The Graz-Innsbruck Market System

v7.2.1

Stefan Palan

User Manual

Content

Preamble
Dear reader,

Thank you for your interet in asset market experiments in general, and in GIMS in particular. Having spent some years in this research field, I decided to create a software package which could serve as a standardized basis for any number of experimental designs. Retaining core elements, it would thus serve as a reliable basis for different research projects, obviating the need to describe the software in detail in every new paper. GIMS was created from scratch, but is based on the experience I have gathered in my work.

This manual is intended as the one-stop-shop on all questions GIMS. Please do not be intimidated by its bulk. While it is lengthy, this is not (only) because there is so much to document about GIMS, but because this manual offers a number of examples and summaries and tries to make it easy for you to get started preparing your own experiment. Nonetheless, the manual is aimed at readers familiar with z-Tree programming. If you are new to z-Tree, please refer to the z-Tree Manual (Fischbacher et al. 2015), read free course slides uploaded by lecturers (consult google to find some) or attend one of the z-Tree courses offered from time to time by universities around the world. Note that I also have some z-Tree material other than GIMS on my website http://academic.palan.biz (see the Downloads section).

Having said that, I'll not keep you from the manual any longer. If you have questions which are not answered in this manual, feel free to contact me under gims@palan.biz. Furthermore, I'm grateful if you let me know of working papers and published papers relying on GIMS and – unless you do not wish it – I will list them on the GIMS website.

Best of luck with your experiment,

Stefan Palan.

1. Program overview

The program offers extensively customizable facilities for conducting market experiments. It runs on z-Tree 3.4.7 (Fischbacher 2007) with a minimum resolution of 1024x768 pixels and has features listed in the subsequent subsections.

A guide to notation: Throughout the manual, \Var refers to the variable Var in the globals table. More generally, tablename.Var refers to the variable Var in the tablename table.

1.1. General

- Multi-language support. The entire treatment can be run in English or German.
- Universal timelog. The program logs every event in a single, unified timelog table, using the precise computer time instead of z-tree's imprecise experiment time.
- Experimenter subject. Experimenter z-leaf can be used to view market proceedings and to easily program experimenter interventions where desired.
- Experimenter interrupt. Experimenter can at all times move all subjects to the next stage or period or terminate the experiment. This is especially valuable for programming and testing.

1.2. Market

- Complete record. No data is ever overwritten or lost. E.g. in the case of the partial transaction of an open limit offer, both the original offer and all transactions resulting therefrom are saved and connected such that the mapping from offer to transaction is clear.
- Choice between single unit trading or multi-unit trading.
- Support for trading multiple assets.
- Free choice of number of periods. (Even random ending is trivial to implement given experimenter interrupt function.)
- Possibility to reset or carry over subject wealth from one period to the next.
- Possibility to pay subjects based on the cash balance at the end of every period or only for the cash balance at the end of the final period.
- Dividends can either be randomly drawn from a freely specifiable discrete distribution, or can be imported from file.
- Possibility to display signals which can either be randomly drawn or imported from file.
- Random assignment of freely specifiable endowment types to traders (optional).
- Optimized layout. The layout of the market interface is designed to facilitate trading.
- Some preparations for multiple market-trading already implemented, but multi-market trading currently not supported.
- Choice between continuous double auction and call auction, or use of both.
- Possibility to short assets and cash.
- Continuous double auction
  - Order validation. Offers can only be submitted if the submitting trader is able to fill them. Offers can only be accepted if the subject accepting them is able to pay for them (accepted ask) or deliver the assets (accepted bid). Subjects can only accept the best offer to buy/sell currently outstanding. Subjects cannot trade with themselves.

- Automatic invalidation of orders. Individually valid orders are allowed, but upon each transaction all existing outstanding orders are evaluated and (optional) their volume reduced (multi-unit setting) or the order invalidated if they are no longer feasible. In other words, a subject can have orders outstanding which he or she could not all fill, but every single outstanding order viewed in isolation is always feasible.
- Possibility to let subjects vote for an early end to trading. Only when all subjects have voted for ending the period early does it actually end.
- SpamControl feature to prevent individual subjects from filling the order book with bids or asks (optional).
- Bid-ask-improvement rule (optional).
- Display of a list of transaction prices (optional).
- Display of a chart of transaction prices (optional).
- Subject can display (i) all his/her non-open orders, (ii) only trades, (iii) only cancelled orders, or (iv) only invalidated orders.
- Order book can be emptied after every trade (optional).
- Price changes in any market are highlighted in the multi-market interface to attract subjects' attention particularly when they are viewing different markets.

- Call auction
  - Possibility to let subjects submit a singe bid and ask pair, or entire bid and ask schedules with or without limits on the number of individual bids and asks.
  - Possibility to submit limit and (optionally) market orders. Note that allowing market orders introduces the risk of subject bankruptcy, since the finally resulting market price is unknown at the time of checking order legality.
  - Price-Market order (price assumed to equal auction price)-Time priority rule or Price-Limit order-Time priority rule.
  - Possibility to display indicative price, updated at freely specifiable intervals.
  - Performance on Intel® Mobile Core™i7-2630QM (4 cores, 2,00 GHz, 6 MB Cache) with 16GB RAM (assuming each price between 1 and the number of limit bids/asks is used once, volume random between [1,100], continuous random submission times [0,1]). First number is time for price determination, second is for trade settlement:

72 limit bids/asks, 12 market bids/asks [sec]:	0.265	0.671	(standard scenario for 24 client-lab)
100 limit bids/asks, 10 market bids/asks [sec]:	0.453	1.139	
100 limit bids/asks, 100 market bids/asks [sec]:	0.687	4.337	
200 limit bids/asks, 0 market bids/asks [sec]:	1.607	3.463	
240 limit bids/asks, 24 market bids/asks [sec]:	2.496	6.411	(stress scenario for 24 client-lab)
500 limit bids/asks, 50 market bids/asks [sec]:	10.015	26.989	
1000 limit bids/asks, 100 market bids/asks [sec]:	43.915	103.257	

  - 5-step price determination procedure (in pseudo-code):
    - Step 0: If the highest buy offer limit price is lower than the lowest sell offer limit price, no trade is possible. This extends to market orders. Set AuctionPrice equal to -1 and stop price determination. Otherwise, continue.
    - Step 1: If there is a single price which maximizes the feasible trading volume, set AuctionPrice equal to this price and stop price determination. Otherwise, continue.
    - Step 2: If there is a single volume maximizing price which minimizes the absolute oversupply, set AuctionPrice equal to this price and stop price determination. Otherwise, continue.
    - Step 3: If there is a surplus of buy (sell) volume at all volume maximizing and absolute oversupply minimizing prices, set AuctionPrice equal to the highest (lowest) of these prices and stop price determination. Otherwise, continue.

- Step 4: Calculate the average of the highest and lowest volume maximizing and absolute oversupply minimizing prices and set AuctionPrice equal to this average, rounded to the nearest tick. In the case of equal distance, round down to the nearest tick.

- This mechanism replicates the price determination mechanism of NASDAQ OMX. Steps 0-1 are the same for all markets analyzed (ASX, BSE India, NASDAQ OMX, NSE India, NYSE, XETRA), and Step 2 is used by all markets except for the NYSE.
  At some point, most exchanges (except for NASDAQ OMX) use a reference price (e.g. the previous closing price) in case earlier steps do not lead to a unique price. However, this is impractical in an experimental market, where there is no such price in the first period, and where – depending on the experiment - periods may be designed to be independent.
- 3-step trade settlement procedure:
  - Step 0: If no trade is possible, stop settlement. Otherwise, continue.
  - Step 1: Sort orders by priority.
    If set to give priority to market orders, then priority is: Better than equilibrium limit orders, market orders, at equilibrium limit orders.
    If set to give priority to limit orders, then priority is: Better than equilibrium limit orders, at equilibrium limit orders, market orders.
    Within better than equilibrium limit orders, priority is by price; within all categories final priority is by submission time.
  - Step 2: Transact orders against each other in order of their priority, starting with the surplus market side, or the bid side in case of no surplus.

