11

Human and Artificially Intelligent Traders in Computer Double Auctions

Dhananjay K. Gode
University of Rochester

Shyam Sunder
Carnegie Mellon University

Why study artificial intelligence in computer-simulated competitive markets? Our study is an attempt to identify those performance characteristics of double auctions[1] that are consequences of their structure, from those that result from behavior of participating traders. The longer term goal of this effort is to understand the linkage between individual decisions in market settings on one hand, and aggregate market behavior on the other. Artificial intelligence (AI) appears to be a promising tool to study this linkage.

In this chapter, we report on three matched sets of computerized double auctions among buyers and sellers with exogenously given redemption value and cost schedules. Each set includes three auctions of six periods each as follows:

1. An auction involving human traders.
2. An auction involving artificially intelligent (i.e., program or AI) traders designed by the human traders who participated in the human trader auctions.
3. An auction involving "zero-intelligence" (ZI) computer traders.

[1]Double auction is a form of market organization in which buyers as well as sellers can make and accept public proposals. Buyer proposals, called bids, are made for a specified price and quantity. Similarly, seller proposals are called offers or asks. All proposals can be improved upon by any trader. Acceptance of a proposal by another trader results in a transaction. Each transaction is final; there is no recontracting.

The first auction in each set involved 13 human traders and was no different from thousands of other auctions that have already been reported in the literature. The participants in these auctions were then asked (after being presented with an introduction to trading in double auction and to the programming environment) to write and submit a computer program to trade on their behalf. The programs were submitted before their writers had the knowledge of the market parameters, and were debugged with the help of researchers. The second auction in each set was populated by these 13 AI or program traders.

Finally, zero-intelligence (ZI) computer buyers randomly generated a bid that was uniformly distributed between the redemption value of the unit the buyer wished to trade and the current bid, provided that the former exceeded the latter. Redemption value limit prevented these buyers from trading at a loss; the current bid limit prevented them from generating futile bids. Aside from these two limits, these programs made no other use of information, and did not learn either within a period of trading, or across periods. The ZI computer sellers are defined analogously, generating random asks that are uniformly distributed between cost of the current unit at the lower end and the current ask at the upper end.

The ZI traders make no explicit attempt to maximize their profits from trading; they only avoid money-losing trades. They submit a bid (or ask) every time they are prompted to do so, and thus, on occasion, even bid against themselves. They do not observe the market prices of the current or the past periods. They do not use the bid-ask data except to avoid making futile bids. They have no memory and do not alter their behavior in light of experience. Performance characteristics of a trading institution that is populated by ZI traders can properly be attributed to the institution itself, and not to the rationality or maximizing behavior of the participating traders. We use the performance characteristics of double-auction markets populated with ZI traders as the datum against which the performance of markets with intelligent traders—whether human or artificial—can be compared; see Gode and Sunder (1992, 1993a, 1993b) for further work on this topic.

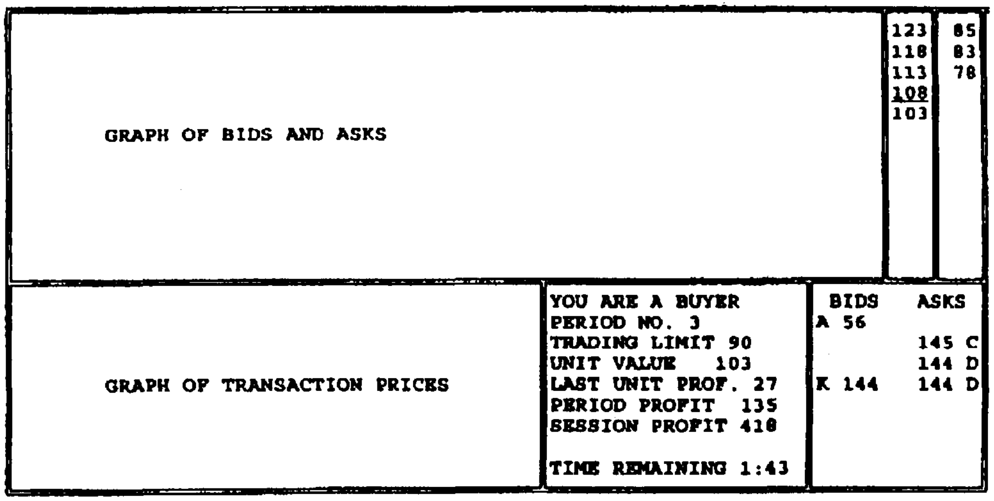
TRADING ENVIRONMENT

At the beginning of each period, every buyer was endowed with a right to buy up to a specified number of units (one in each transaction) at any price between $0 and $200. The redemption value of each unit was guaranteed to be no greater than the redemption value of the preceding unit. Also at the beginning of each period, sellers were endowed with a right to sell up to a specified number of units (one in each transaction) at any price between $0 and $200. The cost of each unit was guaranteed to be no less than the cost of the preceding unit. Traders could (but rarely did, except when they made occasional keyboard errors) enter into money-losing trades. To engage motivation, realized profits of student traders entered into their course grade as a percentage of the equilibrium profit.

The trading screen used in Carnegie Mellon University's (CMU) MARKET-2001 computer double auction is shown in Fig. 11.1. All programs are written in Borland's Turbo Pascal. Most of the upper half of the screen is taken up by a dynamic real-time point graph of bid and offer prices punctuated by vertical lines that indicate that a bid–ask sequence has been terminated and another one started by conclusion of a transaction. On the computer screen, bids appear in white, asks in cyan, and transaction lines in green on a blue screen. The bid, ask, or the transaction line on the screen of the traders who generated it appears in red. The first column on the right-hand side of the bid–ask screen shows the redemption values for buyers and unit costs for sellers, with a red cursor highlighting the value or cost of the current unit being traded. As the trader enters into transactions, the prices of these transactions appear in the right-hand column against the value or cost of the corresponding unit. The lower left window of the trading screen shows a line graph of transaction prices. This is followed by an accounting and timer window in the middle, and a scrolling ticker window to the right.

CMU's double auction program executes crossing bids and offers at the price of the earlier of the two quotes. Bids and offers remain valid only until a transaction occurs, at which time they become void if not executed.

The program trading interface transfers control to the trading program (which is actually a Turbo Pascal procedure) whenever it is not processing



```
123   85
118   83
113   78
108
103

YOU ARE A BUYER            BIDS    ASKS
PERIOD NO. 3               A 56
TRADING LIMIT 90                   145 C
UNIT VALUE    103                  144 D
LAST UNIT PROF. 27         K 144   144 D
PERIOD PROFIT  135
SESSION PROFIT 418

TIME REMAINING 1:43
```

FIG. 11.1. Trading screen of Market 2001.

input from the trading program or from the central control (see Fig. 11.2 for system description). The trading program can respond by choosing one of three possible actions. If the program is assigned the role of a buyer, it can submit a bid at a specified price, or submit a "take" of the outstanding ask, or submit "no action." Similarly, if the program is assigned the role of a seller, it may choose to submit an ask at a specified price, or a "sell" to an outstanding bid, or submit "no action." The trading program has access to individual information (such as redemption values or costs, number of units it can trade, accounting information about profits) and market information

