MT 乱数を用いたサーチ行動のシミュレーション

永　星　浩　一

目　次

は　じ　め　に

サイバー市場における消費者のサーチ行動をシミュレーションするうえで，商品を探すためのキーワード検索自体をサーチとは見なさず，時間をずらして再度検索することをサーチととらえてシミュレーションによる分析を行うことの有益性を拙稿[1]において明らかにした。その結果，一定数の買手のサーチ開始時を束ねて「期」に分け，期のはじめに新たな売手の販売が開始されると設定して表計算のワークシート上でサーチ及び販売経過のシミュレーション結果を得ることができる。このシミュレーションによって，存在する全ての買手の提示価格を把握できる完全検索と一部しか把握できない不完全検索に分けてサーチの有効性について検証することができる。

1)『ネット市場における不確実性とサーチ』（永星〔3〕，2009）

本稿では，拙稿（2009）で用いたExcelのワークシート関数Rand() あるいはVBAのRnd関数を用いず，メルセンヌ・ツイスター（以降MT）乱数を用い，また期間を100から200に拡張し，売手数は100から500へ，買手数は1000から5000へ増やしてシミュレーションを行う。MT乱数は1997年に松本眞氏と西村拓士氏によって開発された高速・高品質の擬似乱数生成法である。乱数はある確率分布をもち，周期性も再現性も無い数列のことであるが，擬似乱数は，アルゴリズムを用いてコンピュータで発生させるものであり，周期性と再現性がある。擬似乱数と称されるゆえんである。再現性は，同じSeed（種）を与えることで全く同じ乱数を発生させることができ，特定のシミュレーションを追試することを可能にする。半面，周期性はシミュレーション結果の信頼性を損ねる可能性がある。シミュレーションで使用する乱数が周期の範囲に収まっていれば問題ないとされる。Excelで用いられるワークシート関数Rand() やVBA関数のRnd() については，大規模に乱数を用いて行うシミュレーションには不向きである[2)]。再現性についても，ソフトを再起動するたびに同じ系列の乱数を発生させるが，Seedを与えて同じ系列の乱数を発生させるようには設計されていない。同じワークシートを読み込むExcelのバージョンが異なると異なる乱数列が現れ，バージョンが同じでも同じ乱数列が発生するとは限らない[3)]。シミュレーションの追試という点からは極めて扱いづらい乱数である。本稿のシミュレーションに用いる乱数は１回のシミュレーションについて，2,502,500個[4)]であり試行のたびにこの個数の乱数が消費されることになる。拙稿（2009）のシミュレーションでは１回の試行あたり2,200個程度の乱数[5)]であったので，試行回数を加味しても

2) Rndの周期は2^{24}＝16,777,216。
3) Excel2003は2004年2月29日の修正パッケージ適用以前は，10％の確率で負の数を出力するというバグがあった。
4) サーチが行われるケースのシミュレーションはこの個数を上回る。検索ミスの確率事象に用いる乱数で，最低価格を確認後の試行における使用されず破棄される乱数も含む。

Rnd 関数の周期は問題とはならなかったが，今回，大幅に規模を拡大するにあたり，乱数の質的問題もクリアする必要性が生じた。また，同時に実行時間も急激に増大することが考えられるので，実行速度の高速化を図るためプログラムの見直しを行った。まず，MT 乱数のアルゴリズム自体高速であることと，同じ処理内容でもより高速な逐次処理ではない書式が存在すること，ワークシート関数が VBA による処理よりも高速であることなど総合的に改良を施し，実質的な 1 試行あたりの実行時間を15分程度に抑えることができた[6]。

1．MT 乱数

もっとも基本的な乱数は等確率性（等出現性）と無規則性が備わった一様分布で与えられるが，コンピュータで発生させる擬似乱数は，等確率性を実現できても無規則性は周期の観点から実現できない。しかし，この規則性はシミュレーションを行う上で Seed をそろえることによって再現性の確認が可能という好都合な性質でもあり，逆に周期という大規模に乱数を消費する場合には結果の信頼性を損なわせる性質でもある。MT 乱数は，周期が 2^{19937}-1と長く，10進表記であれば6,000桁以上[7]であるので，本稿の分析対象であるネット市場における検索，サーチ行動のシミュレーションには十分である。

本稿は Excel のアドイン関数として開発された NtRand[8]を用いて MT 乱数を発生させ，シミュレーションに用いることとする。NtRand は Windows95，98，Me，2000，XP 上の Excel97，2000，XP で稼働するアドインであり，Seed

5) 留保価格の乱数 1,000 個を検索点の乱数を加工して用いることで節約している。
6) デュアルコアの CPU 速度 3GB レベル。シングルの 1GB レベルの場合，1 時間程度要した。なお，マルチスレッド計算対応バージョンによる比較。
7) Rnd は 8 桁。
8) 多変量モンテカルロ Excel アドイン関数（フリーウェア）Copyright©1998-2003 Numerical Technologies Incorporated.

に関して２つの引数を指定でき，ある Seed のペアに対して32,767個の一様乱数を発生させ，表の行方向ないし列方向に出力させることができる。本稿では NtRand の乱数アルゴリズムのうち，2002年１月26日の初期化ルーチン改良版の 53bit 精度（０，１）開区間実数表現の Algorithm＝０で MT 乱数を発生させる。また，シミュレーションに用いた OS は WindowsXP SP3，CPU は AMD Athron64 X2 DualCore 4400＋である。OS ならびに CPU を明確にするのは，数値演算の結果がこれらの要素に影響を受ける可能性があるからである[9]。NtRand が２種類の Seed を指定できるのは好都合である。それは，Seed1 を変化させることで１回のシミュレーションに必要な乱数を発生させることができ，Seed2 を選択的に変化させることで，Seed1 を固定することで（価格分布，販売個数，販売開始点，検索点，留保価格，検索ミスパターン）のいずれかを固定したまま，他の数値を変化させて試行を行わせる VBA の記述を平易にできるからである。

高速化からの観点からは Excel の再計算や VBA を用いる事自体が不利な点であるが，検索やサーチが行われる様子が，売手の売れ残り商品数の形で，時系列で変化するのが視覚的に表現できることと，集計結果をグラフで表現でき，不規則な変化や傾向などを読み取ることが容易であること利点が数多

9) NtRand の開発元である Numerical Tchnologies 社の公開文書（Released 1/6/2003）によると，計算結果が異なるのは CPU 内部の浮動小数点演算ユニット（FPU）の動作モードと CPU チップ内部の ROM にハードコーディングされたマイクロプログラム（micro program）にその原因があり，前者については double 型では Intel x86 系の FPU の仮数部の精度は 24/53/64bit に「プログラムを使って」設定可能であり，その設定は OS やコンパイラレベルで普通は規定されていること。さらに x86 系では Pentium4/Xeon になって以降，128bit，つまり double 型２個を一度に処理する SIMD 最適化（SSE II）ができることも精度に影響するとある。また，後者についても CPU が異なればマイクロプログラムのアルゴリズムも異なり，問題によっては「Linux，Windows，さまざまな商用 UNIX でバラバラの答えが出る」といったことが普通に起こるとある。これらの仕様は PC メーカーによってディスクローズされておらず，他にも積極的に知らされていない問題が数多くあることを指摘し，その上で，金融工学で PC を使用する者たちに対して，「コンピュータは文字通り魔物（deamon）の住む機械」であると警告している。

く存在する。

2．サーチの定義と基本モデル

電子商取引 BtoC において，消費者はキーワードをキーボードで入力して検索ボタンをクリックするだけで直接目的の商品にアクセスすることができる。これをキーワード検索（以降単純に「検索」）と呼ぶ。ネットでは無数の商品が販売されており，それらすべてが統一的なポータル[10]からカテゴリー分類を辿って行き着けるわけではない。キーワード検索なくして最適な商品を見つけ出すことは不可能であり，キーワード検索こそ現代の電子商取引の核となるサービスである。

キーワード検索は，従来の消費者による買い回り（サーチ）行動のコスト０ないし０に近い特殊なケースととらえるのが普通であろう。しかし，本稿ではキーワード検索それ自体をサーチとはとらえない。キーワード検索によって留保価格を下回る商品が見つからない場合，時期をずらして再度キーワード検索を行う行為をサーチととらえる。これは例えて言うならば，高台から望遠鏡で店の提示価格を覗くことをキーワード検索，そのときたまたま望遠鏡を向けそこなったとか，角度が悪く見えない店の提示価格を，時間をずらしたり別な高台に移動したりして，再度望遠鏡で覗いて確認する行動をサーチととらえることに似ている。リアルな市場におけるサーチと同様，時間や手間にかかわるコストが生じるのがサーチであり，完全情報市場となるのは特殊なケースに限られる。

一度のキーワード検索によってすべての選択肢の中から最適な商品を見つけだすことができ，かつ買手はそれが最適であることを知っている場合を「完全検索」と呼ぶ。この「最適」とはその検索時点における「最適」であり，

10) 玄関口となる Web サイト。検索システムやカテゴリー分類などによって目的の情報に辿り着ける仕組みを提供している。

費用をかけて次期に再検索（すなわちサーチ）したとき，その費用を上回る利益があるようなケースも含めての「最適」ではない。したがって，「完全検索」は，時間に関する不確実性は考慮されていないことになる。不完全検索は，一部の選択肢が検索に引っかからないことをいう。本稿のシミュレーションでは確率的に検索漏れとなる選択肢を決めるので，結果的にすべての選択肢が検索されるケースも含んでいる。その意味で，完全検索は不完全検索の特殊ケースということができる[11]。

完全検索にせよ，不完全検索にせよ，留保価格 P_r 以下の提示価格がない場合，次の期に再度検索を行うことになる。この再検索をサーチと呼ぶ。拙稿（2009）において，（０，１］[12]の留保価格と現時点での売手数をかけて１以上であれば[13]，次期において留保価格以下の価格を見つける見込みがあるものとして，次期における再検索を行うように設計した。（すなわち１回サーチが追加されることになる。）サーチ後の再検索で，留保価格から「サーチコスト」を差し引くことで，ハードルを高くできるようにもプログラムした。

しかし，本稿では留保価格はサーチコストが織り込まれているものとし，留保価格は変化させない[14]。次なる期に再検索を行うことで知りうる期待最低価格が留保価格を下回ればサーチを行うし，そうでなければサーチは行わ

11) 検索漏れ率 0% のケース。
12) Rnd 関数で発生する乱数は半開区間である。
13) 例えば，留保価格が 0.5 で，次期の売手が 3 名いるとすると，0.5×3＝1.5（≧1）であり，「見込みがある」と考える。もっとも，完全検索の場合，留保価格を上回ることが分かっている今期の売手の総数を基準に，期待値を計算することが不自然である。次期において全ての売手が入れ替わる訳ではないからである。
14) 留保価格は，サーチコストを織り込み済みであり，特にサーチコストを引く必要はないという考え方も成り立つが，その場合は，「サーチコスト」を 0 とすればよい。本稿のプログラムでは手順の簡略化のため，「サーチコスト」の処理を削除する。ただ，留保価格の変化自体は，例えばサーチしてもなかなか留保価格以下の価格を見いだせない場合，ハードルが下がる，すなわち留保価格が上昇することなどへの応用も考えられる。この場合はマイナスのサーチコストでカバーできる。

れない。一期あたりの販売を開始する新規の売手の数を n とする。買手がリスクニュートラルであると考えることで，この新たに販売を開始する n 人の売手を基準にして，ある留保価格 P_r に対して，$P_r(n+1)\geq 1$ のとき留保価格以下の売手を見つけることができると考え[15]，サーチを行うことにする。

具体例で計算すると，200期ある中で500人の売手の中で次期において販売を開始する売手数は平均的に2.5人である。$3.5P_r$ が１以下であればサーチが行われる[16]。次なる期に検索を行うことで留保価格以下の売手を発見できれば，商品が購入されサーチは終わるし，発見できなければ，その次の期も再々検索（サーチ）を行うことになる[17]。サーチは留保価格以下の価格の商品が見つかるまで，それ以降の期も続けられる。

