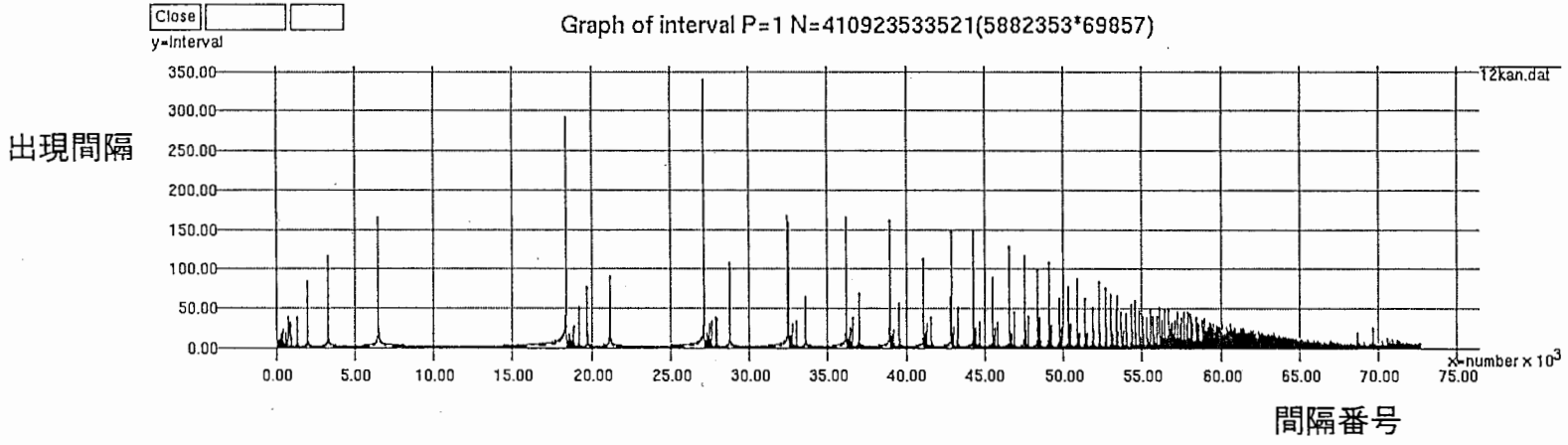


図５-３　最近傍底点の出現間隔 ： 合成数 410923533521

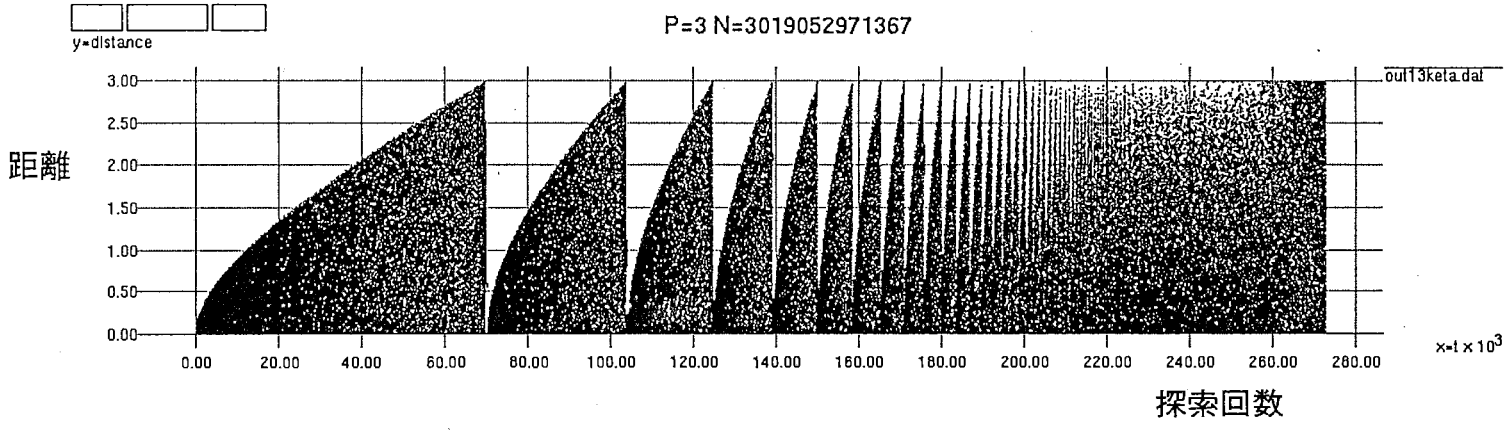


図 6 - 1　双曲線と底点の距離 ： 合成数 3019052971367　素数 3

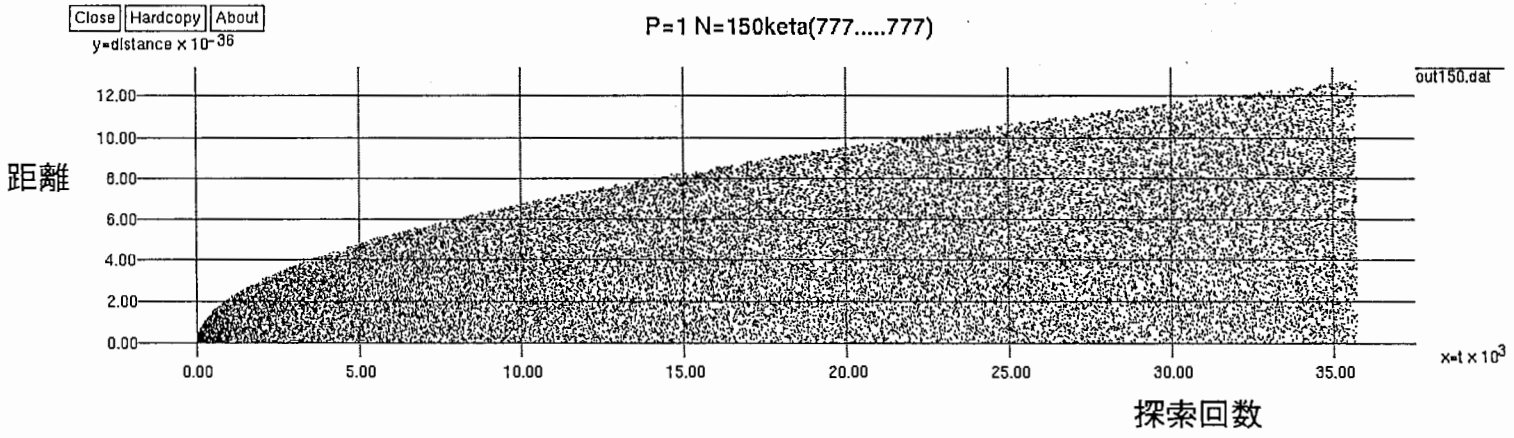


図７-１　双曲線と底点の距離 ： 合成数 150桁のALL7

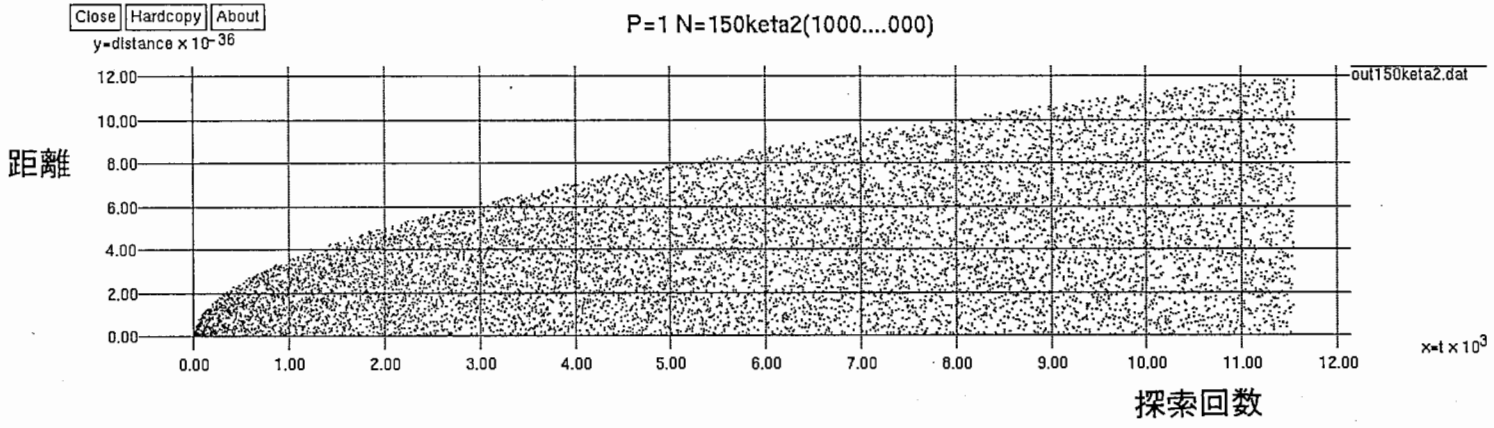


図７-２　双曲線と底点の距離 ： 合成数 10の150乗

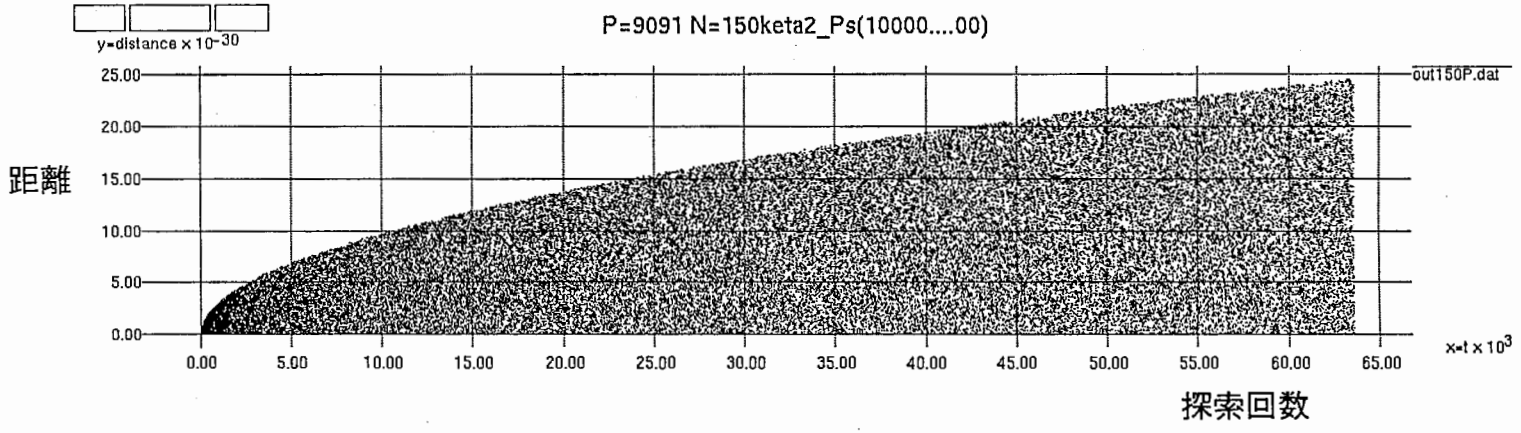


図７-３　双曲線と底点の距離 ： 合成数 10の150乗　素数 9091

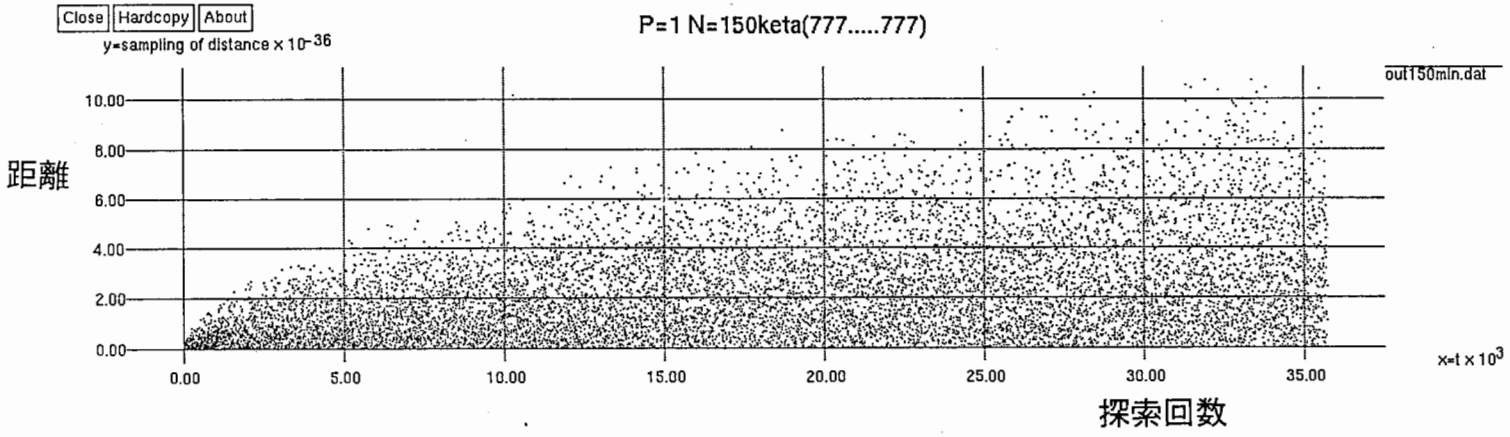


図８-１　双曲線と最近傍底点の距離 ： 合成数 150桁のALL7

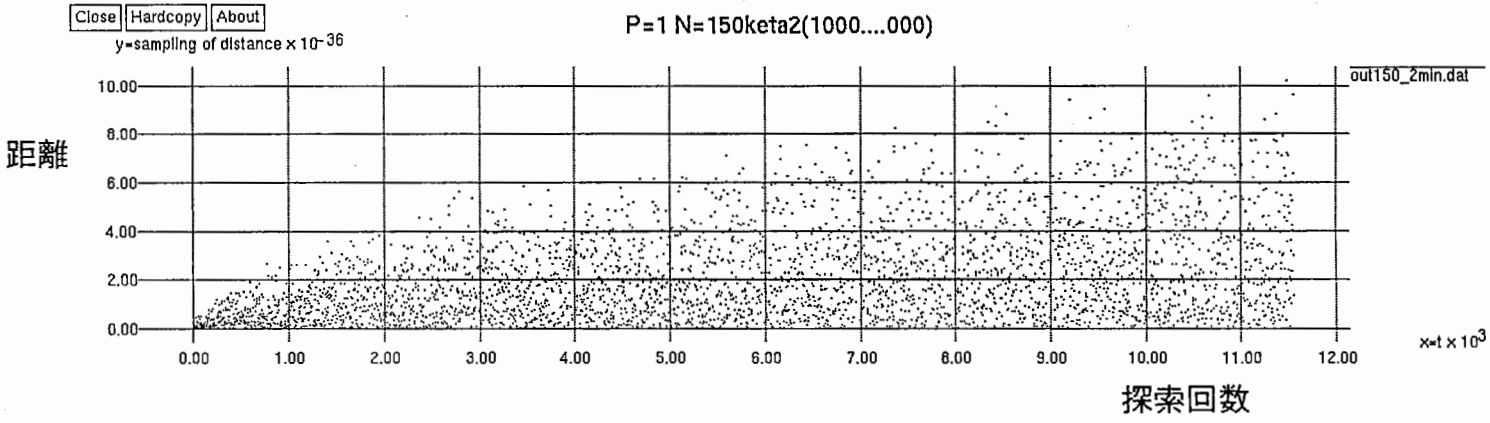


図８-２　双曲線と最近傍底点の距離 ： 合成数 10の150乗

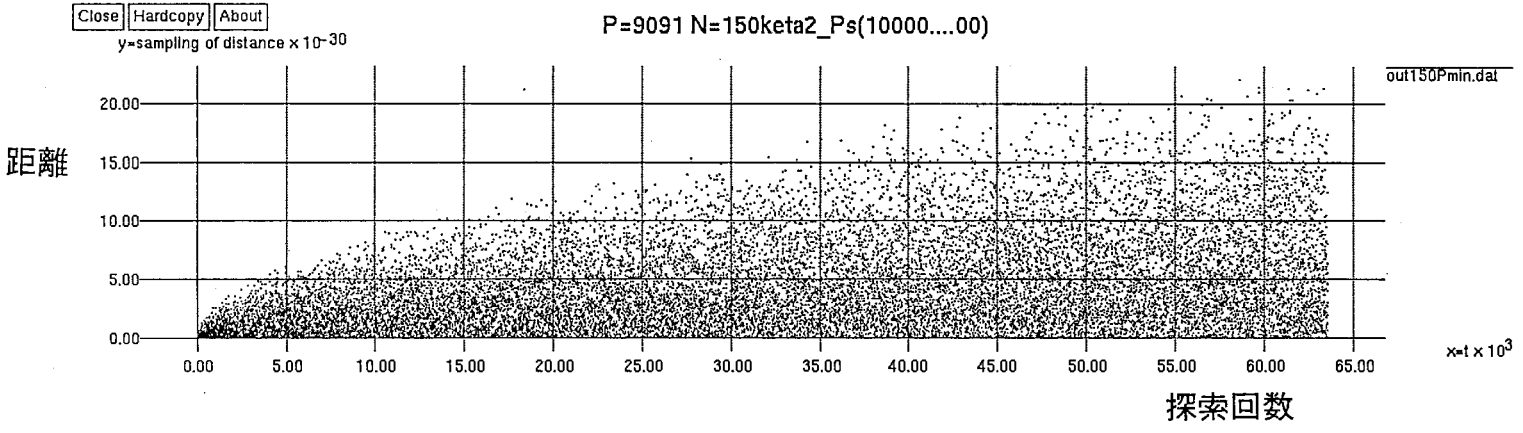


図８-３　双曲線と最近傍底点の距離 ： 合成数 10の150乗　素数 9091

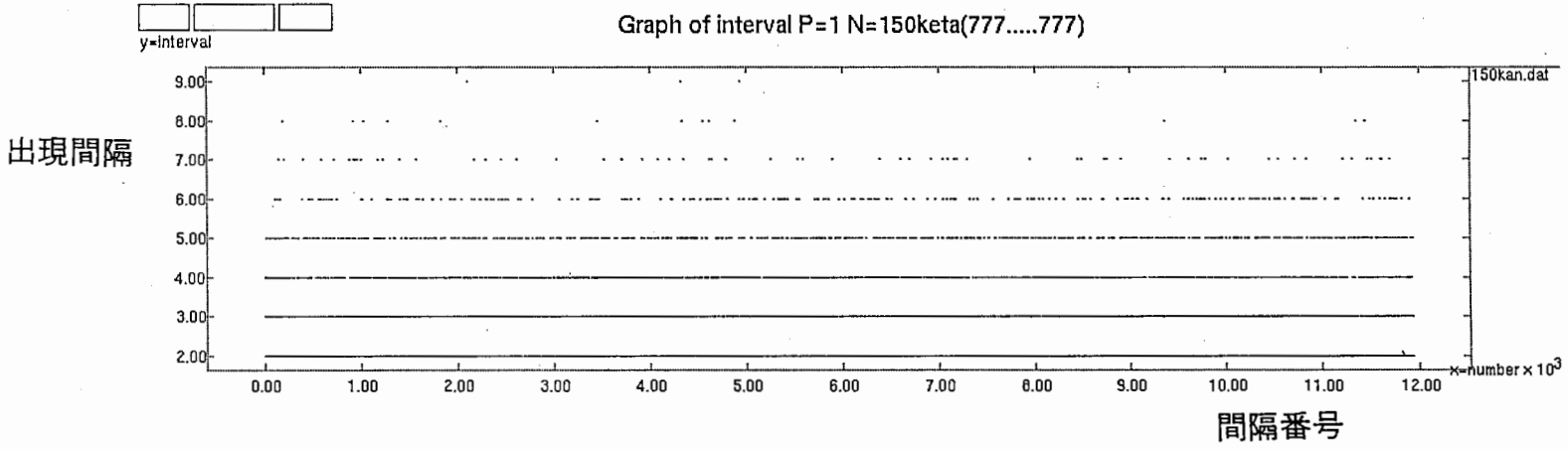


図９-１　最近傍底点の出現間隔 ： 合成数 150桁のALL7

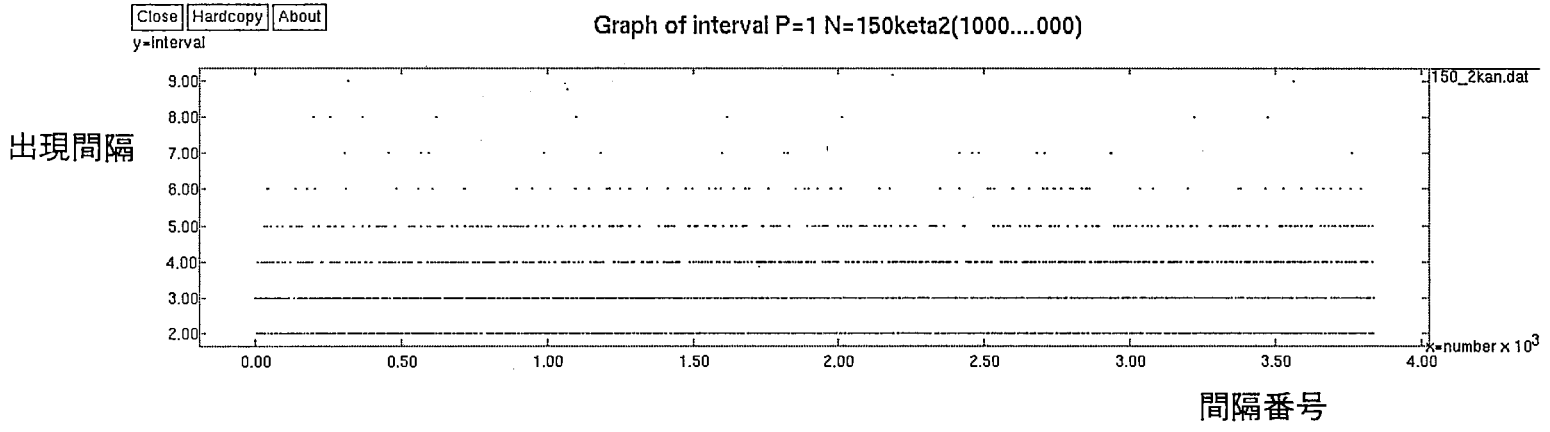


図９-２　最近傍底点の出現間隔 ： 合成数 10の150乗

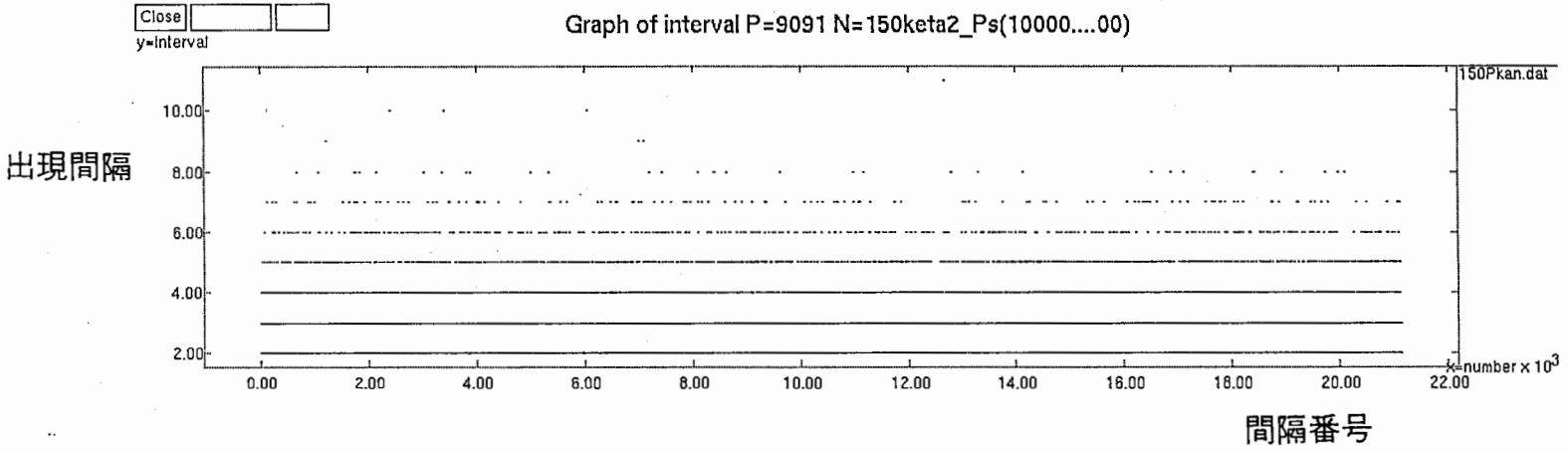


図９-３　最近傍底点の出現間隔 ： 合成数 10の150乗　素数 9091

```
/*
        file name  t_distance.c

          npddec.c(by H. Deguchi)を改良したプログラム
*/

#include        <stdio.h>
#include        <math.h>
#include        "mlib.h"


extern  Num     get_k();
extern  int     fcand();
extern  int     ncand();
extern  int     decomp();

static  int     turn = 0;

main( argc, argv )
int     argc;
char    *argv[];
{
        Num     N_num, n_num, m_num, z_num, P_num;
        char    buf[255];

        setbuf(stdout,NULL);    /* no bufferring */

        /* N */
        if( argc > 1 )
                N_num = stonum( argv[1] );
        else
                N_num = stonum("1315753");
        z_num = stonum("1");
        while(1)
        {
                scanf("%s", buf );
                if( strcmp( buf, "n" ) == 0 )
                {
                        scanf("%s", buf );
                        copynum( stonum(buf), N_num );
                }
                else
                        break;
        }

        P_num = stonum("1");

        n_num = m_num = Newnum;

        printf("XUnitText: x=t¥n");
        printf("YUnitText: y=distance¥n");
        printf("Device: Postscript¥n");
        printf("Disposition: To Device¥n");
        printf("FileOrDev: lpr -Pimagen¥n");
        printf("ZeroColor: black¥n");

        decomp( N_num, n_num, m_num, z_num,P_num );

        printf("TitleText: P=1 N=");
        prnum(N_num);
        CR;

}

/* axb = a * x - b */
Num axbnum( a_num, x_num, b_num, axb_num )
Num a_num, x_num, b_num, axb_num;
{
        static  NUM     tmp;

        return( subnum( mulnum(a_num, x_num, &tmp), b_num, axb_num ) );
}

/* get the first candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        m / n < z
        return 1: if found, 0: otherwise
*/
int fcand( N_num, n_num, m_num, z_num, P_num )
Num     N_num, *n_num, *m_num, z_num, P_num;
{
        static  NUM     nm_num = NUM_0; /* n * m */
        static  NUM     mP_num = NUM_0; /* m - P */
        static  NUM     tmp = NUM_0;    /* tmp */

        *n_num = sqrtnum( divnum(N_num,z_num,&tmp), *n_num );
        if( signum(fraction(*n_num,&tmp)) == 0 )
        {
                *m_num = divnum( N_num, *n_num, *m_num );
                return( 1 );
        }
        ceilingnum( *n_num, *n_num );
        subnum( *n_num, P_num, *n_num );
        *m_num = divnum( N_num, *n_num, *m_num );
        ceilingnum( *m_num, *m_num );
        mulnum( *n_num, *m_num, &nm_num );
```

```
        if( compnum( &nm_num, N_num ) < 0 )
                panic("fcand error : n * m < N¥n");
        subnum( *m_num, P_num, &mP_num );
        mulnum( *n_num, &mP_num, &nm_num );
        if( compnum( &nm_num, N_num ) >= 0 )
                panic("fcand error : n * (m-P) >= N¥n");
        return( 0 );
}

/* k = (-(m-b*n)+sqrt((m-b*n)^2+4*b*(n*m-N)))/2*b */
Num get_k( N_num, n_num, m_num, beta_num, k_num, P_num )
Num     N_num, n_num, m_num, beta_num, k_num, P_num;