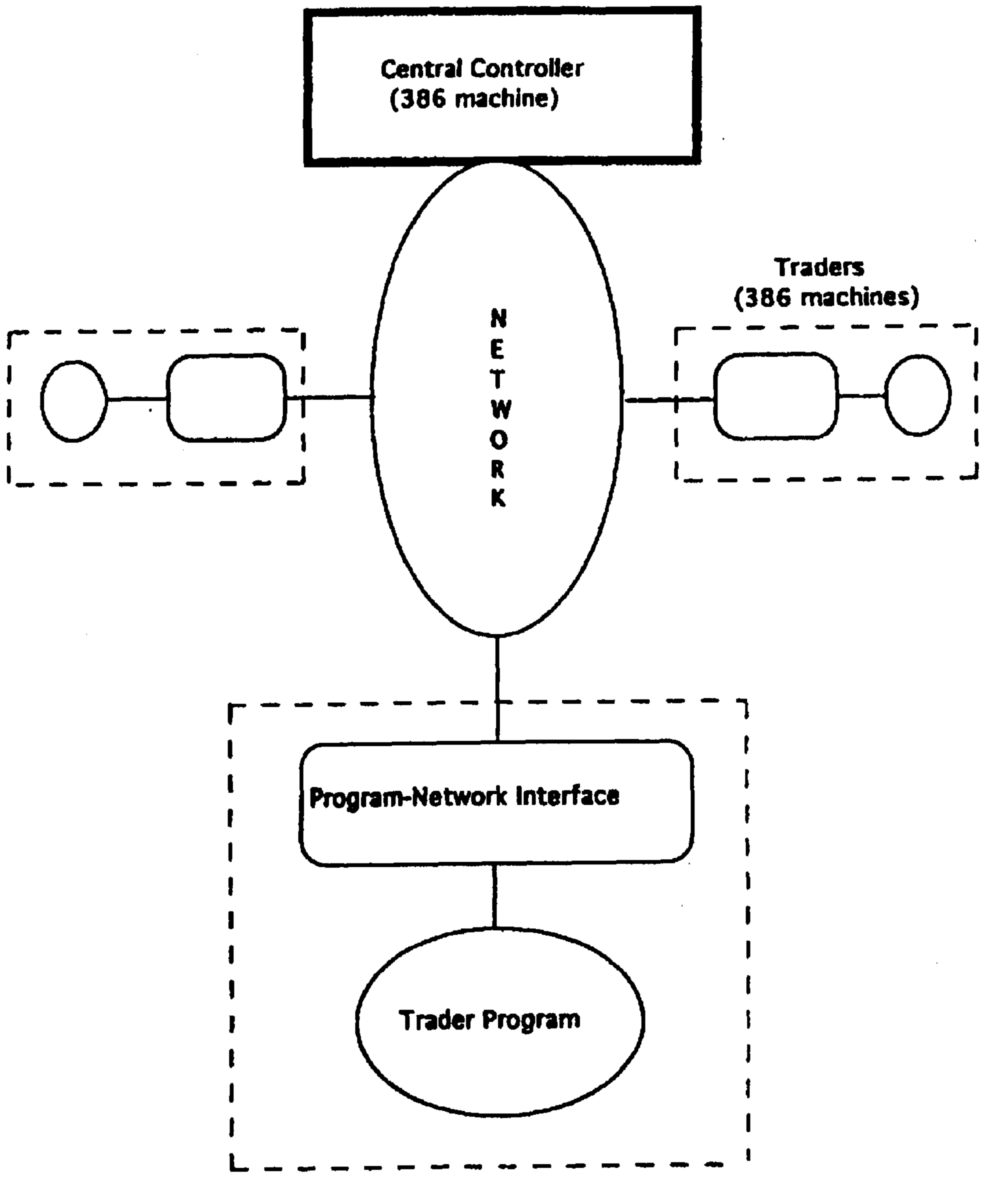


FIG. 11.2. System description.

(such as prices, bids, offers, identities of traders taking various actions, time labels attached to various actions, and current time). In addition, the trading programs can utilize some packaged procedures that help retrieve the past data and compute summary statistics. Details of these variables and programming tools are given in Appendix A.

DECISION RULES AND TRADING PROGRAMS

The 13 trading programs whose performance is reported in this chapter used a variety of trading strategies. The programs are relatively short (150 lines of Pascal code on average) but not always simple in their logic. In Appendix B we provide a flavor of the range of the logical structure of the buyer programs. An analogous description could be prepared for the structure of the seller programs.

DESIGN OF EXPERIMENTS

Table 11.1 shows the buyer redemption values and seller costs for each of the four markets. Also shown are the equilibrium price, equilibrium volume, and equilibrium profits for buyers and sellers in each market. We used the same four sets of parameters for human, program, and ZI trader markets. The only difference was that in the first two human trader markets, there were seven buyers and six sellers; in all other human and in all program and ZI markets, there were six buyers and seven sellers. Parameters were chosen to yield a broad range of equilibrium prices (from $69 in Market 2 to $170 in Market 4) and volumes (from 6 in Market 3 to 28 in Market 1). In all cases, the equilibrium price was unique. All markets were run for six periods of 8 to 4 min each for human traders and 2 min for program and ZI traders (see Friedman & Sunder, 1994, for design of economics experiments).

RESULTS AND ANALYSIS

Prices

The three panels of Fig. 11.3 show the transaction price charts for Market 1 operated with human, program, and ZI traders, respectively. Figures 11.4, 11.5, and 11.6 provide similar charts for the other three markets. After some initial uncertainty, the prices in human trader markets quickly converge to the neighborhood of the equilibrium price in all instances.

TABLE 11.1
Design of Markets

Unit #1	1	2	3	4	5	6	7	8	9	10	Human	P	Q	π	π_T	Program	P	Q	π	π_T
Buyer values	102	97	92	87	82	77					Manual 1	82	28	50	1070	Auto 1	82	28	50	1070
Seller costs	34	46	58	70	82	94					7B, 6S			120		6B, 7S			120	
Buyer values	117	105	93	81	69	57					Manual 2	69	28	120	1140	Auto 2	69	28	120	1140
Seller costs	49	54	59	64	69	74					7B, 6S			50		6B, 7S			50	
Buyer values	133	95	90								Manual 3	95	6	38	263	Auto 3	95	6	38	263
Seller costs	90	95	100								B6, S7			5		6B, 7S			5	
Buyer values	180	175	170	165	160						Manual 4	170	12	15	860	Auto 4	170	12	15	860
Seller costs	90	140	170	190	199						B6, S7			110		6B, 7S			110	