本稿における基本モデルの仮定を以下にまとめる。

⑴　多数の売手[18]が，（０，１）の範囲[19]で価格を付け，ある期に販売を開始し，売り切れるまで，販売価格を下げることなく販売を続ける。

⑵　売手は，販売を開始する期を事前にランダムに決めているものとする。買手の数や相場価格などによって戦略的に売り出し期を決めることはない。

⑶　買手は，期間中のある時点で検索を行い，検索によって確認された売手のうち最も安い価格が留保価格を下回るとき，そこから購入する。

15) 留保価格が P_r で，次期の売手が n 名いるとすると，価格が（0，1）の一様分布と仮定されているので，期待最低価格は $1/(n+1)$ である。$P_r=0.4$，$n=2$ とすると，期待最低価格は 1/3 である。$P_r=0.4\geqq 1/3$（＝期待最低価格）となり，サーチによって，留保価格以下の売手が期待できることになる。これより，$P_r(n+1)\geqq 1$ であればサーチの利益があると考えられる。

16) 逆にいうと，P_r が 1/3.5（≒0.286）以下の買手はサーチを行わないことになる。

17) 新たに販売を開始する売手の平均的な数はそれ以後の期においても変化しなければ，期待最低価格はそれ以後の期においても留保価格を下回ることになる。

18) 無数ではなく，現実的な多数を意味する。

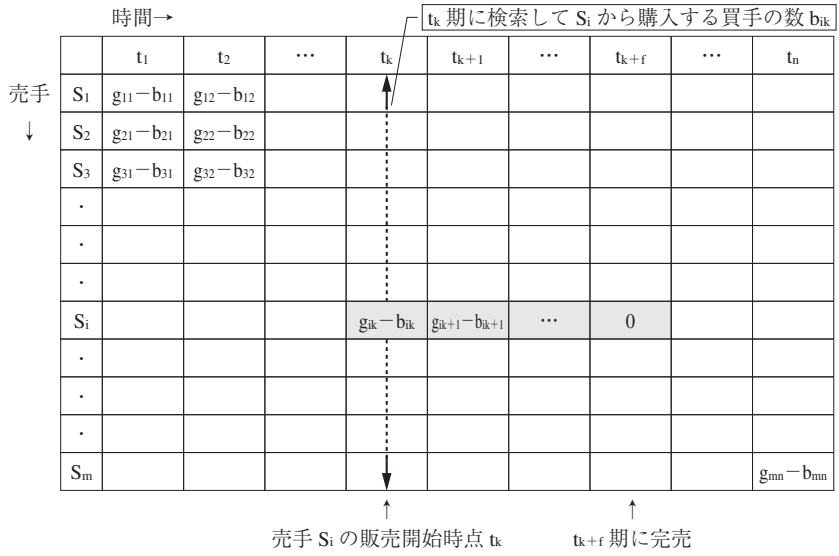
19) NtRand アドインによって発生させる MT 乱数は開区間（0，1）の 53bit 精度開区間の実数表現の擬似乱数である。

品質保証や送料の違いなどの差異はすべて価格に表されているものとする。

⑷ 買手の留保価格は（0，1）の範囲の一様分布とする。当期に留保価格を下回らなければ，次期の期待最低価格が留保価格以下のときサーチを行う。さもなければ購入を諦める。

既に述べたように，売手を500，販売可能な期間は第1期から第200期まであるとし，各売手の販売数量を10と固定する。したがって総販売個数は5,000個である。各期には平均して2.5前後の売手が存在[20]している計算になるが，検索した結果1人売手も確認できなかった買手は購入しないものとする。

図表2－1　検索・購入シミュレーションのワークシート設計[21]



20) 売手総数500を期間数200で割った数。

21) ワークシートの基本設計は『ネット市場における不確実性とサーチ』（永星〔3〕，2009）で用いたものと同じである。

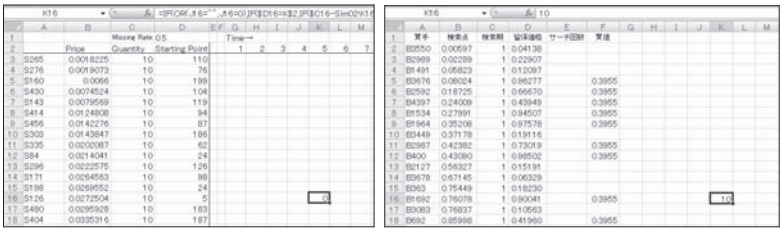
図表２－１は時間の流れを考慮した検索・購入シミュレーションのワークシートである。売手は行に並べ，時間の流れを列方向にとる。買手数は5,000で，MT 乱数を基に（０，200）の実数表現の一様分布で発生させた数（検索点）を基に並べられる。さらに整数部の数字＋１を検索「期」とする。各期に現れる買手の平均数は25である。その並びは検索点に関して昇順になっている。買手はある時点で検索を行い最も安い価格が留保価格を下回るとき購入する。図表２－１では，売手 S_i の販売開始時点は t_k，販売数量 g_{ik} であり，t_k 以前は販売していない状態である。t_k 時点ではじめて買手の検索対象となり，t_k 時点に検索された中で最も安い価格でかつ買手の留保価格を下回るとき，購入してくれる客を獲得することになる。その客の合計数が b_{ik} で表され，結果として，t_k 期における店頭在庫は $g_{ik}-b_{ik}$ ということになる。その期の買手による検索が終了した時点で残る店頭在庫は，次の期 t_k+1 に繰り越され，同じ値段で再度販売され，t_k+1 期に検索する買手の検索の対象となる。こうして店頭在庫が０になるまで在庫の繰り越しが行われることになる。最終的に，潜在的売手がこの期間 t_l～t_n に販売可能な売手として現れる期間が決まる。売手 S_i の販売期間は図表２－１の網掛けで表された期間である。

３．シミュレーション例

シミュレーションは Excel の VBA（マクロ）を用いる。最初のワークシート（名前を Sim01 とする）を基本に，価格の低い順にソートされた売手が並び，販売開始とともに販売数量（ここでは10個）がセルに入る。第２のワークシート（Sim02）には検索点の早い順に買手が並び，検索点の小さい順に「期」の順番に検索を行う。買手は検索の結果としての最低価格（Sim01 の販売数量を行方向に検索して最初に数字がある売手）が留保価格を下回るとき購入し，Sim02 の当該期のセルに購入者として加算される。Sim01 の当該

図表３－１　ワークシートの連携による販売ー購入シミュレーション

（Sim01）　　（Sim02）



期の販売数は，Sim02 の当該期の購入者数が引かれた結果が表示される。結果として売れ残った場合の販売数は次期に繰り越されることになる。この試行が，すべての買手について最終期まで繰り返される[22]。

ここで，売手は販売価格について昇順にソートされているが，ソートすることでプログラムの実行が早くなる一方[23]，ソートすることで結果が変わることはない。

図表３－１は，前述のワークシート Sim01 と Sim02 の一部である。K16 セルは，第５期から販売を開始する売手126（販売価格，約0.027）が当該期において完売している（Sim02 に10人の買手が付き，Sim01 の販売個数が０になっている）ことが示されている。

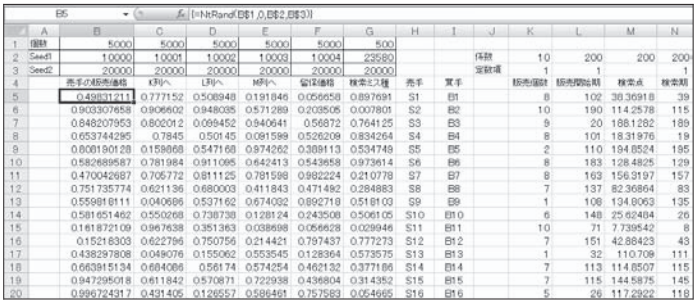
一方，Sim01 および Sim02 で用いられる乱数は，別のワークシート（MT_Rand）で発生させている。数式バーにあるように，NtRand は，MT 乱数の個数，Seed1，Seed2 と３つの引数を持つ配列関数[24]である。Seed1 を変えることで異なる乱数列が得られるので，これらを販売価格や販売開始期，検索

22) プログラムは末尾の資料参照

23) もちろん，ソートせずにシミュレーションを行うことは可能であるが，その場合は，最小価格を検索するプログラムが複雑になり，処理回数が増えることによりシミュレーションの実行時間が余分にかかることになる。また，グラフ化する際に価格帯別に分けることが困難になる。

24) 数式バー行末でクリック後，Ctrl＋Shift＋Enter で一気に確定する。

図表３－２　MT 乱数を発生させるワークシート（MT_Rand）



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	個数	5000	5000	5000	5000	5000	500							
2	Seed1	1.0000	1.0001	1.0002	1.0003	1.0004	23580			休憩	10	200	200	200
3	Seed2	20000	20000	20000	20000	20000	20000			定数項	1	1		1
4		売手の販売価格	x列へ	L列へ	M列へ	留保価格	検索ミス確	売手	買手		販売個数	販売開始期	検索点	検索期
5		0.498311211	0.777152	0.508948	0.191846	0.056658	0.897691	S1	B1		8	102	36.36918	39
6		0.903307658	0.906602	0.948035	0.571289	0.209505	0.007805	S2	B2		10	190	114.2578	115
7		0.848207953	0.802012	0.699452	0.940641	0.56872	0.764125	S3	B3		9	20	188.1282	189
8		0.653744295	0.7845	0.50145	0.091599	0.526209	0.834264	S4	B4		8	101	18.31976	19
9		0.800190128	0.159868	0.547168	0.974262	0.389113	0.534749	S5	B5		2	110	194.8524	195
10		0.582689587	0.781984	0.911095	0.642413	0.543658	0.973614	S6	B6		8	183	128.4825	129
11		0.470042687	0.705772	0.811125	0.781598	0.982224	0.210778	S7	B7		8	163	156.3197	157
12		0.751735774	0.621136	0.680003	0.411843	0.471482	0.284883	S8	B8		7	137	82.36864	83
13		0.559916111	0.040686	0.537162	0.674092	0.892718	0.518103	S9	B9		1	108	134.8063	135
14		0.581651462	0.550268	0.736738	0.128124	0.243508	0.506105	S10	B10		6	148	25.62484	26
15		0.161872109	0.967638	0.351363	0.038698	0.056628	0.029946	S11	B11		10	71	7.739542	8
16		0.15218303	0.622796	0.750756	0.214421	0.797437	0.777273	S12	B12		7	151	42.88423	43
17		0.438297808	0.049076	0.155062	0.553545	0.128364	0.573575	S13	B13		1	32	110.709	111
18		0.663915134	0.684086	0.56174	0.574254	0.462132	0.377186	S14	B14		7	113	114.8507	115
19		0.947295018	0.611842	0.570871	0.722938	0.436804	0.314352	S15	B15		7	115	144.5875	145
20		0.996724917	0.491405	0.126557	0.586461	0.757583	0.054665	S16	B16		5	26	117.2922	118

点，留保価格などに割り当てる。Seed2 は，シミュレーションを繰り返し行う際に異なる乱数列を発生させるのに用いる。乱数の Seed が２種類あることは，プログラムをわかりやすくできるという利点がある。