{
        static  NUM     mbn_num = NUM_0;        /* m-b*n */
        static  NUM     mbn2_num = NUM_0;       /* (m-b*n)^2 */
        static  NUM     nmN_num = NUM_0;        /* n*m-N */
        static  NUM     tmp1 = NUM_0;           /* for working */
        static  NUM     tmp2 = NUM_0;           /* for working */
        static  NUM     tmp3 = NUM_0;           /* for working */

        subnum( m_num, mulnum( beta_num, n_num, &tmp1 ), &mbn_num );
        mulnum( &mbn_num, &mbn_num, &mbn2_num );
        axbnum( n_num, m_num, N_num, &nmN_num );
        mulnum( mulnum( Four, beta_num, &tmp1), &nmN_num, &tmp2 );
        sqrtnum( addnum( &mbn2_num, &tmp2, &tmp1 ), &tmp2 );
        subnum( &tmp2, &mbn_num, &tmp1 );
        divnum( &tmp1, mulnum( mulnum( Two, P_num, &tmp3 ), beta_num, &tmp2 ),
            k_num );
        return( k_num );
}

/* get the next candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        return 1: if found, 0: otherwise
*/
int ncand( N_num, n_num, m_num, P_num )
Num     N_num, *n_num, *m_num, P_num;
{
        static  NUM     beta_num = NUM_0;       /* beta = f[m/n] */
        static  NUM     k_num = NUM_0;          /* k */
        static  NUM     nmN_num = NUM_0;        /* n * m - N */
        static  NUM     tmp1 = NUM_0;
        static  NUM     tmp2 = NUM_0;
        int     i;
        int     fk;

        /* output t */
        printf("%d ",turn);

        floornum( divnum( *m_num, *n_num, &tmp1 ), &beta_num );

        divnum( N_num, *n_num, &tmp1 );
        subnum( P_num, subnum( *m_num, &tmp1, &tmp2 ), &tmp1 );
        prnum( &tmp1 );
        CR;

        get_k( N_num, *n_num, *m_num, &beta_num, &k_num, P_num );
        fk = signum( fraction( &k_num, &tmp1 ) );
        ceilingnum( &k_num, &k_num );

        /* n = n -  k */
        subnum( *n_num, mulnum( P_num, &k_num, &tmp1 ), *n_num );

        /* m = m + P * b * k */
        mulnum( mulnum( P_num, &beta_num, &tmp2), &k_num, &tmp1 );
        addnum( *m_num, &tmp1, *m_num );
        if( fk == 0 )
        {
                return( 1 );
        }
        /* fk != 0 */
        for( i = 0;
            (fk = signum( axbnum( *n_num, *m_num, N_num, &nmN_num ) )) < 0;
            i++ )
        {
                if( i > 2 )
                {
                        /*i = -1;*/
                        copynum( Zero, &tmp1 );
                        divnum( N_num, *n_num, &tmp1 );
                        ceilingnum( &tmp1, *m_num );
                }
                else
                {
                        addnum( *m_num, P_num, *m_num );
                }
        }
        if( fk == 0 )
        {
                return( 1 );
        }
        else
        {
                return( 0 );
        }
}

int decomp( N_num, n_num, m_num, z_num, P_num )
```

```
Num     N_num, n_num, m_num, z_num, P_num;
{
        static  NUM     tmp;
        static  Num     two_num;

        if( fcand( N_num, &n_num, &m_num, z_num, P_num ) ){
                printf("¥"%d n=",turn);
                prnum(n_num);
                printf(" m=");
                prnum(m_num);
                CR;
                addnum( m_num, P_num, m_num);
        }

        two_num = stonum("2");
        while( turn++, signum( subnum( n_num, two_num, &tmp) ) > 0 )
        {
                if( ncand( N_num, &n_num, &m_num, P_num) ){
                        printf("¥"%d n=",turn);
                        prnum(n_num);
                        printf(" m=");
                        prnum(m_num);
                        CR;
                        addnum( m_num, P_num, m_num );
                }
        }
}

/* end of t_distance.c */
```

```
/*
        file name  t_k.c

          npddec.c(by H. Deguchi)を改良したプログラム
*/

#include        <stdio.h>