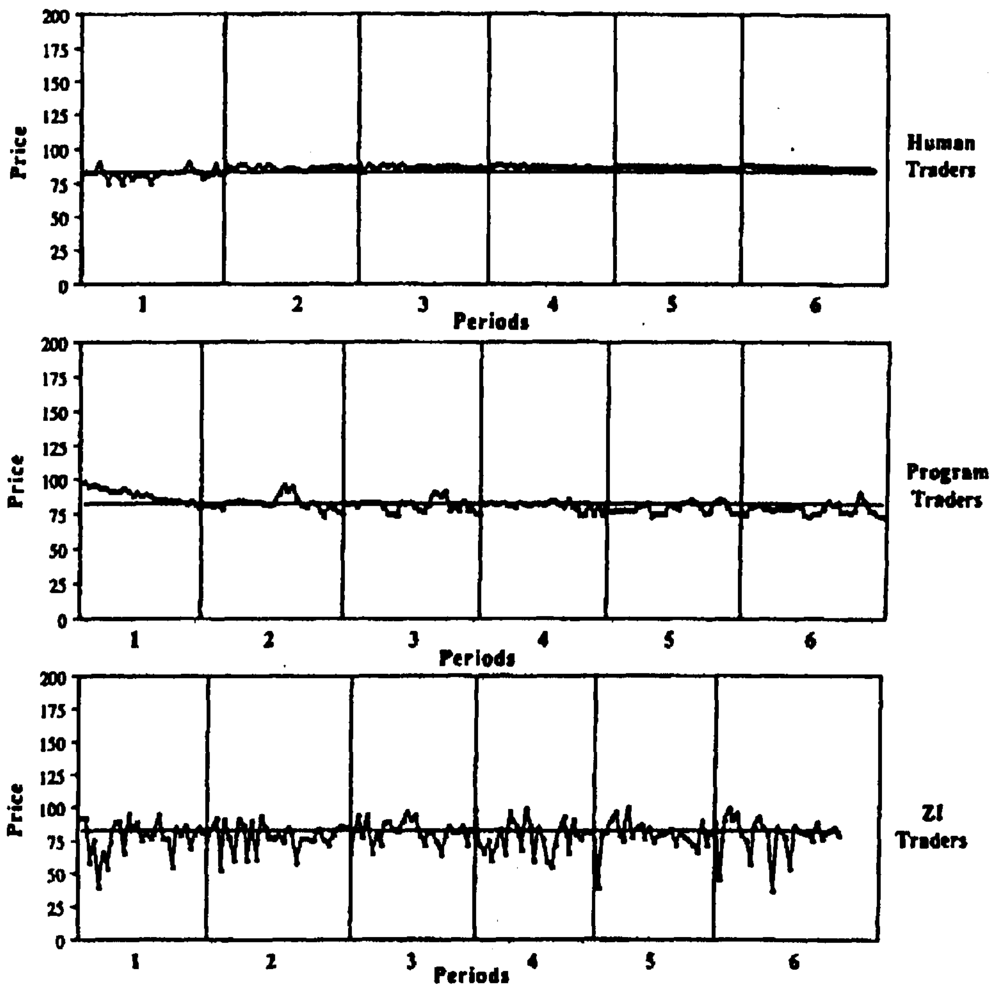


FIG. 11.3. Transaction prices, Market 1.

These markets are characterized by the remarkable stability of price and volume.

In comparison, the prices in program trader (AI) markets are not as stable as in the human trader markets, but are more stable than in the ZI markets. The approach to equilibrium prices in AI markets is slower than in the human trader markets but faster than in the ZI markets. Even the ZI market prices converge to the neighborhood of equilibrium in the last few transactions. However, the ZI markets are characterized by persistent volatility of price within all periods and across all periods. This should not be surprising because, unlike the human traders and AI traders, ZI traders have no ability either to observe or to learn from the market phenomena. Their behavior is statistically identical across all periods of a market.

Late, but eventual, convergence of ZI markets to the neighborhood of equilibrium prices is particularly noteworthy. In these markets this convergence does not take place because the traders try to maximize their profits, nor because they learn and remember the market prices—by construction,

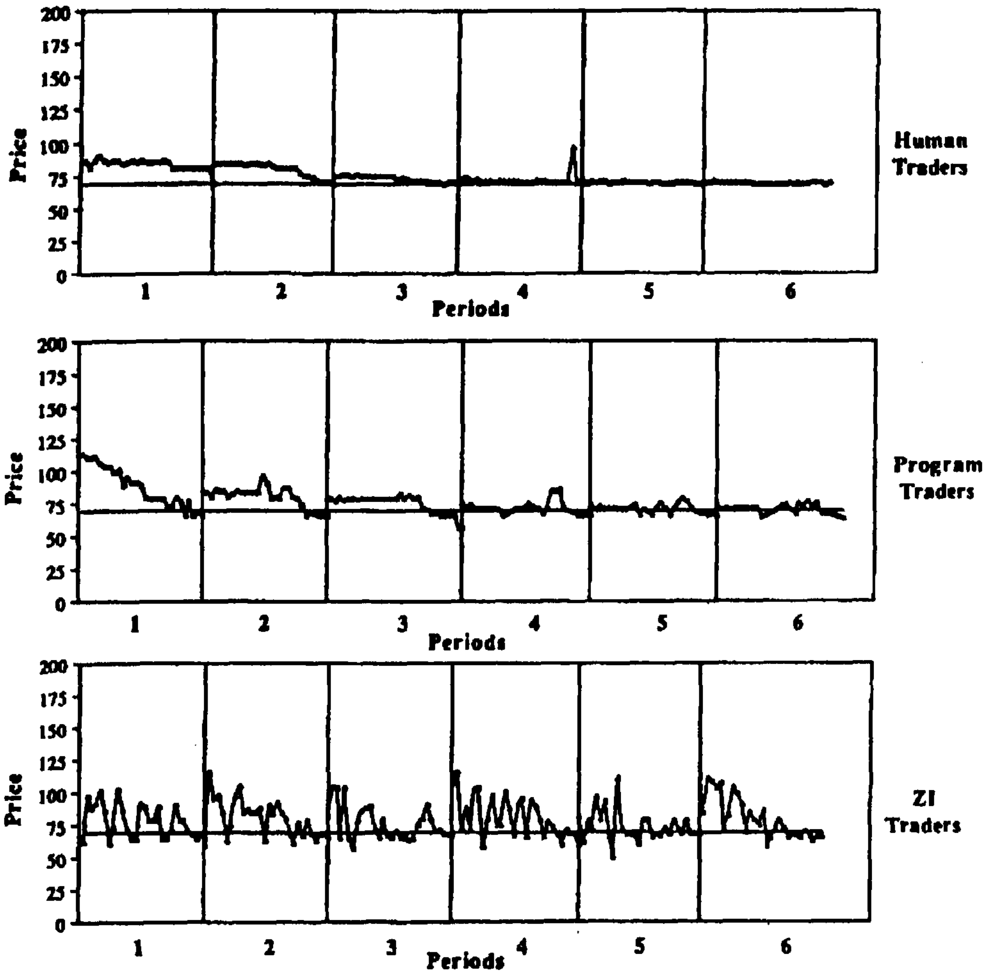


FIG. 11.4. Transaction prices, Market 2.

ZI traders are incapable of such behavior. It happens simply because the higher valued units in the hands of the buyers and the lower cost units in the hands of the sellers are exhausted first. In later parts of each trading period, the units traded have redemption values only slightly higher than their cost to the sellers. This tightening of the range in which profitable transactions are feasible funnels the transaction prices toward equilibrium in the later part of each period. It follows, then, that if the difference between the cost and value of the last units traded is large, the ZI markets will not converge as precisely to the equilibrium level.