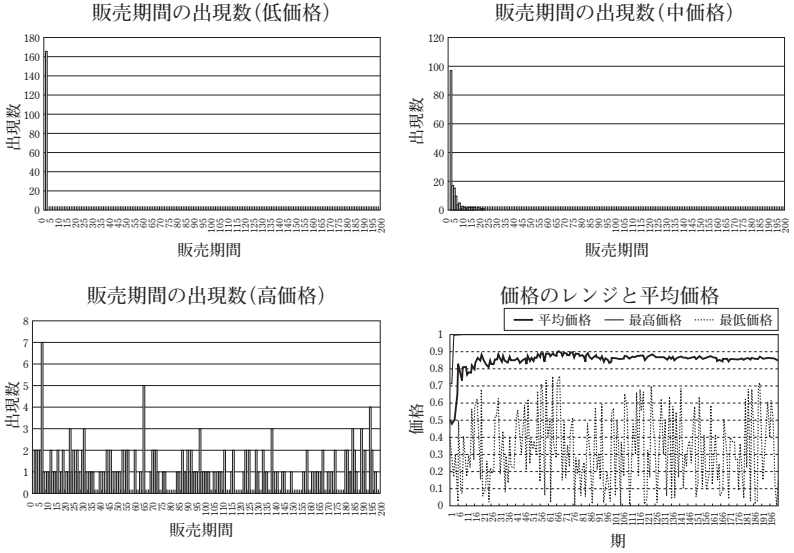
完全検索

完全検索は，「買手が存在する全ての売手を検索し比較できる」ことを意味する。したがって，販売中の最も上位に位置する[25]売手が留保価格との比較対象となる。図表３－１によると，第５期においては売手126が販売を開始し，当該期を検索期とする留保価格が0.0272504以上の‘先着’10名の買手が完全検索によって発見・購入したことを意味する。この売手126より上はより低価格の売手であるが全て空白で販売していないことが分かる。

完全検索のケースで，売手の販売期間の長短はどのようになっているのか，また販売中の商品価格のレンジはどのようになっているのか，グラフで表すことが容易にできるのも Excel を用いる利点の一つである。売手の販売価格を３等分し，低価格帯，中価格帯，高価格帯に分けて，販売期間別の売手数を表したのが図表３－３である。このシミュレーション結果は，図表３－２

25) ソートされているので最低価格の売手である。

図表３－３　完全検索の販売期間ごとの売手数（価格帯別）および価格レンジ



の MT 乱数に対する実行結果である。当然，高価格帯の売手はなかなか販売することができず[26]，長い期間にわたって販売を継続することになる[27]。

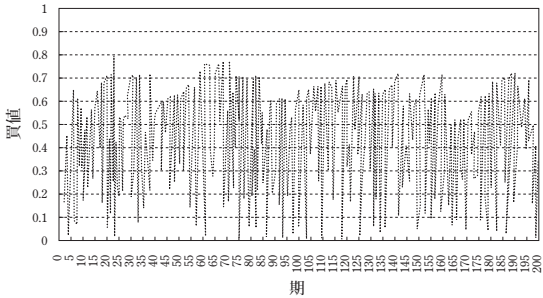
価格レンジにおける最低価格の推移は，各期における検索結果にほぼ一致する。検索結果が切れ切れになっているのは，存在した最低価格と購入に至った最低価格との差であり，留保価格よりも高く購入に至らなかったケースが影響している。

図表３－４より，このシミュレーションの設定の範囲で言えることは，検索結果が検索期による当たり外れが大きく，留保価格の範囲で安く買えるか

26) 高い価格にもかかわらず，比較的短期間に売り切ってしまう売手の存在などについては永星〔3〕（2009）参照。

27) 高価格帯の売手のデータは，途中から販売を開始し，最終期（第 200 期）まで売れ残ったものも含んでおり，200 期までという期間の制約が結果的に短い期間のデータとして表れている。本来はもっと長期に偏った形をしている。

図表３－４　販売期間別の検索結果（購入価格）



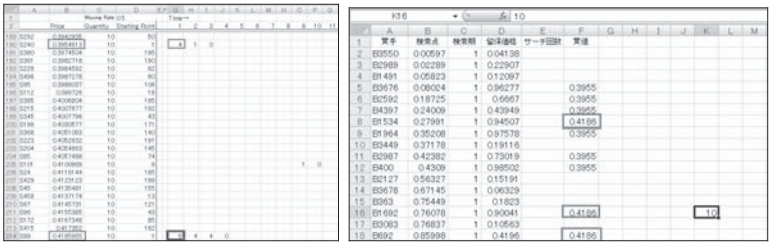
高いものを掴まされるのかは時の運次第という結果になっている。

不完全検索

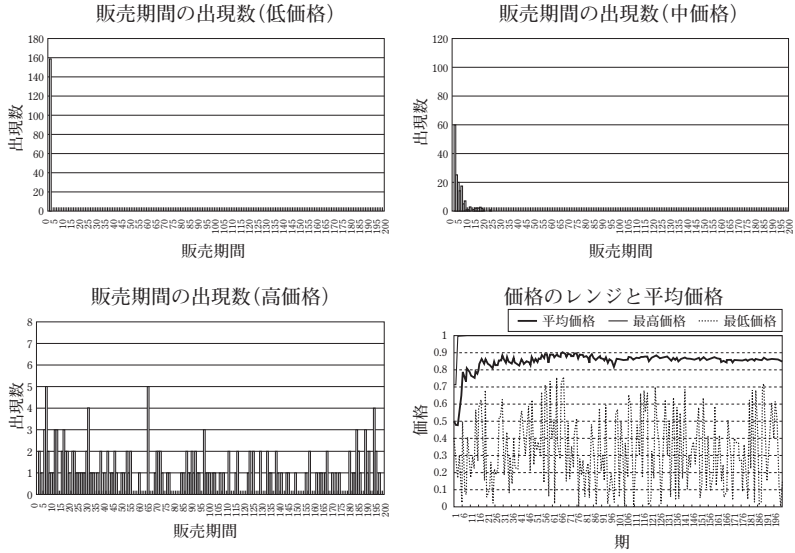
不完全検索は，「一定の割合で買手が存在する売手の一部を認知できない」検索を意味する。したがって，販売中の最も低価格の売手であっても，認知できなければ購入することができないことになり，次いで低価格の販売中の売手が認知できるなら留保価格との比較対象となる。検索ミスは完全検索の図表３－１と図表３－５を比較するとその特徴が分かる。第１期における最低価格は売手240で価格は0.3955あるが，買手1534，1692および692が２番目に安い売手99から0.4186で購入している。このように，不完全検索の場合，同じ期であっても先に検索する方が必ずしも有利になるとは限らない。

不完全検索のケースの販売期間ごとの売手数ならびに価格レンジは図表３－５で示された通りで完全検索のケースと大差ない。短期間の売り切れのケースが若干減り，それに伴い最低価格に期のシフトが若干見られる程度である。完全検索のケースと明らかに異なる傾向を示しているのが各期における検索結果である。図表３－５で見られるように，期の範囲内においても検索結果が上下するため，検索結果の変化が大きくなっている。

図表３－５　不完全検索における検索ミス（Sim01，Sim02）

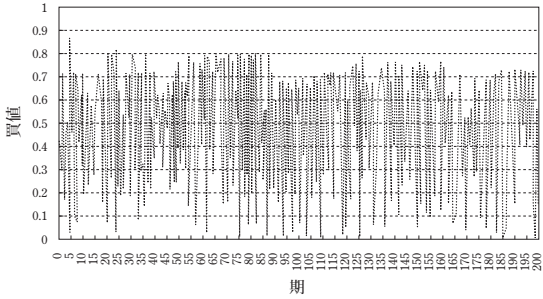


図表３－６　不完全検索の販売期間ごとの売手数（価格帯別）および価格レンジ



MT 乱数の Seed を合わせることで，これまで見てきた結果は，不完全検索の検索ミスの要素を除いた以外はすべて同じ条件での実験となっている。すなわち，図表３－２の乱数列によるシミュレーションの比較ということになる。問題は，これ以外の乱数列の場合どうかということである。これまで見てきた差異は，異なる乱数列を用いても同様の傾向を示すが，それ以外で

図表３－７　販売期間別の検索結果（購入価格）



図表３－８　完全検索(a)と不完全検索(b)の比較

	Seed2	消費者余剰		差	平均購入価格		差	非購入者数		差
		(a)	(b)		(a)	(b)		(a)	(b)	
1	20000	924.78053	901.5852	23.1953	0.37939492	0.38691956	−0.0075	1413	1344	69
2	20001	985.24593	975.49046	9.75547	0.36062637	0.36527148	−0.0046	1438	1387	51
3	20002	986.72204	989.86646	−3.1444	0.35702326	0.3585693	−0.0015	1322	1309	13
4	20003	919.40324	914.73716	4.66607	0.38599265	0.389651	−0.0037	1404	1371	33
5	20004	949.36222	941.17339	8.18882	0.36727945	0.37175001	−0.0045	1355	1312	43
6	20005	944.48365	931.17966	13.304	0.37717791	0.38199651	−0.0048	1415	1371	44
7	20006	973.62979	962.39479	11.235	0.36132167	0.36699072	−0.0057	1385	1337	48
8	20007	980.09098	968.87926	11.2117	0.36235269	0.36707248	−0.0047	1368	1329	39
9	20008	1031.1478	1015.7587	15.3891	0.33985137	0.34534395	−0.0055	1395	1346	49
10	20009	1019.4697	996.87485	22.5949	0.35290526	0.35995352	−0.007	1366	1297	69

完全検索と不完全検索の違いについて見ていこう。

図表３－８は留保価格から実際の購入価格を引き，全ての購入者について合計した消費者余剰について完全検索と不完全検索で比較したものである。

完全検索における消費者余剰が「消費者余剰１」，不完全検索のそれが「消費者余剰２」，それ以外に平均購入価格の差，購入することができなかった買手の数についてもカウントしている。シミュレーションは図表３－２の MT 乱数に関して，Seed2 を変化させつつ試行を繰り返した。図表３－８は

10種類の乱数列に関するシミュレーションとなっている。環境にもよるが，これだけで実行時間は４時間を要する[28]。実際には50のケースについての結果を得ており，ここで見られる傾向は全体を通して観察される。

図表３－８より，明らかに完全検索の方が不完全検索よりも消費者余剰が大きいことが分かる。平均購入価格が常に完全検索の方が低いことがその理由の一端を示しているが，購入に至らなかった消費者が完全検索の方が多いことも影響している可能性がある。すなわち，不完全検索の場合，必ずしも最低価格の売手が総取りされるとは限らないため，留保価格が比較的低い買手でも運良く条件に見合った売手を発見できる可能性が高くなる。逆に留保価格が比較的高い買手がそのことで購入の機会を失うことは少ない。なぜなら，遅からず条件に見合う売手を発見できる可能性は前者よりも高いからである。このことは，サーチの有効性を高める働きがある。

逆に留保価格が低く，サーチを行わない０－0.286の範囲に留保価格を持つ買手[29]の場合，検索ミスは，たった１回のチャンスを逃す可能性がある。実際，この範囲の消費者余剰が少ないことが確認できる。図表３－９は留保価格別の購入件数をグラフ化したものである。この例では，０－0.3の範囲の購入数は，検索ミス無しが245，検索ミス有りが224となっており，これ以外の Seed シミュレーションでも同様の傾向が確認できる。検索ミスが購入者数を増加させるのは，留保価格がサーチの行われる水準を若干上回る層においてサーチを行わない層のマイナスを補って余りある程度の増加がもたらされることで達成される。