#include        <math.h>
#include        "mlib.h"


extern  Num     get_k();
extern  int     fcand();
extern  int     ncand();
extern  int     decomp();

static  int     turn = 0;

main( argc, argv )
int     argc;
char    *argv[];
{
        Num     N_num, n_num, m_num, z_num, P_num;
        char    buf[255];

        setbuf(stdout,NULL);    /* no bufferring */

        /* N */
        if( argc > 1 )
                N_num = stonum( argv[1] );
        else
                N_num = stonum("1315753");
        z_num = stonum("1");
        while(1)
        {
                scanf("%s", buf );
                if( strcmp( buf, "n" ) == 0 )
                {
                        scanf("%s", buf );
                        copynum( stonum(buf), N_num );
                }
                else
                        break;
        }

        P_num = stonum("1");

        n_num = m_num = Newnum;

        printf("XUnitText: x=t¥n");
        printf("YUnitText: y=distance¥n");
        printf("Device: Postscript¥n");
        printf("Disposition: To Device¥n");
        printf("FileOrDev: lpr -Pimagen¥n");
        printf("ZeroColor: black¥n");

        decomp( N_num, n_num, m_num, z_num,P_num );

        printf("TitleText: P=1 N=");
        prnum(N_num);
        CR;

}

/* axb = a * x - b */
Num axbnum( a_num, x_num, b_num, axb_num )
Num a_num, x_num, b_num, axb_num;
{
        static  NUM     tmp;

        return( subnum( mulnum(a_num, x_num, &tmp), b_num, axb_num ) );
}

/* get the first candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        m / n < z
        return 1: if found, 0: otherwise
*/
int fcand( N_num, n_num, m_num, z_num, P_num )
Num     N_num, *n_num, *m_num, z_num, P_num;
{
        static  NUM     nm_num = NUM_0; /* n * m */
        static  NUM     mP_num = NUM_0; /* m - P */
        static  NUM     tmp = NUM_0;    /* tmp */

        *n_num = sqrtnum( divnum(N_num,z_num,&tmp), *n_num );
        if( signum(fraction(*n_num,&tmp)) == 0 )
        {
                *m_num = divnum( N_num, *n_num, *m_num );
                return( 1 );
        }
        ceilingnum( *n_num, *n_num );
        subnum( *n_num, P_num, *n_num );
        *m_num = divnum( N_num, *n_num, *m_num );
        ceilingnum( *m_num, *m_num );
        mulnum( *n_num, *m_num, &nm_num );
        if( compnum( &nm_num, N_num ) < 0 )
```

```
                panic("fcand error : n * m < N¥n");
        subnum( *m_num, P_num, &mP_num );
        mulnum( *n_num, &mP_num, &nm_num );
        if( compnum( &nm_num, N_num ) >= 0 )
                panic("fcand error : n * (m-P) >= N¥n");
        return( 0 );
}

/* k = (-(m-b*n)*sqrt((m-b*n)^2+4*b*(n*m-N)))/2*b */
Num get_k( N_num, n_num, m_num, beta_num, k_num, P_num )
Num     N_num, n_num, m_num, beta_num, k_num, P_num;
{
        static  NUM     mbn_num = NUM_0;        /* m-b*n */
        static  NUM     mbn2_num = NUM_0;       /* (m-b*n)^2 */
        static  NUM     nmN_num = NUM_0;        /* n*m-N */
        static  NUM     tmp1 = NUM_0;           /* for working */
        static  NUM     tmp2 = NUM_0;           /* for working */
        static  NUM     tmp3 = NUM_0;           /* for working */

        subnum( m_num, mulnum( beta_num, n_num, &tmp1 ), &mbn_num );
        mulnum( &mbn_num, &mbn_num, &mbn2_num );
        axbnum( n_num, m_num, N_num, &nmN_num );
        mulnum( mulnum( Four, beta_num, &tmp1), &nmN_num, &tmp2 );
        sqrtnum( addnum( &mbn2_num, &tmp2, &tmp1 ), &tmp2 );
        subnum( &tmp2, &mbn_num, &tmp1 );
        divnum( &tmp1, mulnum( mulnum( Two, P_num, &tmp3 ), beta_num, &tmp2 ),
            k_num );
        return( k_num );
}

/* get the next candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        return 1: if found, 0: otherwise
*/