Bids and Offers

Figures 11.7-11.10 show the average number of bids and offers made by various traders per completed transaction in the market. Human trader

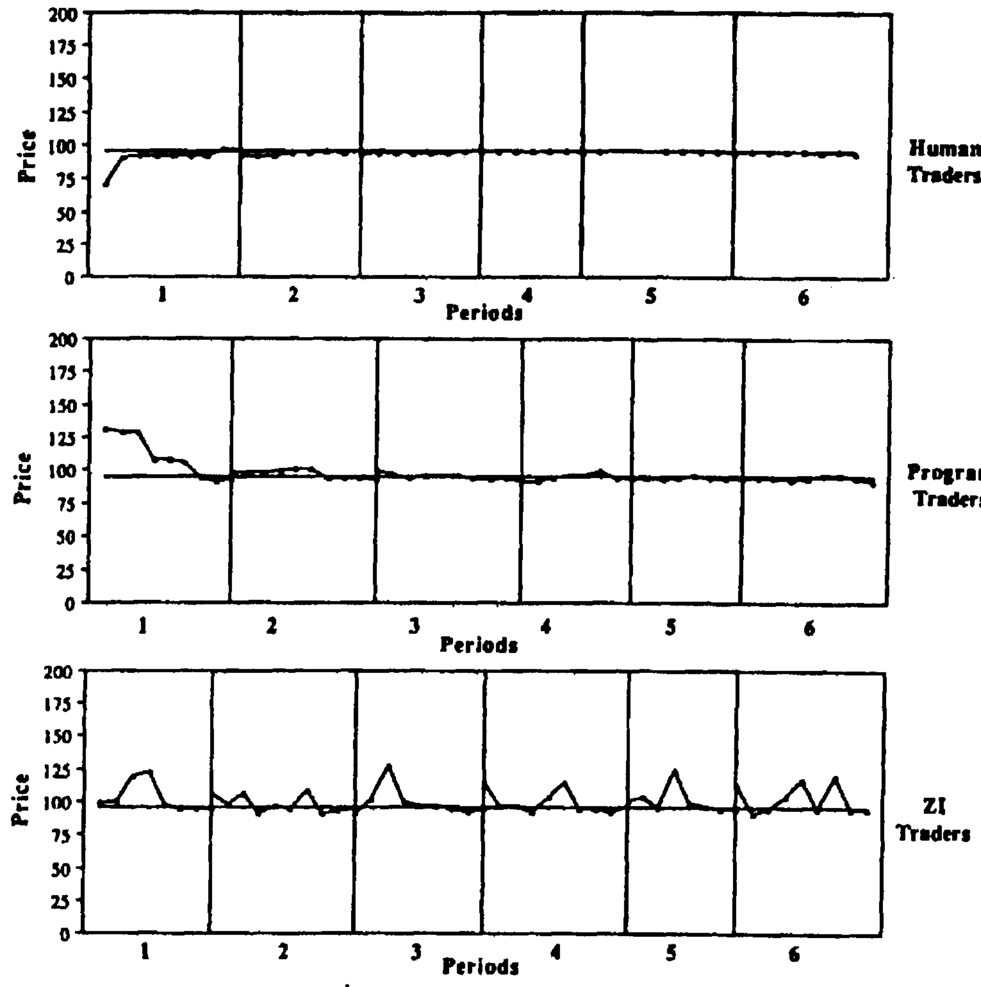


FIG. 11.5. Transaction prices, Market 3.

markets are especially efficient in executing their trades with a minimum amount of activity and effort as measured by the bids and offers. This number remains in the neighborhood of 4–5 for human trader markets. The AI markets, on the other hand, are particularly inefficient in utilizing bids and offers because they seem to need approximately 15 bids and offers to consummate a transaction. The ZI markets are surprisingly more efficient than AI markets in this respect, using only 8–10 messages per transaction.

Bids and asks per transaction can be taken as a measure of how efficiently the communications resources of the market environment are utilized by the traders. Markets with intelligent traders (human and program) become generally more efficient in utilizing communications as they gain experience from trading in earlier periods. Without any capacity to learn, the ZI traders show no such tendency.

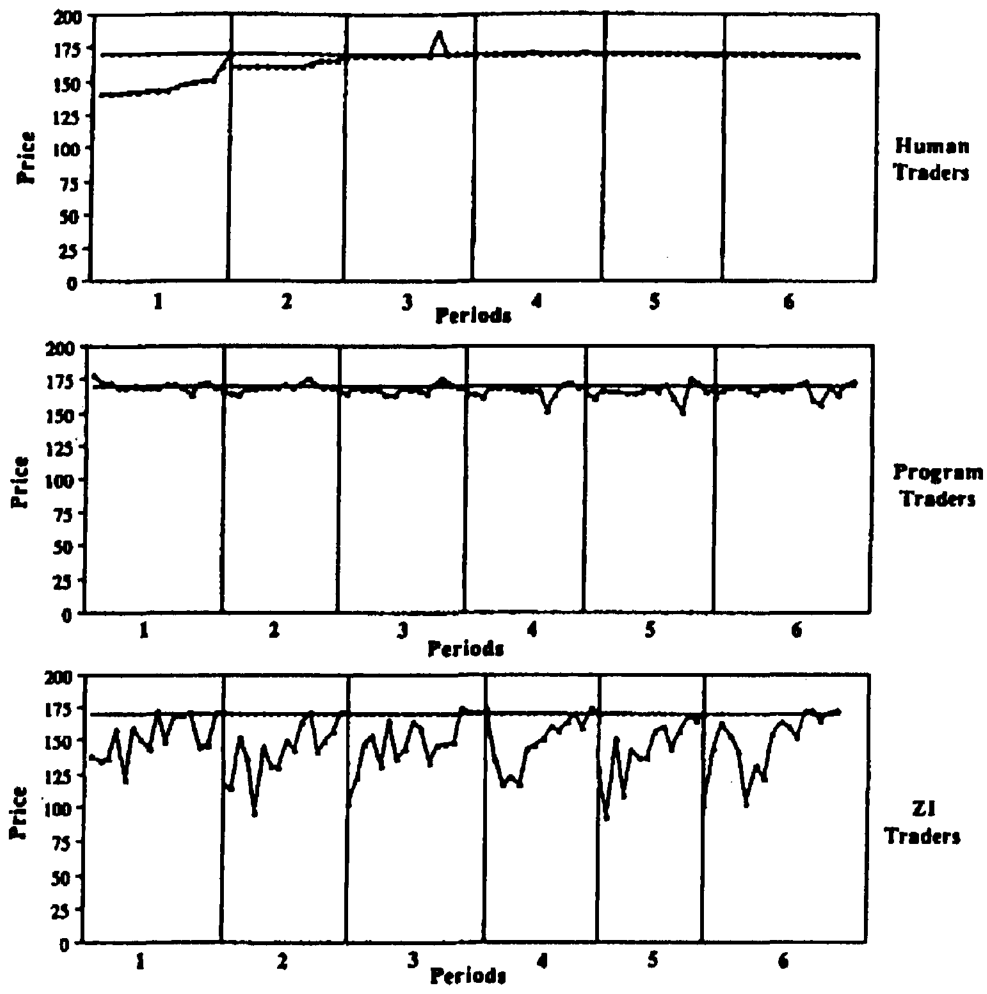


FIG. 11.6. Transaction prices, Market 4.

In the market environment reported here, communications were available for free. The only cost incurred by the traders was the cost of entering the message into the system. This entry cost, given the limitations of the keyboard entry, may be higher for the human traders than for program traders, and may explain the behavior observed in these markets. If the traders were charged a price for entering each bid or ask, perhaps a better measure of the economic trade-offs in utilizing the communication resources of the market environment could be developed.

Distribution of Profits and Efficiency

Figures 11.11–11.14 show the efficiency of Markets 1–4 plotted against the cross-sectional coefficient of variation of individual profits of traders.