つまり，検索ミスはサーチの有効性を高め購入者を増やすが消費者余剰にマイナスの影響を与え平均購入価格を引き上げる働きがあることになる。

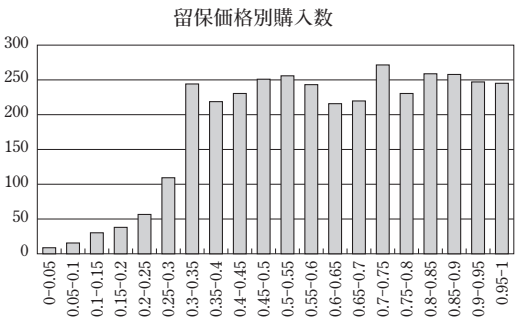
実際のサーチは留保価格と新規に販売開始する期待売手数で決まるが，不

28) 注６参照
29) 注16参照。

図表 3 － 9　留保価格ごとの購入件数（Seed2＝20049，検索ミスの有無別）

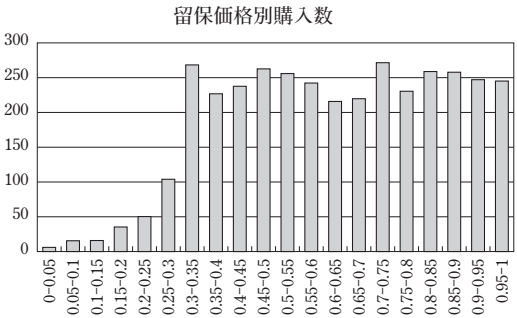
留保価格別購入件数 Seed2＝20049，検索ミス無し

0–0.05	0.05	5
0.05–0.1	0.1	13
0.1–0.15	0.15	28
0.15–0.2	0.2	36
0.2–0.25	0.25	54
0.25–0.3	0.3	109
0.3–0.35	0.35	246
0.35–0.4	0.4	220
0.4–0.45	0.45	232
0.45–0.5	0.5	253
0.5–0.55	0.55	258
0.55–0.6	0.6	246
0.6–0.65	0.65	217
0.65–0.7	0.7	222
0.7–0.75	0.75	276
0.75–0.8	0.8	233
0.8–0.85	0.85	261
0.85–0.9	0.9	259
0.9–0.95	0.95	250
0.95–1	1	248



留保価格別購入件数 Seed2＝20049，検索ミス50%

0–0.05	0.05	6
0.05–0.1	0.1	14
0.1–0.15	0.15	17
0.15–0.2	0.2	35
0.2–0.25	0.25	49
0.25–0.3	0.3	103
0.3–0.35	0.35	271
0.35–0.4	0.4	231
0.4–0.45	0.45	241
0.45–0.5	0.5	266
0.5–0.55	0.55	259
0.55–0.6	0.6	245
0.6–0.65	0.65	218
0.65–0.7	0.7	222
0.7–0.75	0.75	276
0.75–0.8	0.8	233
0.8–0.85	0.85	261
0.85–0.9	0.9	259
0.9–0.95	0.95	250
0.95–1	1	248



完全検索のせいで，低価格の売手が販売を繰り越す可能性について買手が気付いているとすれば，不完全検索の場合，サーチに踏み切る条件が完全検索よりも低くなる可能性があり，このシミュレーションではサーチしない設定となっている買手の一部がサーチを行い，購入に至るケースも出てくるかも知れない。

また，不完全サーチは，より低い価格のライバルがあるにもかかわらず，２番手以降が販売のチャンスを得ることを可能とすることにより，消費者余

剰の一部を売手にもたらす効果がある。

4．シミュレーションの有効性と課題

仮説を立証するのは事実において他にない。本稿におけるシミュレーションが，ある仮説の下で一定の論理的な帰結をもたらすことは間違いないところであるが，それは事実とは異なる世界である。その帰結を以て仮説が立証されるわけではない。そこで，シミュレーションの有効性とはどのようなものであるのか再確認しておく必要がある。

まず，仮説を帰結に導く論理すなわちモデルが一定の合理性を持っていることがシミュレーションで確かめられる点である。その帰結が明らかに事実に反するものであれば，シミュレーション自体に間違いがなければ，仮説かモデルのいずれかに瑕疵があることになる。したがって，シミュレーションの帰結は事実の観察によって再確認されなければならない。それは，第２の仮説をもたらすものという位置づけがなされうるだろう。しかしそれなら，はじめから事実によって検証すればよいという考え方も出てくるかも知れない。しかし，膨大な事実の中から何が仮説の帰結なのか選び出すことは困難であることも多い。シミュレーションが，一種の合理性の立証であるなら，そのモデルの持つ問題点も帰結の中から確認することが容易になるはずである。

膨大で曖昧な事実の中から普遍的な法則を見つけ出すことが理論研究の道であるなら，（結果的には）正しい仮説を立て，モデルで論理を組み立て，モデルの論理が正しいことを証明し，最後は事実の一部が説明できなければならない。論理は数学的な証明で正当性が確認されるが，もう一つの道としてシミュレーションによっても確認できるというのが本稿の立場である。

シミュレーションは，法廷における立証手法の一つである再現 VTR のようなものである。事実に近い状態で再現したとき，それまで気付かなかった

新たな「事実」が浮かび上がる[30]場合がある。その「事実」は，合理的なモデルの下での実験というフィルターを通して見た事実の断片である。しかし，そこで再現された「事実」のすべてが事実の一部ということにはならない。シミュレーションによって想定外の現象が見られたとき，それは事実の一部の発見ではなく，新たな仮説の発見であることがほとんどである[31]。その仮説を基にした新たなモデルとシミュレーションにより再確認のうえ，事実による立証が必要である。

本稿のシミュレーションは，売手が販売を開始する期と買手が検索を初めて行う期という「時間」をモデルに導入し，期毎の検索・購入・販売・サーチの各行動を価格サーチ理論の考え方に基づきシミュレーションした。元々，表計算は有限の離散的なデータ処理しかできないようになっている。元々現実の世界が有限であるので，処理時間の問題さえ解決できれば有限性自体がシミュレーションの制約になることはない。

序論で説明したように本稿では，期，売手数，買手数すべてに関して可能な限り大きな数字を採用したが，実は販売される商品数は売手数×10の5,000で，買手数と一致させている。本稿のシミュレーションでは，各売手の販売数量を乱数で発生させるサブルーチンもあり，平均10で発生させれば，確率的に需給一致の状況となるが，あえて一致しない数字でシミュレーションできるようになっている。

---

30) 法廷における再現ビデオで，犯行が困難であるという立証が行われ無罪となった例がある。

31) 例えば，図表3－8における傾向に反する例，すなわち消費者余剰も平均購入価格も非購入者数も不完全検索の方が大きい結果が，多数回の試行の末に確認できたとする。それは，現実の不完全検索の観察によって得られる結果とは異なる扱いにならざるを得ない。後者が実現した事実そのものであるのに対し，前者は事実の反映ではなく乱数列の偶然の偏りによってもたらされるかも知れないからである。その特異な現象が検討に値するのは，その乱数列の偶然の偏りと同様の偶然が無視できない確率で現実にも起きている場合だけである。この種の偏りに関しては永星〔5〕（2008）において分析している。

実際のシミュレーション結果では，購入に至らない買手[32]の割合が約27%であり，サーチをしない買手の割合（28.6%：注16参照）に近い。このサーチをしない買手の割合は，実は期の数200と売手数500から導き出された数字（次期に販売を開始する平均売手数）が基になっている。したがって，期の数と売手数の当初設定した数字が恣意的に影響しているのか，あるいは実際に有限な現実の世界でも重要な意味を持つのか[33]については別途検討を要する課題である。

拙稿〔3〕(2009）において，サーチコストを設定し，その分留保価格を下げるという手法で一種のストッピング・ルールを設けた。この方法だと，サーチコストにもよるがサーチはずっと早く収束する。実際の検索行動を考えても，何度も熱心に検索を繰り返す消費者は一般的な消費者よりもずっと価格に敏感でより安い提示価格を探していると考えられる。このような消費者の留保価格が低いと解釈すると，本稿のモデルではサーチを行わない可能性が高い。では本稿の設定は現実離れしているかというとそうではない。低い価格を求めている消費者が，検索したところ全く以て安値で買える見込みがないと判断した場合，購入自体をあきらめるという考え方にも一理ある。留保価格が不変であればその方が自然な考え方である。しかし拙稿〔3〕(2009）の「留保価格が変化する」という考え方も別途検討の余地がある。留保価格のハードルは，払った犠牲（サーチコスト）に比例して高くなるという考え方の他に，価格サーチの結果，安値がほとんど得られないという実質的な価格分布を認識することで，留保価格のハードルが下がるという逆のサーチ行動も想定できるのである。これら留保価格の有り様に関しては，現実の調査データを基にあらためてモデルを作り，そのモデルに基づいたシ

32) 購入に至らない買手は2種類ある。まず，サーチをしないタイプでその期の検索結果が留保価格を下回らなかった者。もう一つは，サーチを繰り返すが，最終期までに条件に見合う売手を見つけられない者。後者は無限の期・売手を仮定すると解消する。

33) 現実のネット市場で，購入のあてもほとんどなく延々と検索を繰り返す消費者がどれだけいるのか，信頼のおけるデータは今のところない。

ミュレーションを行う事ことができるだろう。

お わ り に

本稿では，拙稿〔3〕(2009) のシミュレーションモデルについて MT 乱数を用いて改良した。ポイントは，MT 乱数では扱いが容易な Seed を与えることで，再現実験が確実に行えること，同じ乱数列に関して検索ミスの有無，検索ミス率などの条件を変えて比較検討が容易になったことが挙げられる。また，MT 乱数自体良質な乱数であることも改善点の一つである。期間数や売手・買手の数はそれぞれ拡大したが，実行時間の増大はプログラムの見直しによって押さえることができた。実行時間の大部分はソートに費やされる[34]。サーチの結果として，末期に近づくほど購入が困難な買手が累積し，１期あたりの買手が増大する。その結果，買手5,000人の実際の検索回数は膨大な回数となり，ソートも多くなる。このことが，現実のネット市場において，希望の価格が見つからないまま，ひたすら検索を繰り返す検索者の累積として観察できるのかどうか，確認の必要がある。あるいは，他にサーチを止めてしまうメカニズムが存在するのか，異なるモデルが必要となるだろう。

拙稿〔3〕(2009) において確認されたタイミングの問題など各種の結論は今回のシミュレーションでも確認できる。それに加えて，同じ価格分布，同じ留保価格分布について，検索ミスの有無など異なる条件で比較を行った。具体的には消費者余剰の比較（それに加えて平均購入価格，非購入者数の比較）である。消費者余剰は完全検索の状態が最も高い。加えて完全検索の状態は，平均購入価格が最も低く，非購入者数は最も多い。不完全検索が，幸

34) ソートは，サーチによって買手が次期に再度検索を行う際，予め設定された検索点に 1 を加え，その買手を含めた，残りの買手をソートして検索点を昇順に並べ替えるために行う。次なる期における何番目にサーチした買手が入るかは，検索点の小数点以下の数字に依存する。それが 0 に近ければ，次期における上位での検索となり，1 に近ければ下位での検索になる。

運な留保価格の高い売手から安値で購入するチャンスを奪って，不運な比較的留保価格の低い買手に購入するチャンスをもたらすという意味で，上記のような結果をもたらすものと考えられる[35]。実際，平均的には検索ミスの率が高い程この傾向が強くなる。ただし各結果のばらつきは検索ミス率が高いほど大きく，個別の試行によっては傾向に反するものも散見されるようになる。このばらつきに関しては偶然が作用している可能性が高いが，偶然の要素以外の意味付けについては検討の余地がある。

はじめに述べたように，Excel の標準的な関数ではなく MT 乱数を用いることは，質的な改善のみならず，モンテカルロ・シミュレーションの基礎となる乱数をブラックボックスの中から取り出す意味がある。そのことで，実行結果の比較も明快になる。しかし，ブラックボックスはまだ存在する。パソコンのハードウェア，Excel あるいは VBA 自体もそうである。この問題は，同一の MT 乱数列を，異なるプラットフォームの異なるプログラム[36]に適用して比較することで検証できるだろう。

参考文献

〔1〕総務省統計局『平成 20 年版情報通信白書』2008 年。
〔2〕経済産業省『平成 19 年度我が国の IT 利活用に関する調査研究』2008 年。
〔3〕永星浩一（2009）「ネット市場における不確実性とサーチ」『福岡大学商学論叢』53/4，369-402。
〔4〕永星浩一（2008）「ネットにおける情報ソースの組み合わせとサーチ行動」『福岡大学商学論叢』53/2，115-139。
〔5〕永星浩一（2008）「買手サーチのシミュレーションによる実効性の検証」『応用経済分析Ⅰ：産業・都市・公共政策』勁草書房，2008。
〔6〕永星浩一（2007）「商品の追加購入を含んだサーチ・モデルの検証」『福岡大学商学論叢』52/2，139-168。
〔7〕津田孝雄（1995）『モンテカルロ法とシミュレーション〈三訂版〉』培風館，1995。
〔8〕脇本和昌（1970）『乱数の知識』森北出版，1970。