int ncand( N_num, n_num, m_num, P_num )
Num     N_num, *n_num, *m_num, P_num;
{
        static  NUM     beta_num = NUM_0;       /* beta = f[m/n] */
        static  NUM     k_num = NUM_0;          /* k */
        static  NUM     nmN_num = NUM_0;        /* n * m - N */
        static  NUM     tmp1 = NUM_0;
        static  NUM     tmp2 = NUM_0;
        int     i;
        int     fk;

        /* output t */
        printf("%d ",turn);

        floornum( divnum( *m_num, *n_num, &tmp1 ), &beta_num );

        get_k( N_num, *n_num, *m_num, &beta_num, &k_num, P_num );
        fk = signum( fraction( &k_num, &tmp1 ) );
        ceilingnum( &k_num, &k_num );
        prnum(&k_num);
        CR;

        /* n = n -  k */
        subnum( *n_num, mulnum( P_num, &k_num, &tmp1 ), *n_num );

        /* m = m + P * b * k */
        mulnum( mulnum( P_num, &beta_num, &tmp2), &k_num, &tmp1 );
        addnum( *m_num, &tmp1, *m_num );
        if( fk == 0 )
        {
                return( 1 );
        }
        /* fk != 0 */
        for( i = 0;
            (fk = signum( axbnum( *n_num, *m_num, N_num, &nmN_num ) )) < 0;
            i++ )
        {
                if( i > 2 )
                {
                        /*i = -1;*/
                        copynum( Zero, &tmp1 );
                        divnum( N_num, *n_num, &tmp1 );
                        ceilingnum( &tmp1, *m_num );
                }
                else
                {
                        addnum( *m_num, P_num, *m_num );
                }
        }
        if( fk == 0 )
        {
                return( 1 );
        }
        else
        {
                return( 0 );
        }
}

int decomp( N_num, n_num, m_num, z_num, P_num )
Num     N_num, n_num, m_num, z_num, P_num;
{
        static  NUM     tmp;
        static  Num     two_num;
```

```
        if( fcand( N_num, &n_num, &m_num, z_num, P_num ) ){
                printf("¥"%d n=", turn);
                prnum(n_num);
                printf(" m=");
                prnum(m_num);
                CR;
                addnum( m_num, P_num, m_num);
        }

        two_num = stonum("2");
        while( turn++, signum( subnum( n_num, two_num, &tmp) ) > 0 )
        {
                if( ncand( N_num, &n_num, &m_num, P_num) ){
                        printf("¥"%d n=", turn);
                        prnum(n_num);
                        printf(" m=");
                        prnum(m_num);
                        CR;
                        addnum( m_num, P_num, m_num );
                }
        }
}

/* end of t_k.c */
```

```
/*
        file name  t_b.c

          npddec.c(by H. Deguchi)を改良したプログラム
*/

#include        <stdio.h>
#include        <math.h>
#include        "mlib.h"


extern  Num     get_k();
extern  int     fcand();
extern  int     ncand();
extern  int     decomp();

static  int     turn = 0;

main( argc, argv )
 int    argc;
 char   *argv[];
{
        Num     N_num, n_num, m_num, z_num, P_num;
        char    buf[255];

        setbuf(stdout,NULL);    /* no bufferring */

/* N */
        if( argc > 1 )
                N_num = stonum( argv[1] );
        else
                N_num = stonum("1315753");
        z_num = stonum("1");
        while(1)
          {
                scanf("%s", buf );
                if( strcmp( buf, "n" ) == 0 )
                {
                        scanf("%s", buf );
                        copynum( stonum(buf), N_num );
                }
                else
                        break;
        }

        P_num = stonum("1");

        n_num = m_num = Newnum;

        printf("XUnitText: x=t¥n");
        printf("YUnitText: y=distance¥n");
        printf("Device: Postscript¥n");
        printf("Disposition: To Device¥n");
        printf("FileOrDev: lpr -Pimagen¥n");