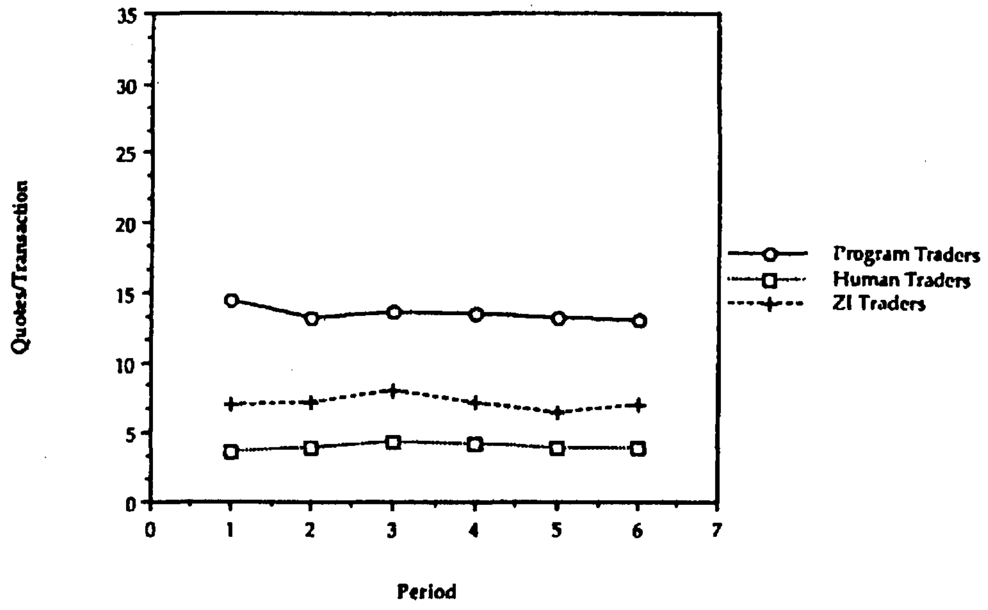


FIG. 11.7. Bids and offers per transaction, Market 1.

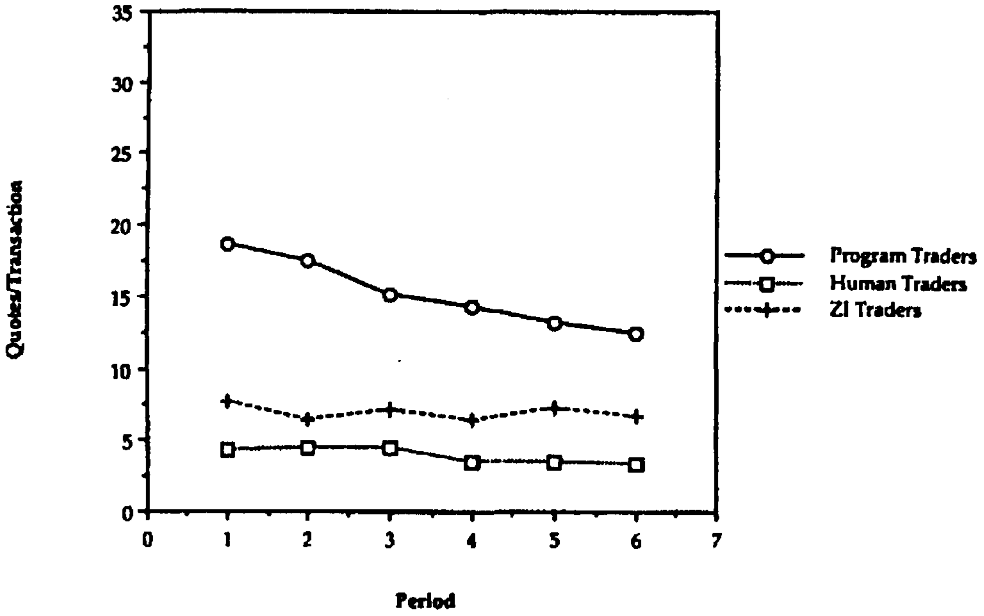


FIG. 11.8. Bids and offers per transaction, Market 2.

Efficiency is the total profit of all traders divided by the sum of consumer and producer surplus for the market. Because the equilibrium profits for buyers and sellers are often quite different in these markets, the coefficient of variation (sample standard deviation divided by sample mean) was computed separately for the buyers and for the sellers, and the mean of those two numbers is presented here as the profit coefficient of variation.

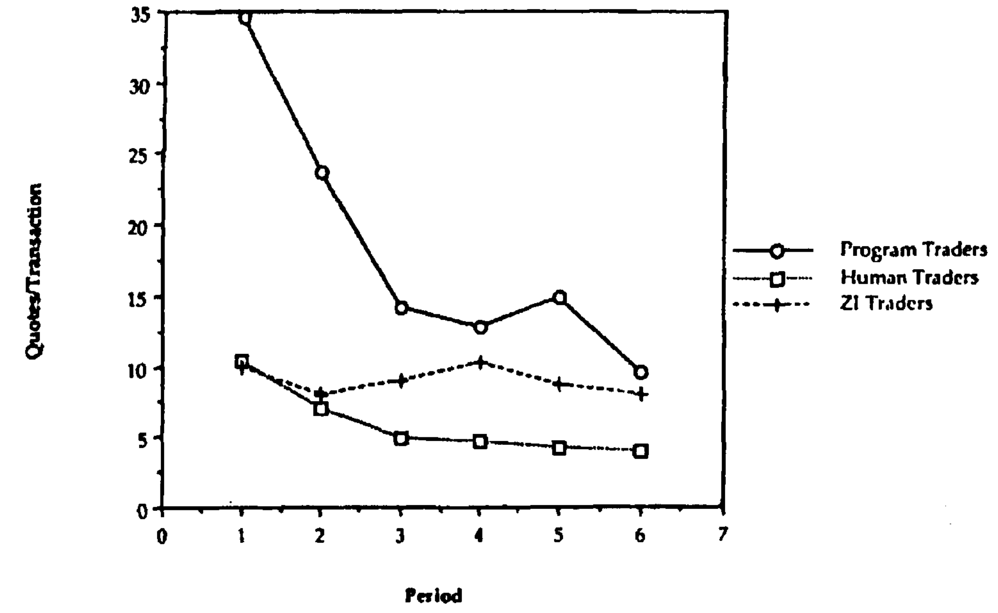


FIG. 11.9. Bids and offers per transaction, Market 3.