1.3. Elicitation of subject characteristics

- Elicitation of CRT (Frederick 2005) before or after the market phases (optional).
- Elicitation of the Holt, Laury 2002 risk aversion measure before or after the market phases (optional).
- Elicitation of the 16 financial literacy questions of van Rooij et al. 2011, or of a subset thereof after the market phases (optional). These questions can be unpaid, paid for every correct question, or paid for one randomly chosen question.

2. Program structure

This section is intended to give the reader a good idea of the logic behind GIMS. Some of the program structure may deviate from other asset markets' designs, thus necessitating some explanation.

2.1. General

As many settings as possible are set in the Background. Note that throughout this section, when we refer to "P1", we mean the globals table program titled "//### P1: General parameters ###" in GIMS' Background; by "P2" we mean "//### P2: Stage settings ###", etc. Have a look through these programs to get an idea of the settings available for modification directly from GIMS' Background. For example, the timout of individual stages is not set in the stage definition, but is centralized in P2. The same applies to many other settings.

In addition to the programs settings, the Background contains programs which initialize the variables used in GIMS. In z-Tree, it is vital that all variables are assigned an initial value at some point in the background. Ignoring this rule risks z-Tree displaying error messages which sometimes have no recognizable connection to the source of the problem (e.g., there may be a warning that variable x is unknown, when in fact variable y was not initialized).

2.2. Endowments, dividends and signals

In GIMS, subjects' initial endowments of assets and cash are saved in the endowments table. This allows you great flexibility in assigning endowments to subjects. Write only a single endowment type into this table (consisting of a type ID Type, a number of assets Assets and an amount of cash Cash) and every subject gets the same endowment. Write more than one endowment type into the table, and each endowment type is assigned to the same number of subjects (see section 4.2 for an example, including a description of what happens in the case where the subject number is not divisible by the number of endowment types). Write as many endowment types as there are subjects into the table, and each subject gets assigned one of the endowment types. Furthermore, GIMS offers the possibility to import an endowment table from an ASCII file, thus letting the experimenter quickly switch different endowment distributions. The same mechanism is used for signals to be provided to subjects (e.g., if each subject receives one or multiple signals for the asset's fundamental value) and for possible dividend amounts to be drawn and paid after every period.[1]

2.3. Layout

The layout of background colors and font sizes is centralized as much as possible and practical. While some settings are still made in distributed places, many layout settings are pooled in the layout table. Currently the background colors used for the various boxes on the trading screens, as well as e.g. the color of the line in the transaction price chart in the continuous double auction are already set in a single place, by modifying P6, which fills the layout table. In the future, also settings like font sizes and the like are intended to be set using the layout table.

2.4. Timelog

Every subject and many program actions are logged in a unified timelog table in GIMS. This table contains one entry per "event", with the event ID indicating the type of event being logged. Examples of events are the program start, subjects entering a stage, a subject submitting a buy offer, or a subject clicking a button to see a different filtered set of past orders. The timing of the events is saved in terms of seconds since the treatment's start, with millisecond precision. If a specific subject can be tied to the event (e.g., the subject submitting an order), his or her subject number is saved. Furthermore, the array variable Data can be used to save additional information about the event.

[1] See Table 1 in section 4.3 for an example showing the structure of such an ASCII file.

2.5. Experimenter subject and progression through stages and periods

In addition to the regular subjects participating in the experiment, GIMS has the option to allow for experimenter subjects, i.e. z-Leaf clients which are run on the server computer and controlled by the experimenter. This offers many opportunities for displaying summary data to the experimenter during the experiment, for allowing the experimenter inputting data into z-Tree, and for allowing the experimenter to move subjects through a treatment. By default, the number of experimenter subjects is set to 1 and the last z-Leaf to connect to the server is the experimenter's.

2.5.1. Displaying summary data to the experimenter

The experimenter z-Leaf can be used to display information about the current situation in the experiment. Making use of this possibility, GIMS by default displays a modified of the call auction and continuous double auction screens to the experimenter (see Figure 1). The researcher can thus follow the events of the market in real time, seeing all bis, asks, transactions and signals both in the continuous double auction and in the call auction.

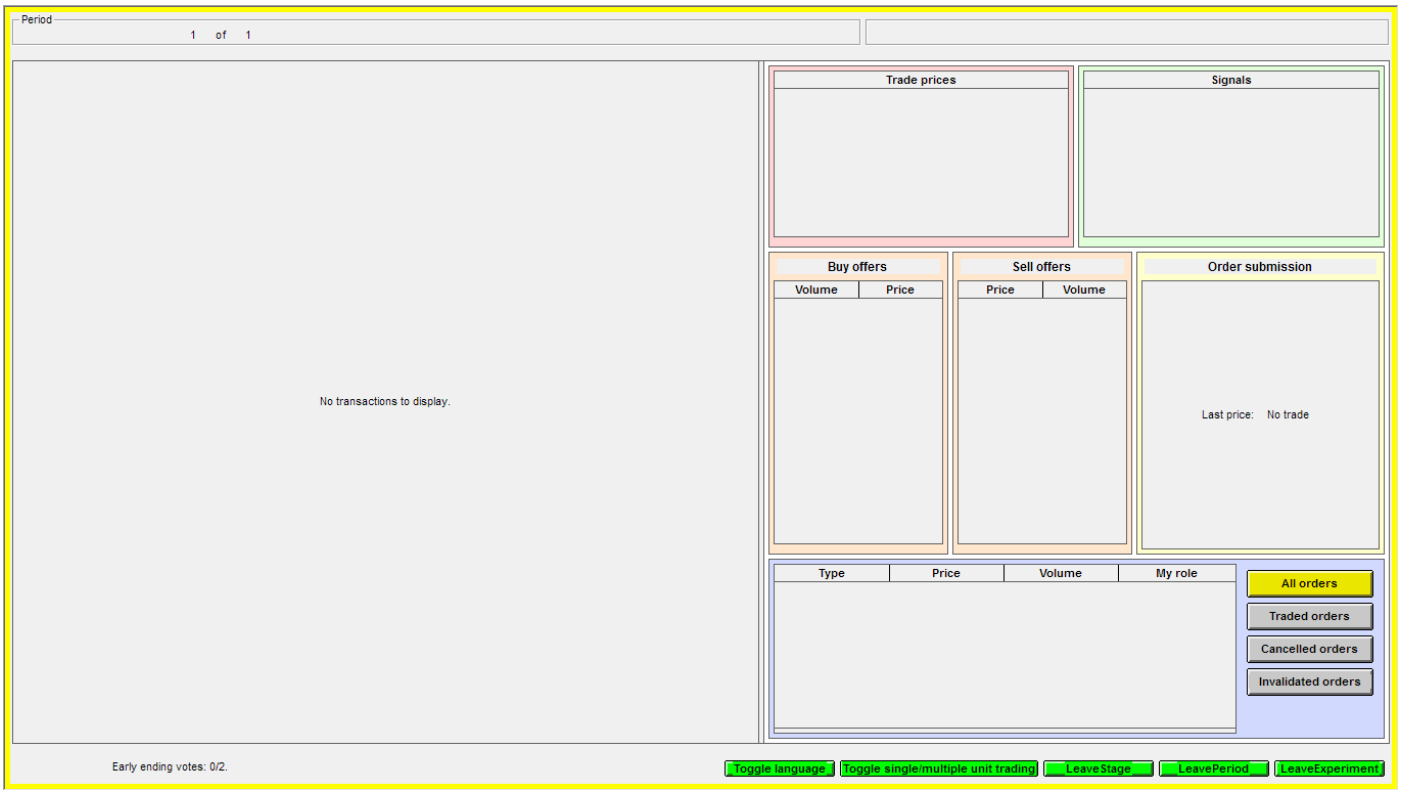

Figure 1 Experimenter screen in continuous double auction stage

2.5.2. Data input by the experimenter

While not built into GIMS by default, the experimenter z-Leaf can easily be configured to allow the experimenter to enter data into z-Tree. A common application of this feature would be to allow the experimenter to draw a random number using an offline randomization device (e.g., a die, or a bingo cage) and then enter this number into z-Tree such that z-Tree can use this value in the remainder of the experiment (e.g., to pay out only a randomly drawn dividend, or to assign a specific role to a randomly drawn subject).