        printf("ZeroColor: black¥n");

        decomp( N_num, n_num, m_num, z_num,P_num );

        printf("TitleText: P=1 N=");
        prnum(N_num);CR;

}

/* axb = a * x - b */
Num axbnum( a_num, x_num, b_num, axb_num )
 Num a_num, x_num, b_num, axb_num;
{
static  NUM     tmp;

        return( subnum( mulnum(a_num, x_num, &tmp), b_num, axb_num ) );
}

/* get the first candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        m / n < z
        return 1: if found, 0: otherwise
*/
int fcand( N_num, n_num, m_num, z_num, P_num )
 Num    N_num, *n_num, *m_num, z_num, P_num;
{
static  NUM     nm_num = NUM_0; /* n * m */
static  NUM     mP_num = NUM_0; /* m - P */
static  NUM     tmp = NUM_0;    /* tmp */

        *n_num = sqrtnum( divnum(N_num,z_num,&tmp), *n_num );
        if( signum(fraction(*n_num,&tmp)) == 0 )
        {
                *m_num = divnum( N_num, *n_num, *m_num );
                        return( 1 );
        }
        ceilingnum( *n_num, *n_num );
        subnum( *n_num, P_num, *n_num );
        *m_num = divnum( N_num, *n_num, *m_num );
        ceilingnum( *m_num, *m_num );
        mulnum( *n_num, *m_num, &nm_num );
        if( compnum( &nm_num, N_num ) < 0 )
```

```
                panic("fcand error : n * m < N¥n");
        subnum( *m_num, P_num, &mP_num );
        mulnum( *n_num, &mP_num, &nm_num );
        if( compnum( &nm_num, N_num ) >= 0 )
                panic("fcand error : n * (m-P) >= N¥n");
        return( 0 );
}

/* k = (-(m-b*n)*sqrt((m-b*n)^2+4*b*(n*m-N)))/2*b */
Num get_k( N_num, n_num, m_num, beta_num, k_num, P_num )
 Num    N_num, n_num, m_num, beta_num, k_num, P_num;
{
static  NUM     mbn_num = NUM_0;        /* m-b*n */
static  NUM     mbn2_num = NUM_0;       /* (m-b*n)^2 */
static  NUM     nmN_num = NUM_0;        /* n*m-N */
static  NUM     tmp1 = NUM_0;           /* for working */
static  NUM     tmp2 = NUM_0;           /* for working */
static  NUM     tmp3 = NUM_0;           /* for working */

        subnum( m_num, mulnum( beta_num, n_num, &tmp1 ), &mbn_num );
        mulnum( &mbn_num, &mbn_num, &mbn2_num );
        axbnum( n_num, m_num, N_num, &nmN_num );
        mulnum( mulnum( Four, beta_num, &tmp1), &nmN_num, &tmp2 );
        sqrtnum( addnum( &mbn2_num, &tmp2, &tmp1 ), &tmp2 );
        subnum( &tmp2, &mbn_num, &tmp1 );
        divnum( &tmp1, mulnum( mulnum( Two, P_num, &tmp3 ), beta_num, &tmp2 ),
                k_num );
        return( k_num );
}

/* get the next candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        return 1: if found, 0: otherwise
*/
int ncand( N_num, n_num, m_num, P_num )
 Num    N_num, *n_num, *m_num, P_num;
{
static  NUM     beta_num = NUM_0;       /* beta = f[m/n] */
static  NUM     k_num = NUM_0;          /* k */
static  NUM     nmN_num = NUM_0;        /* n * m - N */
static  NUM     tmp1 = NUM_0;
static  NUM     tmp2 = NUM_0;
        int     i;
        int     fk;

        /* output t */
        printf("%d ",turn);

        floornum( divnum( *m_num, *n_num, &tmp1 ), &beta_num );
        prnum( &beta_num ); CR;

        get_k( N_num, *n_num, *m_num, &beta_num, &k_num, P_num );
        fk = signum( fraction( &k_num, &tmp1 ) );
        ceilingnum( &k_num, &k_num );

        /* n = n -  k */
        subnum( *n_num, mulnum( P_num, &k_num, &tmp1 ), *n_num );

        /* m = m + P * b * k */
        mulnum( mulnum( P_num, &beta_num, &tmp2), &k_num, &tmp1 );
        addnum( *m_num, &tmp1, *m_num );
        if( fk == 0 )
        {
                return( 1 );
        }
        /* fk != 0 */
        for( i = 0;
                (fk = signum( axbnum( *n_num, *m_num, N_num, &nmN_num ) )) < 0;
                        i++ )
        {
                if( i > 2 )
                {