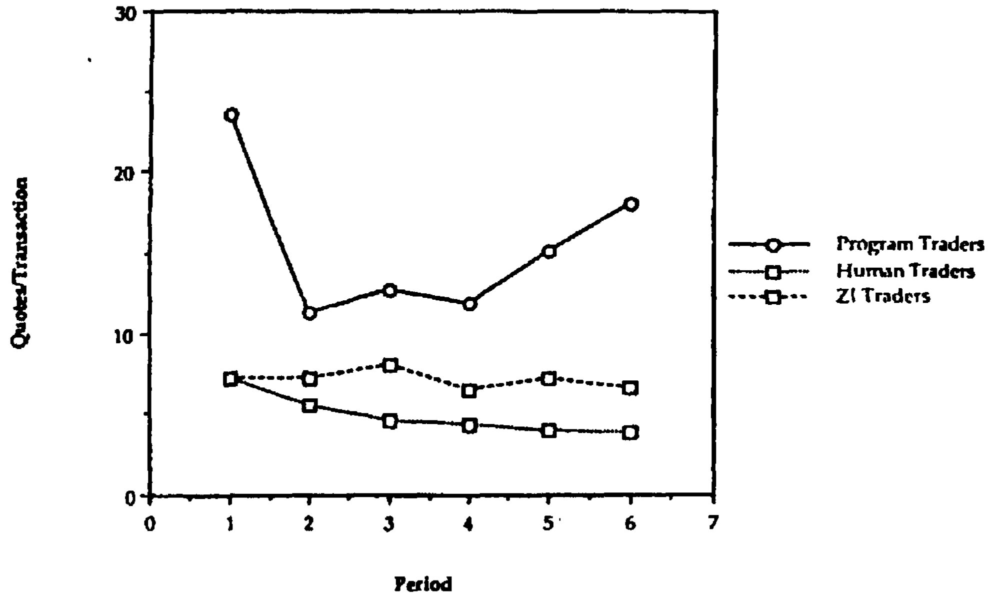


FIG. 11.10. Bids and offers per transaction, Market 4.

The efficiency of human trader markets (shown by hollow squares on the charts) is 100% in most periods with occasional shortfall of 1 or 2 percentage points, mostly in the first period of the markets. Efficiency of the AI markets (shown by hollow circles on the charts) is very close to 100%, but generally falls 1 or 2 percentage points short of the efficiency of the human trader markets. The same is true of the efficiency of the ZI

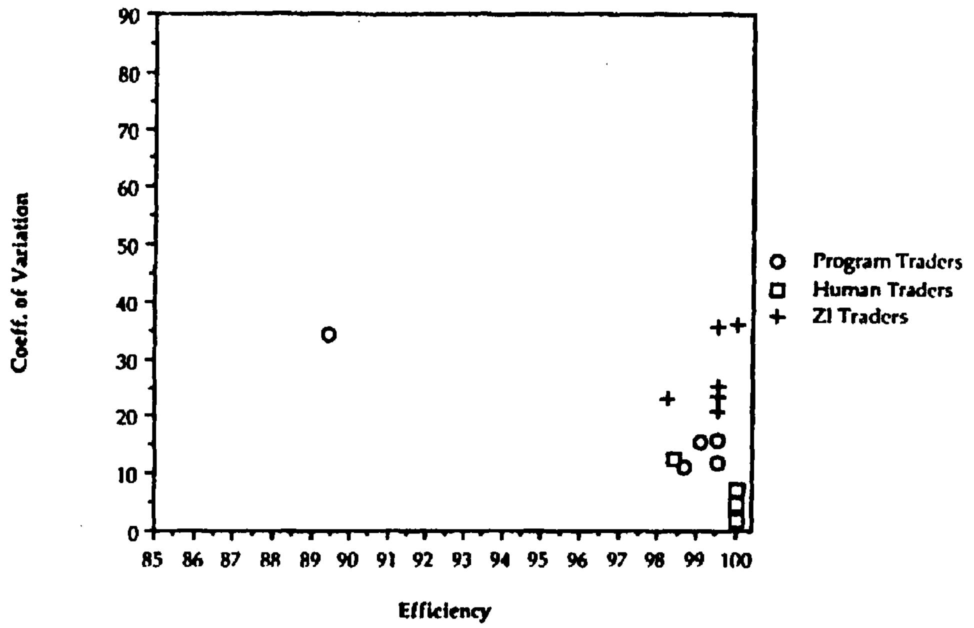


FIG. 11.11. Efficiency versus profit coefficient of variation, Market 1.

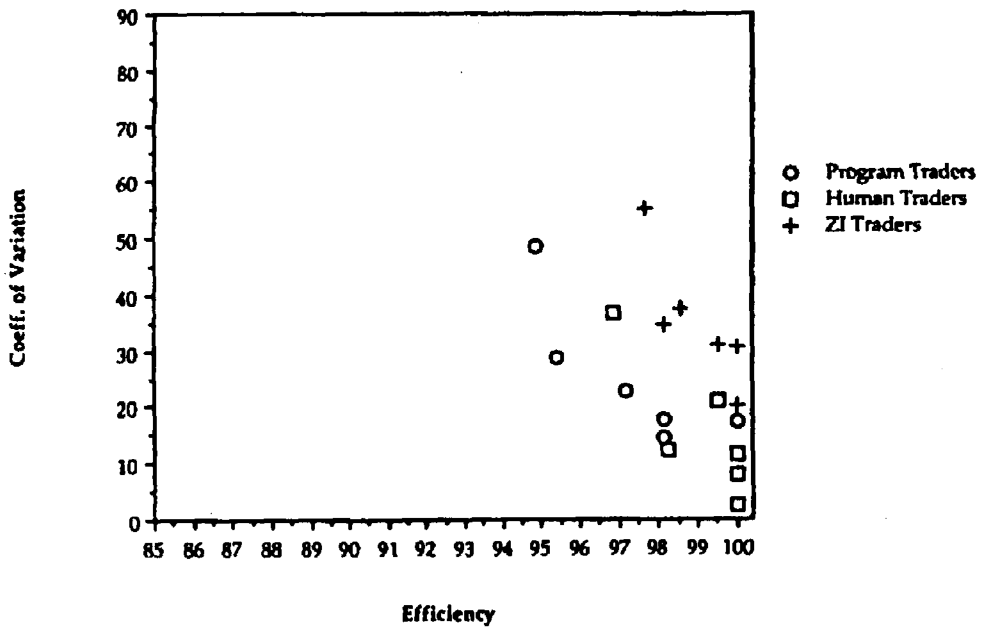


FIG. 11.12. Efficiency versus profit coefficient of variation, Market 2.

markets—they too are virtually 100% efficient, and almost indistinguishable from the human markets by efficiency criterion.

Although there are no significant differences in the ability of the human, AI, and ZI markets to exploit the total surplus in these double auction markets, there are significant differences in the way this total surplus is

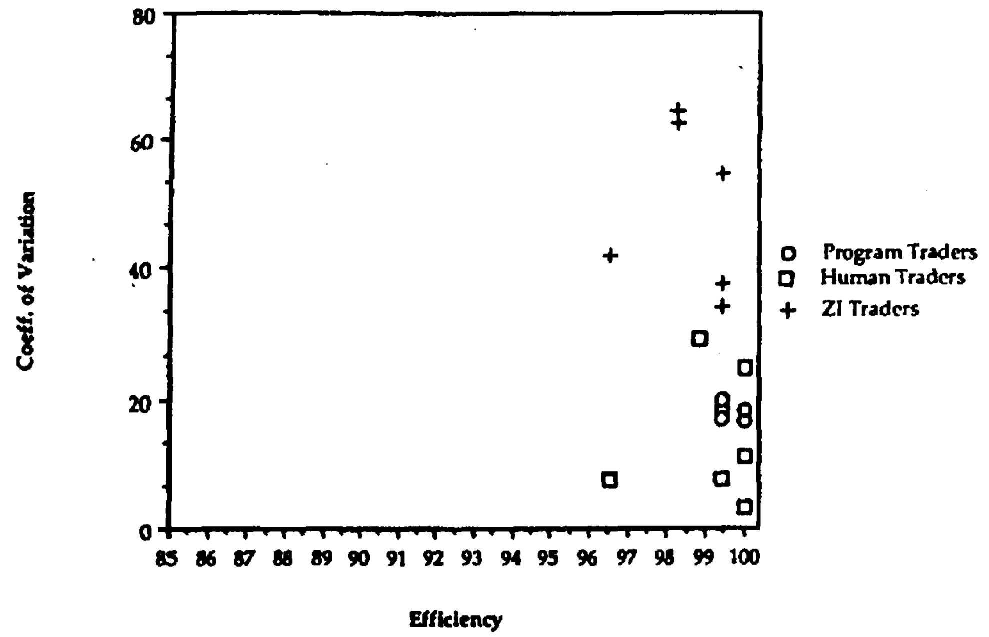


FIG. 11.13. Efficiency versus profit coefficient of variation, Market 3.