35) 巻末の検索ミス別の消費者余剰差と留保価格別購入件数の実行結果（図表）参照。
36) 例えば，Linux の C プログラムの配列計算で置き換えるなど。

プログラムおよび実行結果

```
Sub main01()　‘サーチあり検索ミスなしのシミュレーション
   Seller01
   Buyer02
   Search03
End Sub
```

```
Sub main02()　‘サーチあり検索ミスありのシミュレーション
   Seller01
   Buyer02
   Search04
End Sub
```

```
Sub main03()　‘検索ミス有無別，消費者余剰シミュレーション
Dim i, j As Integer
For i＝1 To 50
For j＝2 To 7
Sheets(“MT_Rand”).Cells(3,j).Value＝19999＋i
Next j
   Seller01
   Buyer02
   Search03
Sheets(“Sim02”).Cells(5007＋i,5).Value＝19999＋i
Sheets(“Sim02”).Cells(5007＋i,6).Value＝Sheets(“Sim02”).Cells(5004,6).Value
Sheets(“Sim02”).Cells(5007＋i,9).Value＝Sheets(“Sim02”).Cells(5002,6).Value
Sheets(“Sim02”).Cells(5007＋i,12).Value＝Sheets(“Sim02”).Cells(5003,6).Value
   Seller01
   Buyer02
   Search04
Sheets(“Sim02”).Cells(5007＋i,7).Value＝Sheets(“Sim02”).Cells(5004,6).Value
Sheets(“Sim02”).Cells(5007＋i,10).Value＝Sheets(“Sim02”).Cells(5002,6).Value
Sheets(“Sim02”).Cells(5007＋i,13).Value＝Sheets(“Sim02”).Cells(5003,6).Value
Sheets(“Sim02”).Cells(5007＋i,8).Value＝Sheets(“Sim02”).Cells(5007＋i,6).Value -
Sheets(“Sim02”).Cells(5007＋i,7).Value
Sheets(“Sim02”).Cells(5007＋i,11).Value＝Sheets(“Sim02”).Cells(5007＋i,9).Value -
Sheets(“Sim02”).Cells(5007＋i,10).Value
Sheets(“Sim02”).Cells(5007＋i,14).Value＝Sheets(“Sim02”).Cells(5007＋i,12).Value -
Sheets(“Sim02”).Cells(5007＋i,13).Value
Next i
End Sub
```

```
Sub Seller01()　‘売手の設定 02（通し番号，売値，販売数：10，販売開始期）
Dim MTArea01, UriteArea As String
Dim MTArea02, MTArea04, PriceArea, SellQArea, SellSArea As Variant
‘売手ラベルの MT 乱数 Sheet からの複写
With Sheets(“MT_Rand”)
      MTArea01＝.Range(“H 5”,.Range(“H65536”).End(xlUp)).Address
      MTArea02＝.Range(“B 5”,.Range(“B65536”).End(xlUp)).Address
```

```
        MTArea04＝.Range(“L 5”,.Range(“L65536”).End(xlUp)).Address
End With
With Sheets(“Sim01”)
        UriteArea＝.Range(“A3”,.Range(“A502”)).Address
        PriceArea＝.Range(“B3”,.Range(“B502”)).Address
        SellQArea＝.Range(“C3”,.Range(“C502”)).Address
        SellSArea＝.Range(“D3”,.Range(“D502”)).Address
End With
Sheets(“Sim01”).Range(UriteArea).Value＝Sheets(“MT_Rand”).Range(MTArea01).Value　‘売手ラベル
Sheets(“Sim01”).Range(PriceArea).Value＝Sheets(“MT_Rand”).Range(MTArea02).Value　‘売値
Sheets(“Sim01”).Range(SellQArea).Value＝10　‘販売数
Sheets(“Sim01”).Range(SellSArea).Value＝Sheets(“MT_Rand”).Range(MTArea04).Value　‘販売開始期
‘売手を価格の安い順にソート
Sheets(“Sim01”).Range(Sheets(“Sim01”).Cells(3, 1), Sheets(“Sim01”).Cells(502, 4)).Sort key 1 :＝Sheets(“Sim01”).Cells(3, 2),order 1 :＝xlAscending
End Sub
```

```
Sub Seller02()　‘売手の設定 01（通し番号，売値，販売数：1～10 の一様乱数，販売開始期）
Dim MTArea01, UriteArea As String
Dim MTArea02, MTArea03, MTArea04, PriceArea, SellQArea, SellSArea As Variant
‘売手ラベルの MT 乱数 Sheet からの複写
With Sheets(“MT_Rand”)
        MTArea01＝.Range(“H5”,.Range(“H65536”).End(xlUp)).Address
        MTArea02＝.Range(“B5”,.Range(“B65536”).End(xlUp)).Address
        MTArea03＝.Range(“K5”,.Range(“K65536”).End(xlUp)).Address
        MTArea04＝.Range(“L5”,.Range(“L65536”).End(xlUp)).Address
End With
With Sheets(“Sim01”)
        UriteArea＝.Range(“A3”,.Range(“A502”)).Address
        PriceArea＝.Range(“B3”,.Range(“B502”)).Address
        SellQArea＝.Range(“C3”,.Range(“C502”)).Address
        SellSArea＝.Range(“D3”,.Range(“D502”)).Address
End With
Sheets(“Sim01”).Range(UriteArea).Value＝Sheets(“MT_Rand”).Range(MTArea01).Value　‘売手ラベル
Sheets(“Sim01”).Range(PriceArea).Value＝Sheets(“MT_Rand”).Range(MTArea02).Value　‘売値
Sheets(“Sim01”).Range(SellQArea).Value＝Sheets(“MT_Rand”).Range(MTArea03).Value　‘販売数
Sheets(“Sim01”).Range(SellSArea).Value＝Sheets(“MT_Rand”).Range(MTArea04).Value　‘販売開始期
‘売手を価格の安い順にソート
Sheets(“Sim01”).Range(Sheets(“Sim01”).Cells(3, 1), Sheets(“Sim01”).Cells(502, 4)).Sort key 1 :＝Sheets(“Sim01”).Cells(3, 2),order 1 :＝xlAscending
End Sub
```

```
Sub Buyer01()　‘見出しと買手の検索時設定（留保価格 1 のケース）
Range(Sheets(“Sim02”).Cells(2, 5),Sheets(“Sim02”).Cells(5001, 212)).Clear
Dim MTArea05, KaiteArea As String
Dim MTArea06, MTArea07, SPointArea, SPeriodArea, RPriceArea As Variant
‘買手ラベルと検索時（1～200），留保価格（0～1）の MT_Rand からの複写
With Sheets(“MT_Rand”)
    MTArea05＝.Range(“I5”,.Range(“I65536”).End(xlUp)).Address
    MTArea06＝.Range(“M5”,.Range(“M65536”).End(xlUp)).Address
    MTArea07＝.Range(“N5”,.Range(“N65536”).End(xlUp)).Address
    KaiteArea＝.Range(“A2”,.Range(“A5001”)).Address
    SPointArea＝.Range(“B2”,.Range(“B5001”)).Address
    SPeriodArea＝.Range(“C2”,.Range(“C5001”)).Address
    RPriceArea＝.Range(“D2”,.Range(“D5001”)).Address
End With
Sheets(“Sim02”).Range(KaiteArea).Value＝Sheets(“MT_Rand”).Range(MTArea05).Value　‘買手ラベル
Sheets(“Sim02”).Range(SPointArea).Value＝Sheets(“MT_Rand”).Range(MTArea06).Value　‘検索点
Sheets(“Sim02”).Range(SPeriodArea).Value＝Sheets(“MT_Rand”).Range(MTArea07).Value　‘検索期
Sheets(“Sim02”).Range(RPriceArea).Value＝1　‘買手の留保価格
‘買手を検索点に関して昇順にソート
‘5000 に 1 加えるのは見出し用
Sheets(“Sim02”).Range(Sheets(“Sim02”).Cells(2, 1), Sheets(“Sim02”).Cells(5001, 5)).Sort key 1 :＝Sheets(“Sim02”).Cells(2, 2), order 1 : ＝xlAscending
End Sub
```

```
Sub Buyer02()　‘見出しと買手の検索時設定（留保価格を乱数で決める）
Range(Sheets(“Sim02”).Cells(2, 5), Sheets(“Sim02”).Cells(5001, 212)).Clear
Dim MTArea05, KaiteArea As String
Dim MTArea06, MTArea07, MTArea08, SPointArea, SPeriodArea, RPriceArea As Variant
‘買手ラベルと検索時（1～200），留保価格（0～1）の MT_Rand からの複写
With Sheets(“MT_Rand”)
    MTArea05＝.Range(“I5”,.Range(“I65536”).End(xlUp)).Address
    MTArea06＝.Range(“M5”,.Range(“M65536”).End(xlUp)).Address
    MTArea07＝.Range(“N5”,.Range(“N65536”).End(xlUp)).Address
    MTArea08＝.Range(“F5”,.Range(“F65536”).End(xlUp)).Address
    KaiteArea＝.Range(“A2”,.Range(“A5001”)).Address
    SPointArea＝.Range(“B2”,.Range(“B5001”)).Address
    SPeriodArea＝.Range(“C2”,.Range(“C5001”)).Address
    RPriceArea＝.Range(“D2”,.Range(“D5001”)).Address
End With
Sheets(“Sim02”).Range(KaiteArea).Value＝Sheets(“MT_Rand”).Range(MTArea05).Value　‘買手ラベル
Sheets(“Sim02”).Range(SPointArea).Value＝Sheets(“MT_Rand”).Range(MTArea06).Value　‘検索点
Sheets(“Sim02”).Range(SPeriodArea).Value＝Sheets(“MT_Rand”).Range(MTArea07).Value　‘検索期
```

```
Sheets(“Sim02”).Range(RPriceArea).Value＝Sheets(“MT_Rand”).Range(MTArea08).Value　‘留保価格

‘以下のようにすると非常に時間がかかる
‘For i＝1 To 5000
　‘Sheets(“Sim02”).Cells(i＋1, 1).Value＝”B”&i　‘買手のラベル
　‘Sheets(“Sim02”).Cells(i＋1, 2).Value＝Sheets(“MT_Rand”).Cells(i＋4, 29).Value　‘検索点
　‘Sheets(“Sim02”).Cells(i＋1, 3).Value＝Sheets(“MT_Rand”).Cells(i＋4, 30).Value　‘検索期
　‘Sheets(“Sim02”).Cells(i＋1, 4).Value＝Sheets(“MT_Rand”).Cells(i＋4, 6).Value　‘留保価格
‘Next i
‘買手を検索点に関して昇順にソート
‘5000 に 1 加えるのは見出し用
Sheets(“Sim02”).Range(Sheets(“Sim02”).Cells(2, 1), Sheets(“Sim02”).Cells(5001, 5)).Sort key 1 :＝Sheets(“Sim02”).Cells(2, 2), order 1 : ＝xlAscending
End Sub
```

```
Sub Search02()　‘1 期だけの検索，検索ミス対応版
Dim MinPrice, Smiss As Double
Dim SearchTime, MinRaw, MinColumn As Integer
Smiss＝Sheets(“Sim01”).Cells(1, 4).Value　‘検索ミスの確率
Range(Sheets(“Sim02”).Cells(2, 6), Sheets(“Sim02”).Cells(5001, 206)).Clear　‘Sim02 のクリア
‘商品の検索・比較ならびに購入の決定
For j＝1 To 5000　‘買手の数だけ試行
　MinPrice＝1
　SearchTime＝Sheets(“Sim02”).Cells(j＋1, 3).Value
　MinRaw＝0　‘ある時点で最低価格の売手（行番号）の初期値
　MinColumn＝0　‘同上の列番号の初期値
　Sheets(“MT_Rand”).Cells(2, 7).Value＝11000＋j　‘MT 乱数の Seed1 を j に応じて変化させる。（500 個の 0～1 の一様乱数）
　i＝3
　Do
　　If Sheets(“Sim01”).Cells(i, SearchTime＋6).Value＜＞””And Sheets(“Sim01”).Cells(i, SearchTime＋6).Value＜＞0 Then
　　　If Sheets(“MT_Rand”).Cells(i＋2, 7).Value＞＝Smiss Then　‘検索ミスしない確率で以下を実行。左辺は MT 乱数。
　　　　MinPrice＝Sheets(“Sim01”).Cells(i, 2).Value
　　　　MinRaw＝i
　　　　MinColumn＝SearchTime＋6
　　　　Exit Do
　　　End If
　　End If
i＝i＋1
Loop Until i＞502
　If MinPrice＜1 Then
　　‘第 2 シートへの売れ数の書き込み
　　Sheets(“Sim02”).Cells(MinRaw, MinColumn).Value＝Sheets(“Sim02”).Cells(MinRaw, MinColumn).Value＋1
```

```
    ‘第 2 シートへの各買手の買値の書き込み
    Sheets(“Sim02”).Cells(j＋1, 6).Value＝Sheets(“Sim01”).Cells(MinRaw, 2).Value
  Else
    ‘第 2 シートへの買えなかった買手への書き込み
    Sheets(“Sim02”).Cells(j＋1, 6).Value＝“×”
  End If
Next j
End Sub
```

```
Sub Search02()　‘1 期だけの検索，検索ミス対応版
Dim MinPrice, Smiss As Double
Dim SearchTime, MinRaw, MinColumn As Integer
Smiss＝Sheets(“Sim01”).Cells(1, 4).Value　‘検索ミスの確率
Range(Sheets(“Sim02”).Cells(2, 6), Sheets(“Sim02”).Cells(5001, 206)).Clear　‘Sim02 のクリア
‘商品の検索・比較ならびに購入の決定
For j＝1 To 5000　‘買手の数だけ試行
  MinPrice＝1
  SearchTime＝Sheets(“Sim02”).Cells(j＋1, 3).Value
  MinRaw＝0　‘ある時点で最低価格の売手（行番号）の初期値
  MinColumn＝0　‘同上の列番号の初期値
  Sheets(“MT_Rand”).Cells(2, 7).Value＝11000＋j　‘MT 乱数の Seed1 を j に応じて変化さ
  せる。（500 個の 0～1 の一様乱数）
  i＝3
  Do
    If Sheets(“Sim01”).Cells(i, SearchTime＋6).Value＜＞””And Sheets(“Sim01”).Cells(i,
    SearchTime＋6).Value＜＞0 Then
      If Sheets(“MT_Rand”).Cells(i＋2, 7).Value＞＝Smiss Then　‘検索ミスしない確率で
      以下を実行。
          MinPrice＝Sheets(“Sim01”).Cells(i,2).Value
          MinRaw＝i
          MinColumn＝SearchTime＋6
          Exit Do
      End If
    End If
i＝i＋1
Loop Until i＞502
  If MinPrice＜1 Then
    ‘第 2 シートへの売れ数の書き込み
    Sheets(“Sim02”).Cells(MinRaw,　MinColumn).Value＝Sheets(“Sim02”).Cells(MinRaw,
    MinColumn).Value＋1
    ‘第 2 シートへの各買手の買値の書き込み
    Sheets(“Sim02”).Cells(j＋1, 6).Value＝Sheets(“Sim01”).Cells(MinRaw, 2).Value
  Else
    ‘第 2 シートへの買えなかった買手への書き込み
    Sheets(“Sim02”).Cells(j＋1, 6).Value＝“×”
  End If
Next j
End Sub
```

```
Sub Search03()　‘複数期間の検索，検索ミス対応版
Dim MinPrice, SearchTime, MinRaw, MinColumn, Scost, Rprice, k, BNum, SNum, ESNum, Smiss, Sticket, Rticket
BNum＝Sheets(“Sim01”).Cells(103, 3).Value　‘買手の数を BNum に代入
Smiss＝Sheets(“Sim01”).Cells(1, 4).Value　‘検索ミスの確率
Scost＝Sheets(“Sim01”).Cells(1, 2).Value　‘1 回あたりのサーチコスト。0.1～0.5
Range(Sheets(“Sim02”).Cells(2, 6), Sheets(“Sim02”).Cells(BNum＋1, 6)).Clear　‘Sim02 のクリア
Range(Sheets(“Sim02”).Cells(3, 7), Sheets(“Sim02”).Cells(102, 106)).Clear　‘Sim02 のクリア
‘商品の検索•比較ならびに購入の決定
k＝1
Do
  MinPrice＝1
  SearchTime＝Sheets(“Sim02”).Cells(k＋1, 3).Value
If SearchTime＜＝100 Then
  MinRaw＝0　‘ある時点で最低価格の売手（行番号）の初期化
  MinColumn＝0　‘同上の列番号の初期値化
  SNum＝0　‘売手数の初期値
  Sticket＝0　‘サーチをするか否か。初期化（1 のときはする，0 のときはしない）。
  ‘その期の売手数（次期の売手数の推測に使う）
  For i＝3 To 102
    If Sheets(“Sim01”).Cells(i, SearchTime＋6).Value＜＞””Then
      SNum＝SNum＋1
    End If
  Next i
  ‘検索点の乱数を再利用して作成した買手の留保価格を読み込み。
    Rprice＝Sheets(“Sim02”).Cells(k＋1, 4).Value - Sheets(“Sim02”).Cells(k＋1, 5).Value *Scost ‘Rprice＝1　‘基本モデルで留保価格が 1 の時，サーチせず。
  If Rprice＜0 Then Rprice＝0　‘留保価格がマイナスになったとき 0 とする。
    ESNum＝Rprice *SNum　‘留保価格以下の売手の期待数
      If ESNum>＝1 Then　‘1 以上で 1 人の売手は留保価格以下と考え，サーチをする。
    Sticket＝1
  Else
    Sticket＝0
  End If
For i＝3 To 102　‘より安い売手が出れば最安値として記録（繰り返し）
  If Sheets(“Sim01”).Cells(i, SearchTime＋6).Value＜＞””And Sheets(“Sim01”).Cells(i, SearchTime＋6).Value＜＞0 Then
    If Rnd＞＝Sheets(“Sim01”).Cells(1, 4).Value Then　‘検索ミスしない確率で以下を実行
      If MinPrice＞Sheets(“Sim01”).Cells(i, 2).Value Then　‘売手の提示価格の方が安ければ
        MinPrice＝Sheets(“Sim01”).Cells(i, 2).Value
        MinRaw＝i
        MinColumn＝SearchTime＋6
      End If
    End If
  End If
```