                        /*i = -1;*/
                        copynum( Zero, &tmp1 );
                        divnum( N_num, *n_num, &tmp1 );
                        ceilingnum( &tmp1, *m_num );
                }
                else
                {
                        addnum( *m_num, P_num, *m_num );
                }
        }
        if( fk == 0 )
        {
                return( 1 );
        }
        else
        {
                return( 0 );    }
}

int decomp( N_num, n_num, m_num, z_num, P_num )
 Num    N_num, n_num, m_num, z_num, P_num;
{
        static  NUM     tmp;
        static  Num     two_num;

        if( fcand( N_num, &n_num, &m_num, z_num, P_num ) ){
```

```
                printf("¥"%d n=", turn);
                prnum(n_num);
                printf(" m=");
                prnum(m_num);CR;
                addnum( m_num, P_num, m_num);
        }

        two_num = stonum("2");
        while( turn++, signum( subnum( n_num, two_num, &tmp) ) > 0 )
        {
                if( ncand( N_num, &n_num, &m_num, P_num) ){
                        printf("¥"%d n=", turn);
                        prnum(n_num);
                        printf(" m=");
                        prnum(m_num);CR;
                        addnum( m_num, P_num, m_num );
                }
        }
}

/* end of t_b.c */
```

```
/*
        file name Pt_distance.c
             npddec.c(by H. Deguchi)を改良したプログラム
*/

#include        <stdio.h>
#include        <math.h>
#include        "mlib.h"


extern  Num     get_k();
extern  int     fcand();
extern  int     ncand();
extern  int     decomp();

static  int     turn = 0;

main( argc, argv )
 int    argc;
 char   *argv[];
{
        Num     N_num, n_num, m_num, z_num, P_num;
        char    buf[255];

        setbuf(stdout,NULL);    /* no bufferring */

/* N */
        if( argc > 1 )
                N_num = stonum( argv[1] );
        else
                N_num = stonum("1315753");
        z_num = stonum("1");
        while(1)
          {
                scanf("%s", buf );
                if( strcmp( buf, "n" ) == 0 )
                {
                        scanf("%s", buf );
                        copynum( stonum(buf), N_num );
                }
                else
                        break;
        }

        P_num = stonum("1");

        n_num = m_num = Newnum;

        printf("XUnitText: x=t¥n");
        printf("YUnitText: y=distance¥n");
        printf("Device: Postscript¥n");
        printf("Disposition: To Device¥n");
        printf("FileOrDev: lpr -Pimagen¥n");
        printf("ZeroColor: black¥n");
        printf("TitleText: P=");
        prnum(P_num);
        printf("N=");
        prnum(N_num);CR;

        decomp( N_num, n_num, m_num, z_num,P_num );

}

/* axb = a * x - b */
Num axbnum( a_num, x_num, b_num, axb_num )
 Num a_num, x_num, b_num, axb_num;
{
static  NUM     tmp;

        return( subnum( mulnum(a_num, x_num, &tmp), b_num, axb_num ) );
}

/* get the first candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        m / n < z
        return 1: if found, 0: otherwise
*/
int fcand( N_num, n_num, m_num, z_num, P_num )
 Num    N_num, *n_num, *m_num, z_num, P_num;
{
static  NUM     nm_num = NUM_0; /* n * m */
static  NUM     mP_num = NUM_0; /* m - P */
static  NUM     tmp = NUM_0;    /* tmp */

        *n_num = sqrtnum( divnum(N_num,z_num,&tmp), *n_num );
        if( signum(fraction(*n_num,&tmp)) == 0 )
        {
                *m_num = divnum( N_num, *n_num, *m_num );
                        return( 1 );
        }
        ceilingnum( *n_num, *n_num );
        subnum( *n_num, P_num, *n_num );
        *m_num = divnum( N_num, *n_num, *m_num );
        ceilingnum( *m_num, *m_num );
        mulnum( *n_num, *m_num, &nm_num );
        if( compnum( &nm_num, N_num ) < 0 )