2.5.3. Progressing through stages and periods

The experimenter can at any time move all subjects to the next stage or period, or terminate the experiment. She/he simply has to click the corresponding button in the experimenter z-Leaf (see the buttons at the bottom of Figure 1). Programmatically, this is achieved by setting the ForcedExitPeriod and ForcedExitExperiment variables in the globals table equal to 1. Since no stages can be *skipped* per se in z-Tree, every stage checks these variables to determine whether subjects *participate* in the stage or not. Only if neither of these two variables is greater than 0 are subjects allowed to participate in a stage. The value of ForcedExitExperiment is furthermore carried over from one period to

the next, such that, when it is set to 1, subjects pass through all the stages and all the periods, but do not see that they do so since their Participate variable in the subjects table remains set to 0.

Note that the presence of an experimenter subject causes a difficulty when stages end automatically after subjects have made an entry. In this case, z-Tree would normally progress to the next stage when all subjects have made their entry – including the experimenter subject, which in most cases is not required to make an entry similar to the other subjects.

2.6. Advice for working with GIMS

2.6.1. User-defined programs

It is advisable not to modify the pre-defined GIMS programs (P1, P2…). If you need to change variables, create your own programs and start them with a line which reads…

"//#### CUSTOM: Changing the payoff structure ###"

…or something similar. This way, you can easily identify your own settings whenever you work in the program. This also makes it easier to identify problems as being caused by the original GIMS programming (in this case, please send a mail to gims@palan.biz describing the problem), or as being caused by your own programming, which helps you find the error. Finally, it enables you to easily upgrade your experimental treatment to future versions of GIMS which may incorporate bug fixes and additional features.

2.6.2. Program order

As in any z-Tree program, the order of the programs in the Background is very important and should not be modified. Later programs rely on the settings made in earlier programs. For this reason, you also need to think carefully where to insert your user-defined programs.

2.6.3. Testing

To test individual elements, use Testmode variables to turn on or off specific elements of a screen, or even subjects' participation in entire stages. For example, create a variable Testmode2 in the globals table which you set to 1 (0) if you want to turn the test mode on (off). Then, set the Participate condition of all stages you are currently not testing to be 0 if Testmode2 equals 1. For example, if the Participate condition for a stage reads…

Participate=1-IsExperimenter

…modify it to read:

Participate=(1-IsExperimenter)*(1-Testmode2)

This way, you can skip the stage you are not interested in and test only the stage you are currently working on. In fact, Testmode1 is already predefined. If set to 1, buttons to move all subjects to the next stage or period, or end the experiment entirely, appear on all clients' screens (not only on the experimenter's).

3. List of tables and variables

The following tables list all variables used in GIMS. While experienced programmers may naturally modify any element of the programming, variables controlling general program settings (all located in the globals or subjects tables) are printed with a blue background. These are the variables which will most frequently have to be modified to adapt GIMS to any specific experiment. Section 4 gives examples of common treatment variations and their realization in GIMS (using mainly the general program settings variables).

globals

(Life: Period)

Variable	Content
AllowMarketOrders	Switch to turn on (1) or off (0) market order submission in the CallAuction stage.
AskFinLit[16]	Switches for the individual questions.
AssetPrecison	Precision of asset number display. *Note that, due to z-Tree limitations, these are not implemented everywhere in the program. If you modify these settings, please test extensively to ensure these settings are reflected everywhere you want to see them.*
AuctionPrice [\NumMarkets]	Call auction price per asset type in this period.
AuctionVolume [\NumMarkets]	Call auction order volume executable using only limit orders per asset type. Used in price determination.
AuctionVolumeTotal [\NumMarkets]	Call auction order volume executable using only limit orders per asset type. Used in price determination.
BBV[\NumMarkets]	Buyback value of the assets. At the end of the final period (or of every period, if \IndependentPeriods==1), this value times the number of assets held is added to the subject's cash balance for the sake of the profit calculation (it is not really added to the subjects.Cash variable).
BuyMarketVol	Variable used in indicative price determination.
CallAuction	Switch to turn on (1) or off (0) the CallAuction and CallAuctionResults stages.
CallAuctionDone	Counter variable of number of subjects who are finished in CallAuction stage. Used to proceed after final subject has finished, even if Experimenter has not.
CallAuctionSignals [\NumMarkets]	Use signals in the CallAuction for an asset type. *Note that you may have to modify the box dimensions if you want to ensure an optimal screen layout when you turn this setting off.*
CarryOverWealth	Switch to determine whether wealth should be carried over from one period to the next (1) or not (0). See also \IndependentPeriods.
CashPrecison	Precision of cash display. *Note that, due to z-Tree limitations, these are not implemented everywhere in the program. If you modify these settings, please test extensively to ensure these settings are reflected everywhere you want to see them.*
CDATimer	Interval for the ContinuousDoubleAuction Timer task.
ChartXLabelDistance [\NumMarkets]	Distance of the x-axis labels for each asset's chart.
ChartXMax [\NumMarkets]	Maximum x-axis value for each asset's chart.
ChartXTickDistance [\NumMarkets]	Distance between ticks on the x-axis for each asset's chart.
ChartYLabelDistance [\NumMarkets]	Distance of the y-axis labels for each asset's chart.
ChartYMax [\NumMarkets]	Maximum y-axis value for each asset's chart.
ChartYTickDistance [\NumMarkets]	Distance between ticks on the y-axis for each asset's chart.

Variable	Content
ConnectedMarket-Mechanisms	Transactions from call auction are displayed on continuous double auction screens.
ContinuousDouble-Auction	Switch to turn on (1) or off (0) the ContinuousDoubleAuction and ContinuousDouble AuctionResults stages.
ContinuousDouble-AuctionSignals	Set to 1 (0) if the signal box should be displayed in the ContinuousDoubleAuction stage. *Note that you may have to modify the box dimensions if you want to ensure an optimal screen layout when you turn this setting off.*
Currency	1=Euro 2=US Dollars 3=Danish Kroner 4=Australian Dollars
CurrentYear	Current year for financial literacy question 5.
Dividend[\NumMarkets]	Dividend amount for each asset in the current period.
DividendSet	ID number of the set of dividends to use in case of preset dividends.
Done	Used to detect when all subjects are done with a stage.
EarlyEndingVote	If set to 1 (0), subjects can (cannot) vote to end a period in the continuous double auction stage prior to its natural timeout. Subjects who have voted to end the period can continue trading and can withdraw their vote at any time. Only when all subjects have voted to end the period does it really end.
EmptyBookAfterTrade [\NumMarkets]	Deletes all orders from the order book upon a successful trade.
EndowmentFillerType	Value of endowments.Type which is used to fill up endowments for subjects if the subject number is not divisible by the number of endowment types.
EndowmentTypes	Number of different endowment types.
ExchangeRate	Exchangerate, expressed as value of one ECU in CU. In the Profit variable, profit is always saved in actual CU, not ECU. This way, only this variable has to be modified, while the exchange rate in the Background is always 1.
FinLit	Switch to determine whether any financial literacy questions are being asked (1) or not (0).
FinLitRandomPay-Question	Number of the financial literacy question randomly chosen to be paid if PayFinLit Random==1. This number corresponds to the 16 internal question numbers, irrespective of the number actually asked.
FontSizeHeading	Font size used in headings.
FontSizeSubHeading	Font size used in subheadings.
ForcedExitExperiment	1 if experimenter forced exit from the experiment, 0 otherwise.
ForcedExitPeriod	1 if experimenter forced exit from this period, 0 otherwise.
HLAfter	Switch to determine whether the Holt and Laury risk elicitation task is run after the rest of the experiment (1) or not (0).
HLBefore	Switch to determine whether the Holt and Laury risk elicitation task is run before the remainder of the experiment (1) or not (0).
I	Initialization variable, =-77777
i, j, k, l, r	Temporary variables
IndependentPeriods	If set to 0 (1), the subject's cash balance is converted using the exchange rate and written into subjects.Profit only in the last period (in every period). See also \CarryOver-Wealth.
IndicativePriceInterval [\NumMarkets]	Interval (in sec) in which the indicative price of each asset type is being calculated in the CallAuction stage. Set to a negative value to turn off.
Language	Sets the experiment language to English (1) or German (2).
LastPrice[\NumMarkets]	Last transaction price for each asset type.
MarketOrderPriority [\NumMarkets]	If set to 0, priority is given to limit orders in call auction trade processing.
MaxDividend [\NumMarkets]	Highest possible dividend in a period per asset type.