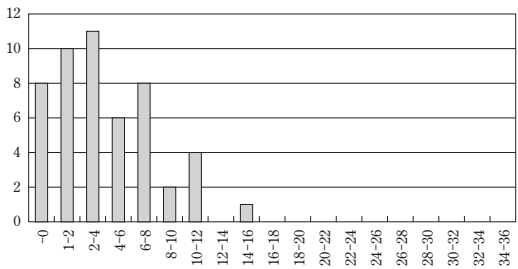
```
Next i
  If MinPrice＜1 Then　‘売手が存在し
      If MinPrice＜＝Rprice Then　‘留保価格よりも安ければ購入
        ‘第 2 シートへの売れ数の書き込み
        Sheets(“Sim02”).Cells(MinRaw, MinColumn).Value＝Sheets(“Sim02”).Cells(MinRaw,
        MinColumn).Value＋1
        ‘第 2 シートへの各買手の買値の書き込み
        Sheets(“Sim02”).Cells(k＋1, 6).Value＝Sheets(“Sim01”).Cells(MinRaw, 2).Value
      ElseIf SearchTime＜100 And Sticket＝1 Then　‘留保価格よりも高く，最終期ではなく，次期に期待があればサーチ
        Sheets(“Sim02”).Cells(k＋1, 5).Value＝Sheets(“Sim02”).Cells(k＋1, 5).Value＋1
        ‘サーチ回数の記録
        ‘売手が見つからない場合，検索期（購入時・購入期とも）を 1 後ろにずらす
        Sheets(“Sim02”).Cells(k＋1, 2).Value＝Sheets(“Sim02”).Cells(k＋1, 2).Value＋1
        Sheets(“Sim02”).Cells(k＋1, 3).Value＝Sheets(“Sim02”).Cells(k＋1, 3).Value＋1
        ‘買手を新しい検索順に再ソート
        Sheets(“Sim02”).Range(Sheets(“Sim02”).Cells(k＋1, 1),Sheets(“Sim02”).Cells(BNum,
        6)).Sort
key 1 : ＝Sheets(“Sim02”).Cells(k＋1, 2), order 1 : ＝xlAscending
        k＝k－1　‘今期の買手が次期にずれるので，次の買手をこの期にずらす。
      End If　‘留保価格より高く，あるいは最終期で次期に期待がなければあきらめる
  ElseIf SearchTime＜100 And Sticket＝1 Then　‘売手が存在せず，最終期ではなく，次期に期待があればサーチ
      Sheets(“Sim02”).Cells(k＋1, 5).Value＝Sheets(“Sim02”).Cells(k＋1, 5).Value＋1
      ‘サーチ回数の記録
      ‘売手が見つからない（いないか留保価格より大きかった）場合，検索期（購入時・購入期とも）を 1 後ろにずらす
      Sheets(“Sim02”).Cells(k＋1, 2).Value＝Sheets(“Sim02”).Cells(k＋1, 2).Value＋1
      Sheets(“Sim02”).Cells(k＋1, 3).Value＝Sheets(“Sim02”).Cells(k＋1, 3).Value＋1
      ‘買手を新しい検索順に再ソート
      Sheets(“Sim02”).Range(Sheets(“Sim02”).Cells(k＋1, 1), Sheets(“Sim02”).Cells(BNum,
      6)).Sort
key 1 : ＝Sheets(“Sim02”).Cells(k＋1, 2), order 1 : ＝xlAscending
      k＝k－1　‘今期の買手が次期にずれるので，次の買手をこの期にずらす。
  End If
End If
  k＝k＋1
Loop Until k＞BNum　‘買手の数まで繰り返す
End Sub
```