```

```
                panic("fcand error : n * m < N¥n");
        subnum( *m_num, P_num, &mP_num );
        mulnum( *n_num, &mP_num, &nm_num );
        if( compnum( &nm_num, N_num ) >= 0 )
                panic("fcand error : n * (m-P) >= N¥n");
        return( 0 );
}

/* k = (-(m-b*n)*sqrt((m-b*n)^2+4*b*(n*m-N)))/2*b */
Num get_k( N_num, n_num, m_num, beta_num, k_num, P_num )
 Num    N_num, n_num, m_num, beta_num, k_num, P_num;
{
static  NUM     mbn_num = NUM_0;        /* m-b*n */
static  NUM     mbn2_num = NUM_0;       /* (m-b*n)^2 */
static  NUM     nmN_num = NUM_0;        /* n*m-N */
static  NUM     tmp1 = NUM_0;           /* for working */
static  NUM     tmp2 = NUM_0;           /* for working */
static  NUM     tmp3 = NUM_0;           /* for working */

        subnum( m_num, mulnum( beta_num, n_num, &tmp1 ), &mbn_num );
        mulnum( &mbn_num, &mbn_num, &mbn2_num );
        axbnum( n_num, m_num, N_num, &nmN_num );
        mulnum( mulnum( Four, beta_num, &tmp1), &nmN_num, &tmp2 );
        sqrtnum( addnum( &mbn2_num, &tmp2, &tmp1 ), &tmp2 );
        subnum( &tmp2, &mbn_num, &tmp1 );
        divnum( &tmp1, mulnum( mulnum( Two, P_num, &tmp3 ), beta_num, &tmp2 ),
                k_num );
        return( k_num );
}

/* get the next candidate pair as n_num and m_num */
/*      N: N_num, n: *n_num, m: *m_num
        n * m >= N && n * (m-1) < N
        return 1: if found, 0: otherwise
*/
int ncand( N_num, n_num, m_num, P_num )
 Num    N_num, *n_num, *m_num, P_num;
{
static  NUM     beta_num = NUM_0;       /* beta = f[m/n] */
static  NUM     k_num = NUM_0;          /* k */
static  NUM     nmN_num = NUM_0;        /* n * m - N */
static  NUM     tmp1 = NUM_0;
static  NUM     tmp2 = NUM_0;
        int     i;
        int     fk;

        /* output t */
        printf("t=%d",turn);

        floornum( divnum( *m_num, *n_num, &tmp1 ), &beta_num );
        printf(" P=");
        prnum( P_num );
        printf(" n=");
        prnum( *n_num );

        printf(" m=");
        prnum( *m_num );

        divnum( N_num, *n_num, &tmp1 );
        subnum( P_num, subnum( *m_num, &tmp1, &tmp2 ), &tmp1 );
        printf(" D=");
        prnum( &tmp1 ); CR;

        get_k( N_num, *n_num, *m_num, &beta_num, &k_num, P_num );
        fk = signum( fraction( &k_num, &tmp1 ) );
        ceilingnum( &k_num, &k_num );

        /* n = n -  k */
        subnum( *n_num, mulnum( P_num, &k_num, &tmp1 ), *n_num );

        /* m = m + P * b * k */
        mulnum( mulnum( P_num, &beta_num, &tmp2), &k_num, &tmp1 );
        addnum( *m_num, &tmp1, *m_num );
        if( fk == 0 )
        {
                return( 1 );
        }
        /* fk != 0 */
        for( i = 0;
                (fk = signum( axbnum( *n_num, *m_num, N_num, &nmN_num ) )) < 0;
                        i++ )
        {
                if( i > 2 )
                {
                        /*i = -1;*/
                        copynum( Zero, &tmp1 );
                        divnum( N_num, *n_num, &tmp1 );
                        ceilingnum( &tmp1, *m_num );
                }
                else
                {
                        addnum( *m_num, P_num, *m_num );
                }
        }
        if( fk == 0 )
        {
                return( 1 );
        }
        else
```

```
        {
                return( 0 );    }
}

int decomp( N_num, n_num, m_num, z_num, P_num )
 Num    N_num, n_num, m_num, z_num, P_num;
{
        static  NUM     tmp;
        static  Num     two_num;

        if( fcand( N_num, &n_num, &m_num, z_num, P_num ) ){
                printf("¥"%d n=", turn);
                prnum(n_num);
                printf(" m=");
                prnum(m_num);CR;
                addnum( m_num, P_num, m_num);
        }