Variable	Content
MeanDividend [\NumMarkets]	Mean dividend in a period per asset type.
MinDividend [\NumMarkets]	Lowest possible dividend in a period per asset type.
NumDividends [\NumMarkets]	Number of possible discrete dividend amounts per asset type.
NumExperimenter-Subjects	Number of experimenter subjects
NumFinLit	Number of financial literacy questions to be asked, if any (0…16).
NumMarkets	Number of simultaneous markets.
NumPeriods	Total number of periods in z-Tree program.
NumSubjects	Number of non-experimenter subjects.
NumTypedOffersCall-Auction[\NumMarkets]	Maximum number of simultaneous bids or asks outstanding per asset in the CallAuction stage by one person. 0 means not active, any positive number means a subject may not have more than this number of bids outstanding, or more than this number of asks outstanding.
NumVolumeMaximizing-Prices[\NumMarkets]	Variable used in indicative price determination. Contains number of offer prices at which the possible trading volume is maximized.
OrderBookDepth [\NumMarkets]	If set to <0, order book depth is unlimited, if set to >=0, only this number of offers is displayed on each side of the order book, excluding a trader's own offers (these are always displayed).
PartialInvalidation [\NumMarkets]	Orders are partially invalidated, i.e. the volume is reduced until the order becomes legal, if possible.
PayFinLit	Switch to determine whether answers on financial literacy questions are incentivized (1) or not (0).
PayFinLitRandom	Switch to determine whether answers on financial literacy questions are incentivized by randomly picking one (1) or paying for all (0).
PayoffAHigh	High Holt and Laury payoff for option A.
PayoffALow	Low Holt and Laury payoff for option A.
PayoffBHigh	High Holt and Laury payoff for option B.
PayoffBLow	Low Holt and Laury payoff for option B.
PayoffFinLit	Total payoff in case randomly chosen question (PayFinLitRandom==1) or all questions (PayFinLitRandom==0) are correct. If PayFinLitRandom==0, fewer than all answers correct reduces payoff proportionately. Expressed in Euro or equivalent.
Period	Period number of the record
PracticePeriod	Switch between practice period (1) and real period (0).
PresetDividends	Switch for whether dividends are randomly generated (0) or read from dividends.zdata (1).
PresetSignals	Switch for whether signals are randomly generated (0) or read from signals.zdata (1).
PriceDeterminationRule [\NumMarkets]	Rule used to determine the price in the Call Auction. See section 5.1. The mapping from \PriceDeterminationRule to the Steps listed in section 5.1 is 0=0, 1=1, 2=2, 3=3 (buy surplus), 4=3 (sell surplus), 5=4.
PriceHighlightColor [\NumMarkets]	Number of the price display font color.
PriceHighlightDuration	Duration for which prices should be highlighted in case \PriceHighlightScheme>0 in the ContinuousDoubleAuction stage. Must be >=0.5.
PriceHighlightScheme	Determines how new transaction prices are highlighted in the ContinuousDoubleAuction stage. If 0, there is no highlighting. If 1, price increases are highlighted in green and price decreases in red. If 2, price increases are highlighted in orange and price decreases are highlighted in blue.

Variable	Content
PriceHighlightTime [\NumMarkets]	Time (obtained from gettime()) until which current highlighting should remain upright.
PricePrecison	Precision of price input and output. *Note that, due to z-Tree limitations, these are not implemented everywhere in the program. If you modify these settings, please test extensively to ensure these settings are reflected everywhere you want to see them.*
RandomPayoffPeriod	If RandomPeriodPayoff is 1, this variable contains the period randomly chosen for payoff.
RandomPeriodPayoff	If set to 1 and if IndependentPeriods is 1, then one period is randomly chosen for payout. If set to 0, all periods are being paid out.
ResultsDone	Counter variable of number of subjects who are finished in Results stage. Used to proceed after final subject has finished, even if Experimenter has not.
SellMarketVol	Variable used in indicative price determination.
SellVolMarket [\NumMarkets]	Temporary variable for call auction price determination. Contains total volume of market sell orders.
ShowCAPracticeResults	Show practice period results screen in CallAuctionResults if \PracticePeriod==1.
SignalSuperSet	Chosen signals_preset.SuperSet.
SpamControl [\NumMarkets]	Maximum number of simultaneous bids or asks outstanding in the ContinuousDouble Auction stage by one person. 0 means not active, any positive number means a subject may not have more than this number of bids outstanding, or more than this number of asks outstanding.
StartTime	Computer time at start of period.
StartTimeCDA	Computer time at start of CDA stage.
Temp[4]	Array of temporary variables.
TempPrice	Variable used in indicative price determination.
Testmode1	Turns on Experimenter buttons on all screens.
TimeContinuousDouble-Auction	Length in seconds of the ContinuousDoubleAuction stage.
TimeContinuousDouble-AuctionResults	Length in seconds of the ContinuousDoubleAuctionResults stage.
TimeFinLitResult	Length in seconds of the FinLitResult stage.
TimeHoltLaury	Length in seconds of the HoltLauryBefore and –After stages.
TimeHoltLauryResult	Length in seconds of the HoltLauryResultBefore and –After stages.
TimerInterval	Interval length for timer tasks in seconds.
TradeProcessingSide	Market side, bid (1) or ask (-1) that is currently being processed in call auction order settlement.
VolumePrecision	Precision of volume input and output. *Note that, due to z-Tree limitations, these are not implemented everywhere in the program. If you modify these settings, please test extensively to ensure these settings are reflected everywhere you want to see them.*

subjects

(Life: Period)

Variable	Content
AnswerFinLit[16]	Binary variable reporting whether answer on this financial literacy question was correct (1) or not (0).
Assets[\Num-Markets]	Subjects' balance of each asset type.
AssetValueTemp	Temporary variable for the calculation of profits in the CallAuction and ContinuousDoubleAuction stages.
Cash	Subject's cash balance.