検索ミス無し(a)と検索ミス率25%(b)の比較

	Seed2	消費者余剰		差	平均購入価格		差	非購入者数		差
		(a)	(b)		(a)	(b)		(a)	(b)	
1	20000	924.78053	922.9914	1.789127	0.37939492	0.38098493	−0.00159	1413	1397	16
2	20001	985.24593	980.8256	4.420357	0.36062637	0.36262603	−0.002	1438	1416	22
3	20002	986.72204	986.3464	0.375605	0.35702326	0.3576969	−0.000674	1322	1316	6
4	20003	919.40324	915.2204	4.182812	0.38599265	0.38801308	−0.00202	1404	1386	18
5	20004	949.36222	945.39	3.972214	0.36727945	0.36928	−0.002001	1355	1336	19
6	20005	944.48365	937.7508	6.732857	0.37717791	0.37953072	−0.002353	1415	1394	21
7	20006	973.62979	959.0871	14.54267	0.36132167	0.36535569	−0.004034	1385	1346	39
8	20007	980.09098	982.3473	−2.25635	0.36235269	0.36284017	−0.000487	1368	1366	2
9	20008	1031.1478	1027.461	3.687091	0.33985137	0.34151264	−0.001661	1395	1380	15
10	20009	1019.4697	1012.775	6.669443	0.35290526	0.35522914	−0.002324	1366	1343	23
11	20010	974.61681	968.141	6.475766	0.36035775	0.36251404	−0.002156	1414	1392	22
12	20011	995.77568	998.3151	−2.5394	0.36051812	0.36227843	−0.00176	1361	1344	17
13	20012	959.83863	948.3268	11.5118	0.36794569	0.37164174	−0.003696	1431	1393	38
14	20013	1017.1735	1014.49	2.683397	0.34854117	0.35059177	−0.002051	1344	1326	18
15	20014	997.36437	997.1905	0.17389	0.35605976	0.35783443	−0.001775	1352	1336	16
16	20015	1000.6978	999.2213	1.476525	0.35757738	0.35919538	−0.001618	1333	1318	15
17	20016	1016.7919	1013.244	3.547691	0.35011876	0.35116377	−0.001045	1391	1379	12
18	20017	935.19883	923.8776	11.32118	0.37549621	0.37905269	−0.003556	1451	1412	39
19	20018	1003.9741	1000.538	3.436053	0.35268956	0.3541549	−0.001465	1313	1296	17
20	20019	963.3194	960.8176	2.501808	0.36790423	0.36871842	−0.000814	1387	1380	7
21	20020	934.19424	926.747	7.44726	0.36334234	0.36591604	−0.002574	1433	1410	23
22	20021	1006.203	999.4799	6.723092	0.35256036	0.35456833	−0.002008	1305	1286	19
23	20022	908.49488	901.7059	6.789015	0.37302464	0.3755215	−0.002497	1437	1412	25
24	20023	918.68238	917.2048	1.477594	0.38169776	0.38277789	−0.00108	1400	1391	9
25	20024	968.09627	960.1658	7.930436	0.35849782	0.36061025	−0.002112	1365	1344	21
26	20025	992.68157	991.8172	0.864364	0.35186549	0.35288887	−0.001023	1403	1393	10
27	20026	989.23978	984.0398	5.199967	0.34967999	0.35139606	−0.001716	1404	1388	16
28	20027	974.61324	974.804	−0.19073	0.35626934	0.35666331	−0.000394	1285	1283	2
29	20028	984.9343	981.2821	3.652161	0.36096282	0.36237018	−0.001407	1341	1330	11
30	20029	953.32971	951.4752	1.854562	0.37160406	0.37263886	−0.001035	1324	1318	6
31	20030	941.90285	940.5628	1.340031	0.3797088	0.38075708	−0.001048	1323	1316	7
32	20031	943.863	935.375	8.488034	0.36611302	0.3696213	−0.003508	1409	1376	33
33	20032	939.69282	935.9263	3.766533	0.37127617	0.37358628	−0.00231	1424	1403	21
34	20033	982.64697	982.0361	0.610821	0.35476235	0.35546857	−0.000706	1324	1320	4
35	20034	948.61058	942.8735	5.737071	0.36761645	0.36999232	−0.002376	1430	1408	22
36	20035	967.7998	961.7926	6.007246	0.35256667	0.35507628	−0.00251	1401	1378	23
37	20036	945.59716	948.0764	−2.47923	0.37289348	0.37316995	−0.000276	1358	1356	2

	Seed2	消費者余剰		差	平均購入価格		差	非購入者数		差
		(a)	(b)		(a)	(b)		(a)	(b)	
38	20037	958.31793	956.1277	2.190218	0.34979966	0.35203807	−0.002238	1514	1492	22
39	20038	946.91224	943.7201	3.192156	0.36875188	0.37074463	−0.001993	1431	1411	20
40	20039	881.16778	875.4908	5.67698	0.3934981	0.39492963	−0.001432	1402	1388	14
41	20040	937.39611	931.5563	5.839796	0.36647263	0.36870963	−0.002237	1360	1340	20
42	20041	961.04437	957.1472	3.897142	0.36673172	0.36858812	−0.001856	1379	1362	17
43	20042	917.30758	918.9306	−1.62306	0.37001989	0.371257	−0.001237	1419	1408	11
44	20043	944.44124	933.789	10.65229	0.3691638	0.37197109	−0.002807	1399	1372	27
45	20044	989.07262	993.3032	−4.23057	0.35970957	0.35981809	−0.000109	1341	1340	1
46	20045	949.02602	952.3969	−3.3709	0.37424651	0.37386713	0.0003794	1259	1262	−3
47	20046	960.70068	961.4544	−0.75373	0.36146746	0.36230591	−0.000838	1399	1392	7
48	20047	1022.0571	1020.755	1.301667	0.35114391	0.35179676	−0.000653	1350	1343	7
49	20048	1060.9098	1049.744	11.16584	0.34391478	0.34730937	−0.003395	1315	1282	33
50	20049	964.5623	956.0914	8.470921	0.36245401	0.36501069	−0.002557	1334	1311	23
			平均	3.84713		平均	−0.001773		平均	16.7
			標準偏差	4.113733		標準偏差	0.0009518		標準偏差	9.769852

余剰	階　級	度数分布
－0	0	8
1－2	2	10
2－4	4	11
4－6	6	6
6－8	8	8
8－10	10	2
10－12	12	4
12－14	14	0
14－16	16	1
16－18	18	0
18－20	20	0
20－22	22	0
22－24	24	0
24－26	26	0
26－28	28	0
28－30	30	0
30－32	32	0
32－34	34	0
34－36	36	0



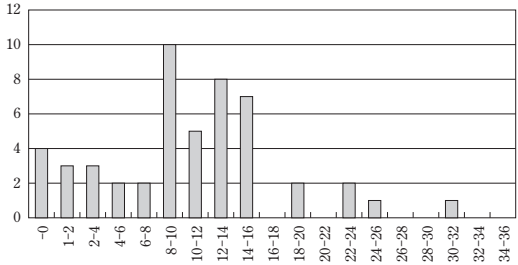
検索ミス無し(a)と検索ミス率50％(c)の比較

	Seed2	消費者余剰		差	平均購入価格		差	非購入者数		差
		(a)	(c)		(a)	(c)		(a)	(c)	
1	20000	924.78053	901.5852	23.19533	0.37939492	0.38691956	−0.007525	1413	1344	69
2	20001	985.24593	975.4905	9.755467	0.36062637	0.36527148	−0.004645	1438	1387	51
3	20002	986.72204	989.8665	−3.14441	0.35702326	0.3585693	−0.001546	1322	1309	13
4	20003	919.40324	914.7372	4.666072	0.38599265	0.389651	−0.003658	1404	1371	33
5	20004	949.36222	941.1734	8.188823	0.36727945	0.37175001	−0.004471	1355	1312	43
6	20005	944.48365	931.1797	13.30399	0.37717791	0.38199651	−0.004819	1415	1371	44
7	20006	973.62979	962.3948	11.235	0.36132167	0.36699072	−0.005669	1385	1337	48
8	20007	980.09098	968.8793	11.21171	0.36235269	0.36707248	−0.00472	1368	1329	39
9	20008	1031.1478	1015.759	15.38915	0.33985137	0.34534395	−0.005493	1395	1346	49
10	20009	1019.4697	996.8748	22.59487	0.35290526	0.35995352	−0.007048	1366	1297	69
11	20010	974.61681	960.2666	14.35017	0.36035775	0.36688328	−0.006526	1414	1350	64
12	20011	995.77568	983.1778	12.59789	0.36051812	0.36644839	−0.00593	1361	1304	57
13	20012	959.83863	935.1573	24.68136	0.36794569	0.37568613	−0.00774	1431	1353	78
14	20013	1017.1735	1017.238	−0.06455	0.34854117	0.35117746	−0.002636	1344	1323	21
15	20014	997.36437	987.7256	9.638806	0.35605976	0.36093431	−0.004875	1352	1307	45
16	20015	1000.6978	1000.744	−0.04589	0.35757738	0.35985042	−0.002273	1333	1320	13
17	20016	1016.7919	1007.886	8.90549	0.35011876	0.35323519	−0.003116	1391	1356	35
18	20017	935.19883	904.7459	30.45293	0.37549621	0.38454694	−0.009051	1451	1361	90
19	20018	1003.9741	997.9477	6.026414	0.35268956	0.35591545	−0.003226	1313	1285	28
20	20019	963.3194	958.4341	4.885271	0.36790423	0.37055916	−0.002655	1387	1364	23
21	20020	934.19424	921.2999	12.8943	0.36334234	0.36834716	−0.005005	1433	1383	50
22	20021	1006.203	1006.077	0.126158	0.35256036	0.35429014	−0.00173	1305	1291	14
23	20022	908.49488	894.3849	14.11	0.37302464	0.37964526	−0.006621	1437	1370	67
24	20023	918.68238	904.3481	14.33432	0.38169776	0.38671026	−0.005013	1400	1354	46
25	20024	968.09627	964.5063	3.589989	0.35849782	0.36071728	−0.002219	1365	1347	18
26	20025	992.68157	978.1512	14.53037	0.35186549	0.35719378	−0.005328	1403	1352	51
27	20026	989.23978	979.5637	9.676103	0.34967999	0.35318492	−0.003505	1404	1371	33
28	20027	974.61324	971.4008	3.212417	0.35626934	0.35830375	−0.002034	1285	1274	11
29	20028	984.9343	975.8909	9.043445	0.36096282	0.36496358	−0.004001	1341	1308	33
30	20029	953.32971	938.7751	14.5546	0.37160406	0.37625008	−0.004646	1324	1287	37
31	20030	941.90285	939.7624	2.140401	0.3797088	0.3818133	−0.002105	1323	1308	15
32	20031	943.863	931.3949	12.46812	0.36611302	0.37190717	−0.005794	1409	1356	53
33	20032	939.69282	933.5186	6.174186	0.37127617	0.37631796	−0.005042	1424	1377	47
34	20033	982.64697	969.8783	12.76863	0.35476235	0.35888002	−0.004118	1324	1291	33
35	20034	948.61058	939.9602	8.650377	0.36761645	0.37284268	−0.005226	1430	1384	46
36	20035	967.7998	948.6635	19.13625	0.35256667	0.35888024	−0.006314	1401	1343	58
37	20036	945.59716	933.4658	12.13136	0.37289348	0.3774145	−0.004521	1358	1315	43

	Seed2	消費者余剰		差	平均購入価格		差	非購入者数		差
		(a)	(c)		(a)	(c)		(a)	(c)	
38	20037	958.31793	956.3732	1.944729	0.34979966	0.35389312	−0.004093	1514	1475	39
39	20038	946.91224	931.4691	15.4431	0.36875188	0.37446534	−0.005713	1431	1375	56
40	20039	881.16778	870.5782	10.58957	0.3934981	0.3973251	−0.003827	1402	1366	36
41	20040	937.39611	928.2227	9.173443	0.36647263	0.36991621	−0.003444	1360	1332	28
42	20041	961.04437	941.4147	19.62972	0.36673172	0.37284935	−0.006118	1379	1323	56
43	20042	917.30758	907.4266	9.880957	0.37001989	0.37530788	−0.005288	1419	1373	46
44	20043	944.44124	932.0621	12.37911	0.3691638	0.37384393	−0.00468	1399	1355	44
45	20044	989.07262	988.0251	1.047554	0.35970957	0.36225098	−0.002541	1341	1316	25
46	20045	949.02602	950.0403	−1.01432	0.37424651	0.37581862	−0.001572	1259	1248	11
47	20046	960.70068	951.8979	8.802764	0.36146746	0.36641595	−0.004948	1399	1352	47
48	20047	1022.0571	1011.856	10.20059	0.35114391	0.35458212	−0.003438	1350	1318	32
49	20048	1060.9098	1049.047	11.86292	0.34391478	0.34877099	−0.004856	1315	1271	44
50	20049	964.5623	951.1834	13.37894	0.36245401	0.36728393	−0.00483	1334	1296	38
			平均	10.41368		平均	−0.004524		平均	41.38
			標準偏差	6.840342		標準偏差	0.0016642		標準偏差	17.50302

余剰	階　級	度数分布
－0	0	4
1－2	2	3
2－4	4	3
4－6	6	2
6－8	8	2
8－10	10	10
10－12	12	5
12－14	14	8
14－16	16	7
16－18	18	0
18－20	20	2
20－22	22	0
22－24	24	2
24－26	26	1
26－28	28	0
28－30	30	0
30－32	32	1
32－34	34	0
34－36	36	0