        two_num = stonum("2");
        while( turn++, signum( subnum( n_num, two_num, &tmp) ) > 0 )
        {
                if( ncand( N_num, &n_num, &m_num, P_num) ){
                        printf("¥"%d n=", turn);
                        prnum(n_num);
                        printf(" m=");
                        prnum(m_num);CR;
                        addnum( m_num, P_num, m_num );
                }
        }
}

/* end of Pt_distance.c */
```

```
/*      file name min.c
        Xgraph用データの為のプログラム
*/

#include        <stdio.h>
#include        <math.h>
#include        <strings.h>
#include        "mlib.h"

FILE *stream;

main( argc, argv )
int argc;
char *argv[];
{
        int an,an2;
        char t[5000],distance[500];
        char t2[5000],distance2[500];
        char t3[5000],distance3[500];


        if (( stream = fopen( argv[1], "r" )) == NULL){
                printf("error file");
                exit(1);
        }

        while( fscanf( stream, "%s",t ) != EOF ){
                if( strcmp( t, "1" ) == 0 ){
                        break;
                }
        }

        fscanf( stream, "%s",distance);
        fscanf( stream, "%s%s", t2 , distance2 );

        while( fscanf( stream, "%s%s", t3 , distance3 ) != EOF ){
                if( strncmp( t, "¥"",1 ) == 0 ){
                        fscanf(stream,"%s",t);
                        continue;
                }
                if( strcmp( distance2, distance) < 0 && strcmp( distance3, distance2) > 0)
                        printf("%s %s¥n",t2,distance2);
                strcpy( t, t2);
                strcpy( distance, distance2);
                strcpy( t2, t3);
                strcpy( distance2, distance3);
        }
        fclose(stream);
}

/* end of min.c */
```

```
/*      file name min2.c
          数字データ用プログラム
*/

#include        <stdio.h>
#include        <math.h>
#include        <strings.h>
#include        "mlib.h"

FILE *stream;

main( argc, argv )
int argc;
char *argv[];
{
        int an,an2;
        char t[5000],distance[500];
        char t2[5000],distance2[500];
        char t3[5000],distance3[500];


        if (( stream = fopen( argv[1], "r" )) == NULL){
                printf("error file");
                exit(1);
        }


        fscanf( stream, "%s%s", t, distance);
        fscanf( stream, "%s%s", t2 , distance2 );

        while( fscanf( stream, "%s%s", t3 , distance3 ) != EOF ){
                if( strncmp( t, "¥"",1 ) == 0 ){
                        fscanf(stream,"%s",t);
                        continue;
                }
                if( strcmp( distance2, distance) < 0 && strcmp( distance3, distance2) > 0)
                        printf("%s %s¥n",t2,distance2);
                strcpy( t, t2);
                strcpy( distance, distance2);
                strcpy( t2, t3);
                strcpy( distance2, distance3);
        }
        fclose(stream);
}

/* end of min2.c */
```

```
/*      file name  kankaku.c       */

#include        <stdio.h>
#include        <math.h>
#include        <strings.h>
#include        "mlib.h"

FILE *stream;

main( argc, argv )
int argc;
char *argv[];
{
        int count;
        int it, it2;
        char t[5000],distance[500];
        char t2[5000],distance2[500];


        if (( stream = fopen( argv[1], "r" )) == NULL){
                printf("error file");
                exit(1);
        }


        fscanf( stream, "%s%s", t, distance);
        count = 1;
        while( fscanf( stream, "%s%s", t2 , distance2 ) != EOF ){
                it = atoi( t );
                it2 = atoi( t2 );
                printf("%d %d¥n",count,it2-it);
                count++;
                strcpy( t, t2);
        }
        fclose(stream);
}

/* end of kankaku.c */
```