Variable	Content
DisplayMarket [\NumMarkets]	Array determining which market is displayed as the primary market (DisplayMarket[1]), or the secondary market (DisplayMarket[2]), etc., for a subject.
DisplayStatus	Status to display in the market's history box.
EarlyEndingVote	Set to 1 when subject has voted to end a period in the continuous double auction stage prior to its natural timeout. Subjects who have voted to end the period can continue trading and can withdraw their vote at any time. Only when all subjects have voted to end the period does it really end. See \EarlyEndingVote.
Endowment-Assigned	1 if endowment has already been assigned, 0 if not.
FinLitID	Financial literacy question ID displayed to subjects.
FinLitRandomPay Question	Number of the financial literacy question randomly chosen to be paid if PayFinLitRandom==1. This number corresponds to the 16 internal question numbers, irrespective of the number actually asked.
FinLitRandomPay QuestionID	Number of the financial literacy question randomly chosen to be paid if PayFinLitRandom==1. This number corresponds to the question number displayed to the subject.
FinLitScore	Number of correctly answered financial literacy questions.
Group	Group the subject belongs to. Corresponds to GroupID in the markets and groups tables.
HL	Subject's Holt and Laury risk elicitation task score, equal to 10-HLChoice. A value of 0 corresponds to 0 risky choices, a value of 10 to the maximum of 10 risky choices.
HLChoice	Set from \I to the subject's switching row [0,10] when subject has at least once clicked on an option in the Holt and Laury risk elicitation task.
HLDieRoll	Random number [0,10] determining whether high or low payoff is paid in Holt and Laury task.
HLPayoff	Payoff in CU for Holt and Laury task.
HLRandomRow	Randomly chosen row to be paid.
InitialAssets [\NumMarkets]	Initial endowment of each asset type.
InitialCash	Initial cash endowment.
IsExperimenter	0 if normal participant, 1 if experimenter subject
Period	Period number of the record
Profit	Period profit, calculated as (Money-StartMoney)*\exchangerate.
r, i, j	Temporary variables
ShortingCapacity-Assets[\Num-Markets]	Amount of assets of each type subject may be short.
ShortingCapacity-Cash	Amount of cash subject may be short.
Subject	Subject number of the record
TotalProfit	Cumulative profit up to this period

offers

(Life: Period)

Variable	Content
ID	Offer ID.
Invalidated	Number of units invalidated due to order legality rules (e.g. cash constraint) (0…Volume) or due to \EmptyBookUponTrade==1.
Market	Market ID.
Offerer	Value of MarketOrders in subject's record in the subjects table.
OfferTime	Time of offer creation.
Price	Offered price.

Variable	Content
Status	Offer status. Open (0), closed after being fully transacted (1), cancelled (2), closed after being fully invalidated (3), expired (4).
StatusTime	Time of last status update.
Transacted	Number of units already transacted (0…Volume).
Type	Buy (1) or sell (-1) offer.
Volume	Offered volume.

auctionoffers

(Life: Period)

Variable	Content
ID	Offer ID.
Market	Market ID.
MarketOrder	0 if limit order, 1 if market order.
Offerer	Value of subjects.Subject of the subject who created this order.
OfferTime	Time of offer creation.
Price	Offered price. Equals \I if auctionoffers.MarketOrder==1.
Priority	Order settlement priority. Equals 999999 if order is non-tradeable at this period's \AuctionPrice.
Status	Offer status. Open, or not traded due to being too late in price-time priority scheme (0), closed after being fully or partially transacted (1), cancelled (2), closed after having been found not to be transactable at \AuctionPrice (3).
StatusTime	Time of last status update.
Transacted	Number of units transacted (0…Volume).
Type	Buy (1) offer, sell (-1) offer.
Volume	Offered volume.

auctionpractice

(Life: Period)

This table contains data for display purposes in the CallAuctionResults stage if \PracticePeriod==1 and \ShowCAPracticeResults==1.

Variable	Content
Market	Market ID.
DisplayMarket	Market ID for display purposes (DisplayMarket=1 means Market=2 and vice versa).
Subject	Value of subjects.Subject of the subject who created this order.
Type	Buy (1) offer, sell (-1) offer.
PriceLow	Lower bound of the price range the call auction's outcome should be displayed for.
PriceHigh	Upper bound of the price range the call auction's outcome should be displayed for.
Volume	Cumulative volume the subject would buy or sell (see auctionpractice.Type) in this price range.

results

(Life: Period)

This table contains data for display purposes in case of multiple markets.

Variable	Content
Amount	Number of assets this subject of this market the subject owns at the end of the period.
DisplayMarket	Market ID for display purposes (DisplayMarket=1 means Market=2 and vice versa).
Dividend	Dividend paid on this asset in this period.
Market	Market ID.
NumBought	Number of assets this subject bought in this market in this period.

Variable	Content
NumSold	Number of assets this subject sold in this market in this period.
Period	Period number of the record
Phase	1=CallAuction, 2=ContinuousDoubleAuction.
Price	\AuctionPrice[Market] in case of CallAuction, average price this subject traded at in the ContinuousDoubleAuction.
Subject	Subject number of the subject the data stems from.

settlement

(Life: Period)
This table contains

Variable	Content
AskID	Value of auctionoffers.ID of the ask to be transacted.
BidID	Value of auctionoffers.ID of the bid to be transacted.
ID	Record ID.
Market	Market ID.
Period	Period number of the record
Volume	Original offered volume.

transactions

(Life: Period)

Variable	Content
AcceptanceID	ID of this transaction.
AccepterID	Subject number of accepting subject in the Continuous Double Auction.
Market	Market ID.
OfferID	Corresponds to ID in offers table.
Price	Offered price.
Time	Transaction time.
Volume	Transacted volume.

pricedetermination

(Life: Period)
Table used to determine equilibrium price in call auction market.

Variable	Content
BySurplus	Surplus of CumBuyVol over CumSellVol.
BuyVol	Total buy order volume at exactly this price, excluding market orders.
CumBuyVol	Total cumulative buy order volume at this or a lower price, including market orders.
CumSellVol	Total cumulative sell order volume at this or a higher price, including market orders.
FeasibleVol	Maximum possible transaction volume at this price, including market orders.
Market	Market ID.
Price	Unique price derived from auctionoffers.price.
SellVol	Total sell order volume at exactly this price, excluding market orders.

indicativepricedetermination

(Life: Period)
Table used to determine indicative equilibrium price in call auction market.

Variable	Content
BySurplus	Surplus of CumBuyVol over CumSellVol.

Variable	Content
BuyVol	Total buy order volume at exactly this price, excluding market orders.
CumBuyVol	Total cumulative buy order volume at this or a lower price, including market orders.
CumSellVol	Total cumulative sell order volume at this or a higher price, including market orders.
FeasibleVol	Maximum possible transaction volume at this price, including market orders.
Market	Market ID.
Old	Set to 1 if record has expired, or to 0 if the record is currently being processed.
Price	Unique price derived from auctionoffers.price.
SellVol	Total sell order volume at exactly this price, excluding market orders.

dividends

(Life: Period)
Filled by either creating new records or copying records from dividends_preset.

Variable	Content
Dividend	Dividend value.
Market	Market ID.
Period	Period number of the record
Type	ID number of this dividend value.

dividends_preset

(Life: Treatment)
Read in from dividends_preset.zdata in the first period.

Variable	Content
Dividend	Dividend value.
Market	Market ID.
Period	Period number of the record
Set	ID number of this dividend set.

signals

(Life: Period)
Filled by either creating new records or copying records from signals_preset.

Variable	Content
Market	Market ID.
Period	Period number of the record
Set	ID number of this signal's set. (e.g., all signals seen by one subject)
Signal	Dividend value.
Type	ID number of this dividend value.

signals_preset

(Life: Treatment)
Read in from signals_preset.zdata in the first period.

Variable	Content
Market	Market ID.
Period	Period number of the record
Set	ID number of this signal's set. (e.g., all signals seen by one subject)
Signal	Signal value.
SuperSet	ID number of this signal's superset. (e.g., all signals used in one session)
Type	Signal type. (e.g., public, private, …)

layout

(Life: Treatment)

This table contains the central list of layout definitions used for screen elements in the program. By changing the layout variables here, the layouts of the elements change.