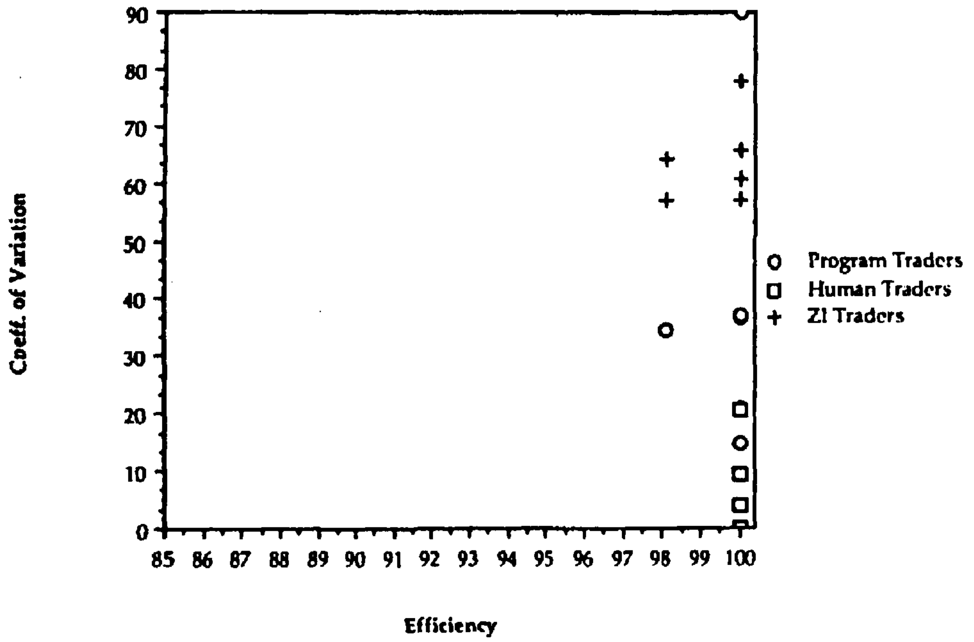


FIG. 11.14. Efficiency versus profit coefficient of variation, Market 4.

distributed among the individual traders. The profit coefficient of variation is the lowest for the human trader markets.[2] In AI markets, profits were less evenly distributed, and this dispersion was the highest in the ZI markets. It is easy to distinguish the degree of intelligence in the markets by the dispersion of individual profits but not by the magnitude of total profits.

CONCLUSIONS

In double auctions with a unique equilibrium price, relatively simple AI traders can achieve convergence to equilibrium price and virtually 100% efficiency. Indeed, little intelligence is necessary for achieving either of these two goals. Markets populated by zero-intelligence traders that make no attempt to maximize profits, and have no power to observe, learn, or remember do almost as well in efficiency and converge to the proximity of equilibrium price, although not as rapidly or as smoothly as the markets with human and AI traders.

Double auction markets populated by human traders do, however, exhibit more efficient utilization of the communications resources of market environment by executing transactions with fewer bids and asks per transaction. They also exhibit superiority in more even distribution of profits across traders as compared to markets with AI and ZI traders.

The ability of the double auctions to yield virtually 100% of the surplus to ZI traders suggests that this ability may be a consequence of the structure of the double auction itself, and is possibly independent of the trader behavior (or capability). The ability of traders to observe, remember, and learn does not seem to affect the efficiency of the simple double auctions we have examined so far with artificially intelligent (and unintelligent) traders. We already know that in certain double auctions, human traders significantly improve their ability to exploit surplus upon replication and experience (see Plott & Sunder, 1982, 1988). Much more work would be needed to delineate boundaries between those performance characteristics of double auctions that arise from their structure and those that arise from the purposive human or artificially intelligent behavior.

These preliminary results leave us with two other tentative thoughts. First, in experimental economics literature, percentage of the surplus exploited has often been used as an index of learning and rationality of subjects, as well as of attainment of control in an experimental economy. Such inferences may not be appropriate for market mechanisms that yield

[2]An occasional large value of this coefficient in human markets occurred because a keyboard error by a trader caused a transaction to take place at a price far removed from the equilibrium and adjacent transaction prices.

all their surplus to zero-intelligence traders. Second, if it is true that surplus exploitation is a property of double (and perhaps other kinds of) auctions independent of individual behavior, the behavioral critique of the rationality assumptions of economics may need a reexamination.

ACKNOWLEDGMENTS

Earlier drafts of this chapter were presented at the meetings of the Economic Science Association and the Institute of Management Sciences' Workshop on Mathematical Organizational Theory. We have benefited from many comments received at these and other presentations. Financial support of this work from Margaret and Richard M. Cyert Family Funds and Information Networking Institute is gratefully acknowledged.

APPENDIX A
LIST OF INFORMATION VARIABLES AND TOOLS AVAILABLE TO PROGRAM TRADERS

This appendix describes the variables as a segment of a Pascal program presented to the students; it can be read by the trader programs to get information about the market activity and status, and about their own performance and status.

Const

MaxPeriod = 10	The maximum number of periods in a session.
MaxTrade = 75	The upper limit of the number of individual buyer's or seller's trades that the system is designed to handle.

Var

Myid	char	Terminal_ID of your computer (may take values from 'A' to 'Q').
ControllerID	char	Terminal_ID of the central system, normally set to 'M.'
Asker	char	Terminal_ID of the player making the current ask.
CurrentAsk	longint	Current asking price at the local system.
Bidder	char	Terminal_ID of the player making the current bid.
CurrentBid	longint	Current bidding price at the local system.

CurrentTransNo	integer	= *n* when the *n*th unit is being transacted in the market. For the first transaction number it is 1, and it is incremented by 1 each time a transaction occurs in the market. Remember this variable counts the number of units in the market as a whole and not the number of your transactions. See rcount.
Rcount	integer	= *n* where you are bidding or asking for your *n*th unit, after having transacted *n*-1 units. Remember this is your personal transaction count.
Info.TradeLimit	integer	This indicates the maximum number of units you can buy (if you are a buyer) or sell (if you are a seller). Please remember that Info.TradeLimit −Rcount +1 is the number of units remaining in your hand.
Buyer	boolean	"TRUE" if you are a buyer.
Seller	boolean	"TRUE" if you are a seller.
LocalRejectFlag	boolean	This is set to "TRUE" if your previous request was rejected locally.
ControllerReject-Flag	boolean	This is set to "TRUE" if your last request was rejected by the central system (controller).
remainmin	integer	Number of minutes remaining in the current period.
remainsec	integer	Number of seconds remaining in the current period.
RemainTotSec	longint	Total seconds remaining in the period.
PeriodTotSec	longint	Total seconds in the period.
Info.PeriodCount	integer	This variable stores the current period number. This is set to 1 at the start of the first period and incremented by 1 at the beginning of every period.
Info.PeriodMinutes	integer	This indicates the length of each period in minutes. It is set at the beginning of every session.
Info.TradeCount	array	[1..MaxPeriod] of integer. This indicates the number of trades you had during each period—that is, Info.TradeCount[1] is the number of trades you had in the first

		period, Info.TradeCount[2] is the number of trades you had in the second period.
Info.RpriceArray	array	[1..MaxTrade] of longint. This array holds your values or costs depending on whether you are a buyer or seller respectively. Info.RpriceArray[rcount] gives the value or cost of the unit that you are currently supposed to trade.
Info.TotalProfit	longint	This variable stores the profits made by you since the beginning of the session.
Info.PeriodProfit-Array	array	[1..MaxPeriod] of longint. Stores the profit for each period.
LastTransProfit	longint	Profit made in the previous transaction.
LastBidTime	integer	Time at which the last bid was made.
LastAskTime	integer	Time at which the last ask was made.