消費者余剰差の分布



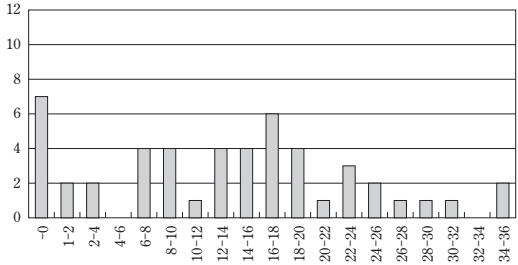
検索ミス無し(a)と検索ミス率75%(d)の比較

	Seed2	消費者余剰		差	平均購入価格		差	非購入者数		差
		(a)	(d)		(a)	(d)		(a)	(d)	
1	20000	924.78053	896.2942	28.48634	0.37939492	0.38892448	−0.00953	1413	1347	66
2	20001	985.24593	967.8951	17.35085	0.36062637	0.36688702	−0.006261	1438	1376	62
3	20002	986.72204	993.9797	−7.25771	0.35702326	0.35843539	−0.001412	1322	1321	1
4	20003	919.40324	888.81	30.5932	0.38599265	0.39599592	−0.010003	1404	1317	87
5	20004	949.36222	925.9956	23.36657	0.36727945	0.37524367	−0.007964	1355	1286	69
6	20005	944.48365	920.4788	24.00483	0.37717791	0.38487585	−0.007698	1415	1354	61
7	20006	973.62979	950.6949	22.93494	0.36132167	0.37018173	−0.00886	1385	1339	46
8	20007	980.09098	961.7603	18.3307	0.36235269	0.36966874	−0.007316	1368	1319	49
9	20008	1031.1478	1024.473	6.674334	0.33985137	0.34552989	−0.005679	1395	1348	47
10	20009	1019.4697	1001.62	17.84972	0.35290526	0.36019567	−0.00729	1366	1306	60
11	20010	974.61681	947.7481	26.86871	0.36035775	0.37042428	−0.010067	1414	1336	78
12	20011	995.77568	986.1797	9.595951	0.36051812	0.36798879	−0.007471	1361	1293	68
13	20012	959.83863	924.1998	35.63879	0.36794569	0.37873083	−0.010785	1431	1343	88
14	20013	1017.1735	1015.473	1.700627	0.34854117	0.35200996	−0.003469	1344	1334	10
15	20014	997.36437	979.7433	17.6211	0.35605976	0.36475433	−0.008695	1352	1281	71
16	20015	1000.6978	998.8021	1.89574	0.35757738	0.36111421	−0.003537	1333	1331	2
17	20016	1016.7919	1006.805	9.986687	0.35011876	0.35429429	−0.004176	1391	1346	45
18	20017	935.19883	892.779	42.41983	0.37549621	0.3881694	−0.012673	1451	1354	97
19	20018	1003.9741	990.8872	13.08688	0.35268956	0.35810732	−0.005418	1313	1277	36
20	20019	963.3194	954.2028	9.116614	0.36790423	0.37219751	−0.004293	1387	1357	30
21	20020	934.19424	921.469	12.72524	0.36334234	0.37035058	−0.007008	1433	1408	25
22	20021	1006.203	999.8506	6.352424	0.35256036	0.35599389	−0.003434	1305	1287	18
23	20022	908.49488	873.6129	34.88198	0.37302464	0.38505007	−0.012025	1437	1322	115
24	20023	918.68238	899.1875	19.49488	0.38169776	0.38970858	−0.008011	1400	1353	47
25	20024	968.09627	969.3717	−1.27547	0.35849782	0.36047978	−0.001982	1365	1362	3
26	20025	992.68157	983.6026	9.078965	0.35186549	0.35744106	−0.005576	1403	1353	50
27	20026	989.23978	985.2467	3.993099	0.34967999	0.35327245	−0.003592	1404	1383	21
28	20027	974.61324	977.5144	−2.90114	0.35626934	0.35833414	−0.002065	1285	1305	−20
29	20028	984.9343	981.0748	3.859451	0.36096282	0.36515076	−0.004188	1341	1319	22
30	20029	953.32971	942.9195	10.4102	0.37160406	0.37708287	−0.005479	1324	1313	11
31	20030	941.90285	943.1952	−1.29231	0.3797088	0.38282835	−0.00312	1323	1312	11
32	20031	943.863	926.734	17.12904	0.36611302	0.37512377	−0.009011	1409	1341	68
33	20032	939.69282	933.1938	6.49888	0.37127617	0.37790417	−0.006628	1424	1373	51
34	20033	982.64697	990.3398	−7.6928	0.35476235	0.35552819	−0.000766	1324	1335	−11
35	20034	948.61058	924.2824	24.32822	0.36761645	0.37772372	−0.010107	1430	1343	87
36	20035	967.7998	945.386	22.41381	0.35256667	0.36087593	−0.008309	1401	1354	47
37	20036	945.59716	928.8247	16.77242	0.37289348	0.37922439	−0.006331	1358	1309	49

	Seed2	消費者余剰		差	平均購入価格		差	非購入者数		差
		(a)	(d)		(a)	(d)		(a)	(d)	
38	20037	958.31793	943.7042	14.61377	0.34979966	0.35851073	−0.008711	1514	1432	82
39	20038	946.91224	933.1795	13.73271	0.36875188	0.37558002	−0.006828	1431	1385	46
40	20039	881.16778	860.2844	20.88336	0.3934981	0.40141792	−0.00792	1402	1339	63
41	20040	937.39611	922.6345	14.76163	0.36647263	0.37175655	−0.005284	1360	1329	31
42	20041	961.04437	947.4258	13.61852	0.36673172	0.37298866	−0.006257	1379	1346	33
43	20042	917.30758	917.4595	−0.15191	0.37001989	0.37482657	−0.004807	1419	1385	34
44	20043	944.44124	928.9085	15.53277	0.3691638	0.37497488	−0.005811	1399	1356	43
45	20044	989.07262	971.5977	17.47494	0.35970957	0.36733501	−0.007625	1341	1284	57
46	20045	949.02602	958.703	−9.67703	0.37424651	0.37544363	−0.001197	1259	1270	−11
47	20046	960.70068	945.3448	15.35589	0.36146746	0.36892044	−0.007453	1399	1353	46
48	20047	1022.0571	1003.492	18.56507	0.35114391	0.35675313	−0.005609	1350	1305	45
49	20048	1060.9098	1042.872	18.03737	0.34391478	0.35083133	−0.006917	1315	1263	52
50	20049	964.5623	957.3894	7.17285	0.36245401	0.36663945	−0.004185	1334	1317	17
			平均	13.69923		平均	−0.006377		平均	44.1
			標準偏差	11.19089		標準偏差	0.0027416		標準偏差	29.29727

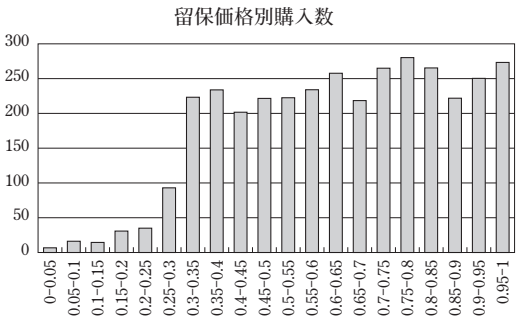
余剰	階　級	度数分布
－0	0	7
1－2	2	2
2－4	4	2
4－6	6	0
6－8	8	4
8－10	10	4
10－12	12	1
12－14	14	4
14－16	16	4
16－18	18	6
18－20	20	4
20－22	22	1
22－24	24	3
24－26	26	2
26－28	28	1
28－30	30	1
30－32	32	1
32－34	34	0
34－36	36	2

消費者余剰差の分布



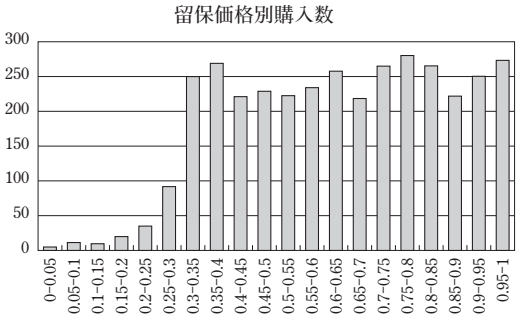
留保価格別購入件数 Seed2＝20000，検索ミス無し

0–0.05	0.05	7
0.05–0.1	0.1	17
0.1–0.15	0.15	15
0.15–0.2	0.2	31
0.2–0.25	0.25	36
0.25–0.3	0.3	93
0.3–0.35	0.35	224
0.35–0.4	0.4	236
0.4–0.45	0.45	203
0.45–0.5	0.5	222
0.5–0.55	0.55	224
0.55–0.6	0.6	236
0.6–0.65	0.65	258
0.65–0.7	0.7	221
0.7–0.75	0.75	267
0.75–0.8	0.8	282
0.8–0.85	0.85	267
0.85–0.9	0.9	222
0.9–0.95	0.95	252
0.95–1	1	274



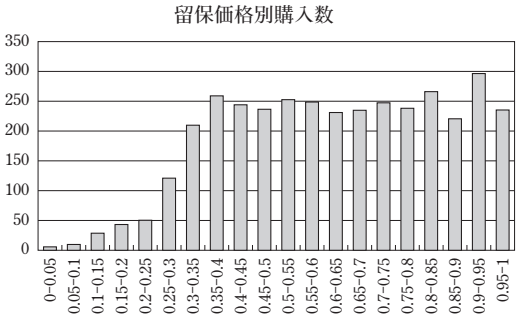
留保価格別購入件数 Seed2＝20000，検索ミス50%

0–0.05	0.05	6
0.05–0.1	0.1	13
0.1–0.15	0.15	12
0.15–0.2	0.2	22
0.2–0.25	0.25	35
0.25–0.3	0.3	93
0.3–0.35	0.35	251
0.35–0.4	0.4	268
0.4–0.45	0.45	224
0.45–0.5	0.5	229
0.5–0.55	0.55	224
0.55–0.6	0.6	236
0.6–0.65	0.65	258
0.65–0.7	0.7	221
0.7–0.75	0.75	267
0.75–0.8	0.8	282
0.8–0.85	0.85	267
0.85–0.9	0.9	222
0.9–0.95	0.95	252
0.95–1	1	274



留保価格別購入件数 Seed2＝20018，検索ミス無し

0–0.05	0.05	4
0.05–0.1	0.1	9
0.1–0.15	0.15	29
0.15–0.2	0.2	44
0.2–0.25	0.25	51
0.25–0.3	0.3	122
0.3–0.35	0.35	210
0.35–0.4	0.4	258
0.4–0.45	0.45	246
0.45–0.5	0.5	237
0.5–0.55	0.55	252
0.55–0.6	0.6	250
0.6–0.65	0.65	231
0.65–0.7	0.7	234
0.7–0.75	0.75	249
0.75–0.8	0.8	239
0.8–0.85	0.85	268
0.85–0.9	0.9	222
0.9–0.95	0.95	295
0.95–1	1	237



留保価格別購入件数 Seed2＝20018，検索ミス50%

0-0.05	0.05	3
0.05-0.1	0.1	7
0.1-0.15	0.15	23
0.15-0.2	0.2	43
0.2-0.25	0.25	38
0.25-0.3	0.3	113
0.3-0.35	0.35	238
0.35-0.4	0.4	270
0.4-0.45	0.45	263
0.45-0.5	0.5	240
0.5-0.55	0.55	252
0.55-0.6	0.6	250
0.6-0.65	0.65	231
0.65-0.7	0.7	234
0.7-0.75	0.75	249
0.75-0.8	0.8	239
0.8-0.85	0.85	268
0.85-0.9	0.9	222
0.9-0.95	0.95	295
0.95-1	1	237