Variable	Content
B	Fraction of blue color in the interval [0…1].
G	Fraction of green color in the interval [0…1].
ID	String variable containing the name of the screen element, usually constructed using the stage and element name, e.g. “CDA_Chart_Line”.
IDNumeric	Numeric ID of the record.
R	Fraction of red color in the interval [0…1].

endowments

(Life: Period)

Variable	Content
Assets[\NumMarkets]	Number of assets of this asset in this endowment type.
Cash	Amount of cash in this endowment type.
Type	Endowment ID.

timelog

(Life: Period)

Variable	Content
Data[…]	Array variable for saving additional data
Event	Event being logged: 1. Program start. 2. Start ContinuousDoubleAuction stage. 3. Subject sells at market price in the ContinuousDoubleAuction stage. offers.ID saved in Data[1]. 4. Subject buys at market price in the ContinuousDoubleAuction stage. offers.ID saved in Data[1]. 5. Subject cancels sales offer in the ContinuousDoubleAuction stage. offers.ID saved in Data[1]. 6. Subject cancels purchase offer in the ContinuousDoubleAuction stage. offers.ID saved in Data[1]. 7. Subject creates sell offer in the ContinuousDoubleAuction stage. offers.ID saved in Data[1]. 8. Subject creates buy offer in the ContinuousDoubleAuction stage. offers.ID saved in Data[1]. 9. Subject switches to DisplayStatus=3 (invalidated) in market 1. 10. Subject switches to DisplayStatus=2 (cancelled) in market 1. 11. Subject switches to DisplayStatus=1 (traded) in market 1. 12. Subject switches to DisplayStatus=-1 (all) in market 1. 13. Experimenter forces LeavePeriod in ContinuousDoubleAuctionResults stage. 14. Experimenter forces LeavePeriod in ContinuousDoubleAuction stage. 15. Experimenter forces LeaveExperiment in ContinuousDoubleAuction stage. 16. Experimenter forces LeaveExperiment in ContinuousDoubleAuctionResults stage. 17. Experimenter forces LeavePeriod in CRTBefore stage. 18. Experimenter forces LeaveStage in CRTBefore stage. 19. Experimenter forces LeaveExperiment in CRTBefore stage. 20. Experimenter forces LeaveStage in ContinuousDoubleAuction stage. 21. Subject clicks OK in CRTBefore stage. 22. Subject clicks OK in CRTAfter stage. 23. Experimenter forces LeaveStage in CRTBefore stage. 24. Experimenter forces LeavePeriod in CRTBefore stage. 25. Experimenter forces LeaveExperiment in CRTBefore stage. 26. Experimenter forces LeaveStage in ContinuousDoubleAuctionResults stage.

Variable	Content
	27. Subject clicks OK in FinLitIntro stage. 28. Subject clicks OK in FinLit01 stage. 29. Subject clicks OK in FinLit02 stage. 30. Subject clicks OK in FinLit03 stage. 31. Subject clicks OK in FinLit04 stage. 32. Subject clicks OK in FinLit05 stage. 33. Subject clicks OK in FinLit06 stage. 34. Subject clicks OK in FinLit07 stage. 35. Subject clicks OK in FinLit08 stage. 36. Subject clicks OK in FinLit09 stage. 37. Subject clicks OK in FinLit10 stage. 38. Subject clicks OK in FinLit11 stage. 39. Subject clicks OK in FinLit12 stage. 40. Subject clicks OK in FinLit13 stage. 41. Subject clicks OK in FinLit14 stage. 42. Subject clicks OK in FinLit15 stage. 43. Subject clicks OK in FinLit16 stage. 44. Experimenter forces LeaveStage in a FinLit stage. 45. Experimenter forces LeavePeriod in a FinLit stage. 46. Experimenter forces LeaveExperiment in a FinLit stage. 47. Subject clicks “OK” in FinLitResult stage. 48. Start CRTBefore stage. 49. Start CRTAfter stage. 50. Start ContinuousDoubleAuctionResults stage. 51. Start FinLitIntro stage. 52. Start FinLit01 stage. 53. Start FinLit02 stage. 54. Start FinLit03 stage. 55. Start FinLit04 stage. 56. Start FinLit05 stage. 57. Start FinLit06 stage. 58. Start FinLit07 stage. 59. Start FinLit08 stage. 60. Start FinLit09 stage. 61. Start FinLit10 stage. 62. Start FinLit11 stage. 63. Start FinLit12 stage. 64. Start FinLit13 stage. 65. Start FinLit14 stage. 66. Start FinLit15 stage. 67. Start FinLit16 stage. 68. Start FinLitResult stage. 69. Experimenter clicks OK in ContinuousDoubleAuctionResults stage in final period. 70. Subject clicks Continue in ContinuousDoubleAuctionResults stage. 71. Subject clicks “Confirm” in HoltLauryBefore stage. Data[1] contains subjects.HLChoice. 72. Subject selects a row in the HoltLauryBefore stage. Data[1] contains subjects.HLChoice. 73. Subject clicks OK in HoltLauryResultBefore stage. 74. Experimenter forces LeaveStage in HoltLauryBefore stage. 75. Experimenter forces LeavePeriod in HoltLauryBefore stage. 76. Experimenter forces LeaveExperiment in HoltLauryBefore stage. 77. Experimenter forces LeaveStage in HoltLauryResultBefore stage. 78. Experimenter forces LeavePeriod in HoltLauryResultBefore stage.

Variable	Content
	79. Experimenter forces LeaveExperiment in HoltLauryResultBefore stage. 80. Subject clicks “Confirm” in HoltLauryAfter stage. Data[1] contains subjects.HLChoice. 81. Subject selects a row in the HoltLauryAfter stage. Data[1] contains subjects.HLChoice. 82. Subject clicks OK in HoltLauryResultBefore stage. 83. Experimenter forces LeaveStage in HoltLauryBefore stage. 84. Experimenter forces LeavePeriod in HoltLauryBefore stage. 85. Experimenter forces LeaveExperiment in HoltLauryBefore stage. 86. Experimenter forces LeaveStage in HoltLauryResultBefore stage. 87. Experimenter forces LeavePeriod in HoltLauryResultBefore stage. 88. Experimenter forces LeaveExperiment in HoltLauryResultBefore stage. 89. Start HoltLauryBefore stage. 90. Start HoltLauryResultBefore stage. 91. Start HoltLauryAfter stage. 92. Start HoltLauryResultAfter stage. 93. Experimenter forces LeaveStage in CallAuctionResults stage. 94. Experimenter forces LeaveStage in CallAuction stage. 95. Experimenter forces LeavePeriod in CallAuctionResults stage. 96. Experimenter forces LeavePeriod in CallAuction stage. 97. Experimenter forces LeaveExperiment in CallAuction stage. 98. Experimenter forces LeaveExperiment in CallAuctionResults stage. 99. Start CallAuction stage. 100. Start CallAuctionResults stage. 101. Subject creates market sell order in the CallAuction stage. offers.ID saved in Data[1]. 102. Subject creates market buy order in the CallAuction stage. offers.ID saved in Data[1]. 103. Subject cancels sales offer in the CallAuction stage. offers.ID saved in Data[1]. 104. Subject cancels purchase offer in the CallAuction stage. offers.ID saved in Data[1]. 105. Subject creates sell offer in the CallAuction stage. offers.ID saved in Data[1]. 106. Subject creates buy offer in the CallAuction stage. offers.ID saved in Data[1]. 107. Experimenter clicks OK in ContinuousDoubleAuctionResults stage in final period. 108. Subject clicks Continue in CallAuctionResults stage. 109. Completed price determination CallAuction stage. 110. Completed settlement CallAuction stage. 111. Subject submits offers in CallAuction stage. 112. - 113. - 114. - 115. - 116. Subject voted to end the period in ContinuousDoubleAuction stage. 117. Subject revoked vote to end the period in ContinuousDoubleAuction stage.
ID	Unique identifier of this entry
Period	Period number of the record
Subject	Subject event was triggered by, (0) if not applicable
Time	Time in seconds since start of treatment

contracts

(Life: Period)

Note that in the case of Call Auctions, buy and sell offers are not directly matched with each other. For this reason, executed buy and sell offers trigger separate contracts. This implies that for example calculating the sum of all transaction volumes in the contracts table would yield a number twice as large as the number of shares actually traded.