In addition to the variables just listed, the following Pascal procedures are also available to the program traders:

1. Access.

```
Procedure Access(Period_no, Trans_no:longint; var Error: integer)
```

This procedure helps locate the pointer for the data file (the transactions file) where bids, asks, and transactions data are stored. It requires two inputs—period number and transaction number. A call to the procedure positions the file pointer in the transactions file (which is maintained on disk) to the record where the data for the specified transaction is stored. The procedure also returns an error code with specified interpretations.

2. PeriodSummary. The structure of the Pascal record is as follows:

```
PeriodSummaryRecType = record
OpeningPrice:      longint;
ClosingPrice:      longint;
Maximum:           longint;
Minimum:           longint;
Mean:              longint;
StdDeviation:      longint;
MaxTrans_no:       integer;
MinTrans_no:       integer;
NoOfTransactions:  integer;
MaximumBid:        longint;
```

```
MinimumAsk:         longint;
LastUnacceptedBid:  longint;
LastUnacceptedAsk:  longint;
end;

PeriodSummaryArrayType = array[1 .. MaxPeriod] of PeriodSum-
maryRecType; { MaxPeriods = 10}
PeriodSummaryArray: PeriodSummaryArrayType;
```

3. TransactionSummary.

```
TransactionSummaryRecType = record
Price:              longint;
OpeningBid:         longint;
OpeningAsk:         longint;
AverageBid:         longint;
AverageAsk:         longint;
NumberOfBids:       integer;
NumberOfAsks:       integer;
TimeTaken:          integer;
end;

TransactionSummaryArrayType = array  [1..MaxTransactions]  of
TransactionSummaryRecType;
TransactionSummaryArray: TransactionSummaryArrayType;
```

To access the Transaction summary of the past period, one has to call the Procedure *TransactionSummary* but the structure of the call is very different.
This procedure should be used to get information about only the past periods.

```
Procedure TransactionSummary(Period_no, Trans_no: integer;
  var Error: integer;
  var TransactionSummaryRec: TransactionSummaryRecType);
    Error: = 1 Invalid Period Number;
    Error: = 2 Invalid Transaction Number;
```

All the bids and asks for the current transaction are also stored in the memory in an array of the following format.

```
  BidAskRecord = record
  Trans_Type: char;
```

```
    Price: longint;
    end;

  BidAskArrayType = array[1..MaxBidAsks] of BidAskRecord;
  BidAskArray: BidAskArrayType;
```

4. LastAccess

```
Procedure LastAccess(Period_no: longint; var Error: integer);
```

This procedure is similar to Access except that it positions the pointer to the last activity record of the specified period.

5. ReadTransaction

```
Procedure ReadTransaction(PeriodNo, TransNo: longint;
  var Error: integer;
  var Trans_rec: Trans_Rec_Type);
```

This procedure is similar to Access except that in addition to positioning the pointer, it also reads the record into Trans_rec.

APPENDIX B
STRUCTURE OF BUYER PROGRAMS

This section gives a short list of the trading decisions made by the programs, and the decision criteria employed by these programs.

Opening Strategy

1. Whether to open the bidding or not.
2. If yes then at what price?
   a. What are the variables on which opening price is based?
   b. How do these variables change with
      i. time and/or
      ii. number of transactions
      iii. across periods?

Possible bidding strategies include bidding at previous period's minimum bid/transaction price minus some multiple of standard deviation. In the first period, current period mean and standard deviation could be used. Some traders decided to wait for a specified number of transactions in the market before opening the bidding.

Bidding Strategy

There are three major decisions to be made in the case of bidding:

1. When to bid:
   i. Only when someone else has bid.
   ii. Only when a new ask has come in.
   iii. When either i or ii is true.
   iv. When a significant amount of time has passed since the last bid or ask.
   v. Very little time is left in the period.
2. Bidding increment: Most programs used a very small increment (it is safe!). Most popular number is an increment of 1 (the minimum possible in this programming environment). Other strategies included:
   i. Increment depends on the difference between the current bid and the trader's value for the current unit. It could either be
      A. A fraction of the difference. This causes the current bid to approach the trader's value asymptotically.
      B. A step function with the size of the steps changing at fixed points.
   ii. Increment is a random number in a small range.
   iii. Increment depends on the difference between the current ask and current bid.
3. When to stop bidding, that is, what is the upper limit on the bids?
   i. Dependent on (mostly a fixed multiple of) the last transaction price.
   ii. Dependent on the mean transaction price of the previous period plus some multiple of the standard deviation of transaction prices in the previous period.
   iii. The maximum transaction price of the previous period (or the maximum bid of the previous period in case the last event of the period was not a transaction.)
   iv. Mean of the last *n* transactions plus a multiple of their standard deviation.
   v. The closing price of the previous period.
   vi. Value of the unit to the buyer.

Take Strategy

1. When a fixed amount of profit can be made, that is, value minus current ask is greater than a fixed number.

2. Current ask less than a fixed percentage of the value.
3. Current ask less than the minimum price of the previous period.
4. Current ask less than mean price minus a multiple of its standard deviation.
5. Current ask less than the last transaction price minus a fixed number.

Urgent Strategy

The traders included a period end strategy to make any profit possible when very little time is left in the period. They were willing to bid against themselves and were also willing to raise the bid to their value.

REFERENCES

Friedman, D., & Sunder, S. (1994). *Experimental methods: A primer for economists.* New York: Cambridge University Press.

Gode, D. K., & Sunder, S. (1992). *What makes markets efficient?* Working paper, Graduate School of Industrial Administration, Carnegie Mellon University.

Gode, D. K., & Sunder, S. (1993a). Allocative efficiency of markets with zero intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy, 101*(1), 119–137.

Gode, D. K., & Sunder S. (1993b). Lower bounds for efficiency of surplus extraction in double auctions. In I. D. Friedman & J. Rust (Eds.), *The double auction market: Institutions, theories, and laboratory evidence* (pp. 199–219). Santa Fe Institute Studies in the Sciences of Complexity. New York: Addison-Wesley.

Plott, C. R., & Sunder, S. (1982). Efficiency of controlled security markets with insider information: An application of rational expectation models. *Journal of Political Economy, 90*(4), 663–698.

Plott, C. R., & Sunder, S. (1988). Rational expectations and the aggregation of diverse information in laboratory security markets. *Econometrica 56*(5), 1085–1118.