Variable	Content
Accepter	Subject ID of accepting subject.
ID	Contracts table record ID.
Market	Market ID.
Offerer	Subject ID of offering subject.
OfferID	Corresponding ID in the offers table.
OfferTime	Time offer was originally created.
Period	Period number of the record
Phase	Entry stems from call auction (1) or continuous double auction (2).
Price	Price.
Status	Traded (1), cancelled (2).
Time	Time of this event.
Transacted	Transacted volume after this transaction.
TransactionID	Corresponding ID in the transactions table.
Type	Purchase (1) or sale (2).
Volume	Original offered volume.
VolumeTraded	Volume traded in this transaction.

summary

(Life: Treatment)

Variable	Content
Period	Period number of the record

session

(Life: Session)

Variable	Content
FinalProfit	Profit to be paid (in EUR) not including the show up fee.
IsExperimenter	Copied from subjects.IsExperimenter at the end of each period. Used to allow experimenter subjects to skip all questionnaire forms except for the address form.
MoneyAdded	Pre-defined z-Tree variable. Credit limit of the subject.
MoneyEarned	= FinalProfit + ShowUpFee.
MoneyToPay	= FinalProfit + ShowUpFee + MoneyAdded.
Participate	Variable determining whether a subject sees a particular form of a questionnaire (1) or not (0). Reset to (1) for each new form in a questionnaire, but can be modified using a program in the form dialog.
ShowUpFee	Pre-defined z-Tree variable.
ShowUpFee-Invested	Pre-defined z-Tree variable. 1, if subject agreed to invest show up fee to be able to proceed in the experiment.
Subject	Subject number of the record

profit

(Life: Treatment)

This table contains summary data for every subject in every period and can be used to display a list of previous periods' results, or for randomly choosing one period for payoff (see \RandomPayoffPeriod). Additional variables of interest can be added to it by modifying the program in the background and the program in the FinLitResult stage.

Variable	Content
Assets[\NumMarkets]	Equals subjects.Assets[.] in this period.
Cash	Equals subjects.cash in this period.
Period	Period number of the record
Profit	Equals subjects.profit in this period.
Subject	Subject number of the record

4. Solution examples

Throughout this section, when we refer to "P1", we mean the globals table program titled "//### P1: General parameters ###" in GIMS' Background; by "P2" we mean "//### P2: Stage settings ###", etc.

4.1. Running a practice period

GIMS is designed for practice periods to be run in a separate treatment, but not using a separate treatment file (.ztt). What does this mean? The idea is that you design the program you need for your experiment and use this same program for your practice period(s). This way, if you want to make a change to the programming, you only have to make it in the one treatment file you use for practice and main experiment. To run one or multiple practice periods, you then do the following:

a. Set the number of periods in the Background to the desired number of practice periods
b. Set \PracticePeriods=1 in P1.

If you run the treatment now, everything will run as if this was a normal experiment, but subjects' profits will not be written to the subjects.Profit variable. This means that z-Tree will not consider the earnings from the practice periods when it comes to subject payment. Once the practice periods are over and you want to run your experiment, do the following:

c. Set the number of periods in the Background to the desired number of periods in your experiment
d. Set \PracticePeriods=0 in P1.

If you want to modify parts of your program for the practice periods (e.g., skip displaying a results screen), you can condition on the \PracticePeriod variable to do so. For example, you can skip an entire stage by setting its Participate condition to:

Participate=(1-\PracticePeriod)

4.2. Changing subjects' endowments

Modifying subjects' cash and asset endowments is something that nearly every experiment is likely to require. Please start by reading section 2 to understand how GIMS saves subjects' endowments. Let's assume you want half of your subjects to get 5 assets and 100 experimental currency units (ECU) in cash, and the other half to get 10 assets and 50 ECU. You would do the following:

a. Open P8. By default, it creates three different endowment types. Modify it to read as follows:

```
//### P8: Endowment assignment ###

endowments.new {
    Type=1;
    Cash=100;
    Assets[1]=5;
    }
endowments.new {
    Type=2;
    Cash=50;
    Assets[1]=10;
    }

EndowmentFillerType=2;
```

Code block 1 Endowment assignment

You have now defined two endowment types. GIMS will by default randomly assign these types to your subjects such that each type is assigned to an equal number of subjects. If this is not possible (e.g., if you have5 subjects), it will use the variable \EndowmentFillerType to determine which endowment type to assign to the remaining subjects. In this example, it would try to assign type 1 and type 2 to one subject each and repeat this until all subjects have been assigned an endowment type, or until the remaining number of subjects is no longer divisible by the number of endowment types. In our example, it would assign type 1 and type 2 to one subject each. At this point, four subjects have been fitted with endowment types and one subject has not. GIMS will now assign the type specified in \EndowmentFillerType to the remaining subject. Thus, you would end up with two subjects with endowment type 1, and three subjects with endowment type 2. Note that in this example, the setting was made for market 1 only. If you are running a multi-market experiment, you also need to specify Assets[2], Assets[3], etc. as needed.

4.3. Drawing a set of dividends once and reusing them in multiple sessions

In order to limit the variation between markets from different sessions, some studies use the same random dividend draw in multiple markets (e.g., Kirchler et al. 2011). In other words, they randomly draw a dividend for each of the periods in the experiment, then use the period drawn for the first (and second, and third…) period for the first (and second, and third…) period of the markets in multiple sessions. GIMS implements this possibility through the option of importing a table of pre-drawn dividends which are then used for the current market. To do this, proceed as follows:

a. Use a random number generator to draw dividends for every period.
b. Save the dividends in a tab-separated ASCII file named dividends_preset.zdata, using the following structure:

dividends_preset	Period	Market	Set	Dividend
dividends_preset	1	1	1	10
dividends_preset	1	1	2	20
dividends_preset	2	1	1	60
dividends_preset	2	1	2	40

Table 1 Pre-drawn dividends table

c. In P3, set PresetDividends=1 and DividendSet=1.

Now the first period's dividend will be equal to 10, while the second period's will be 60. Through the DividendSet and Set variables, GIMS offers the option of pre-defining different sets of dividends which can be chosen to be applied by simply setting one variable in P3.

4.4. Allowing trading in multiple markets/assets

This one is easy:

a. In P3, set NumMarkets to the number of markets/assets you wish to implement.

The trading interfaces for the call auction and the continuous double auction are so far designed for trading in 5 assets but can be modified for more assets at any time.

4.5. Allow subjects to sell short up to 5 units of the asset

Very easy:

b. In P7, set ShortingCapacityAssets[1]=5.

You are done. Subjects may now have a negative asset inventory not "exceeding" 5 units. They automatically pay the dividends due on these units to the subjects they have borrowed them from. Note that in this example, the setting was made for market 1.

4.6. Letting subjects vote for an early ending of a continuous double auction

Sometimes, the experimenter wants to give subjects the option of ending a contiuous double auction before the pre-set time for trading runs out. An example is Smith et al. 1988, who implemented this option to prevent boredom on the parts of their subjects in the case where no subject wished to continue trading in the current period. Such a feature can be implemented in GIMS with the following setting:

a. In P3b, set EarlyEndingVote=1.

With this setting, traders get the option of voting for the end of a period at the bottom of the screen:

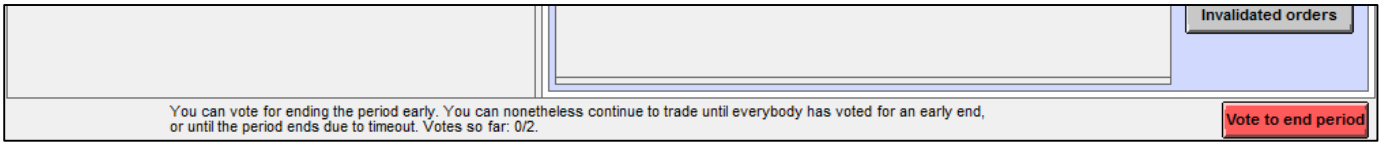

Figure 2 Screenshot Vote to End Period Early

They can continue trading until the vote is unanimous or until the trading phase times out even if they have voted to end the period. They can also, at any time, rescind (and recast) their vote.

Pay a lump-sum at the end of the experiment in addition to the periodic dividends

In order to achieve different fundamental value trajectories, some authors have resorted to paying a lump-sum dividend at the end of the last period of trading in a market in addition to the periodic dividend payments (e.g., Smith et al. 2000, Noussair et al. 2001 and Kirchler et al. 2012).

4.7. Allowing only one limit bid and one limit ask per subject in a sealed bid-offer call auction

In some experiments, subjects should be able to submit one limit bid and one limit ask in a call auction market mechanism (an example would be Haruvy et al. 2007). To program this, do the following:

a. In program P2, set CallAuction=1 and ContinuousDoubleAuction=0 (unless you want subjects to trade both in a call auction and in a continuous double auction).
b. In P3a, set NumTypedOffersCallAuction=1 and AllowMarketOrders=0.

You are done.

5. Overview of call auction mechanism

Note that the total graphs of price determination and trade settlement (i.e., a single file for each) are available from http://academic.palan.biz/downloads/gims.

5.1. Price determination

5.1.1. Schematic overview of call auction price determination mechanism in GIMS

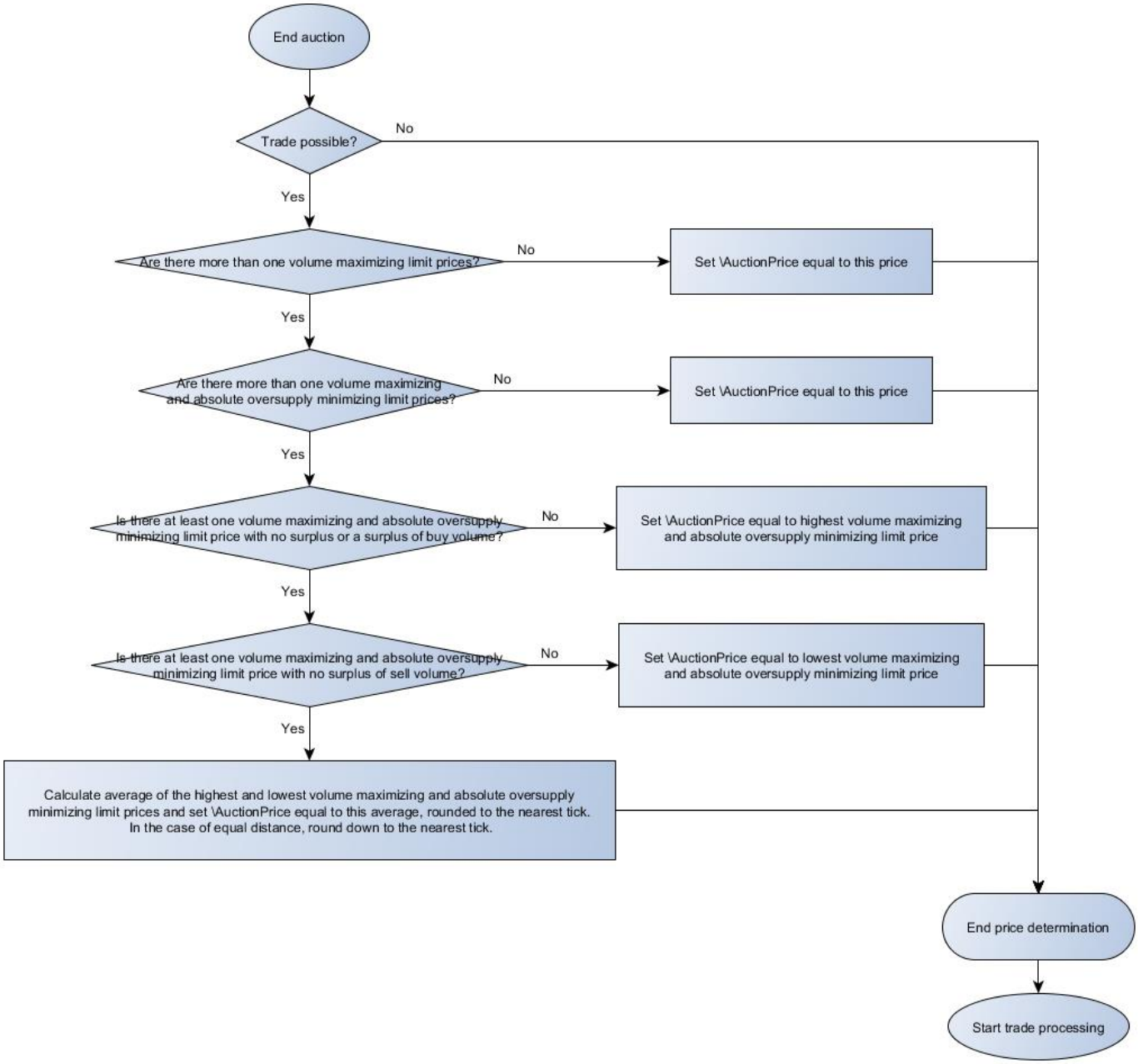

Figure 3 Price determination overview

5.2. Trade settlement

5.2.1. Overview

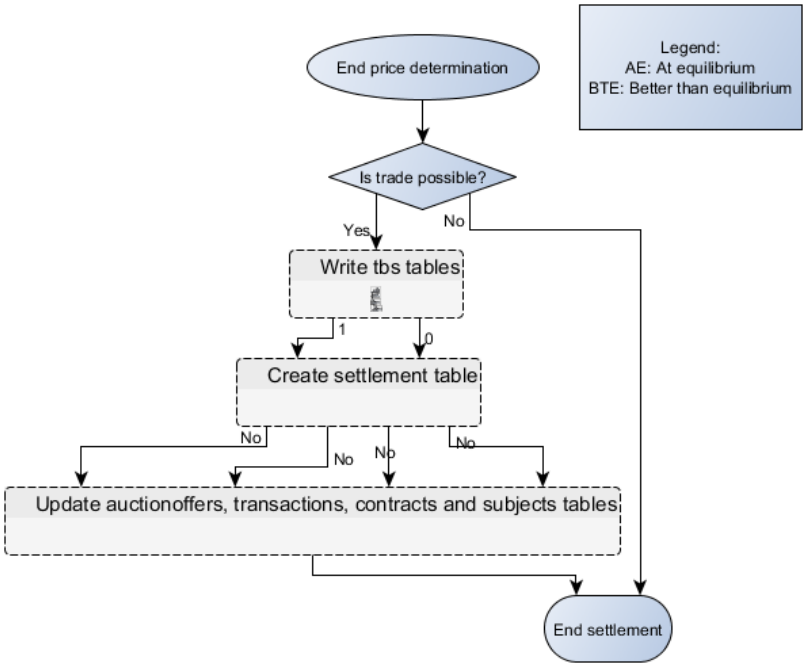

Figure 4 Trade settlement overview

5.2.2. Writing of tbs tables

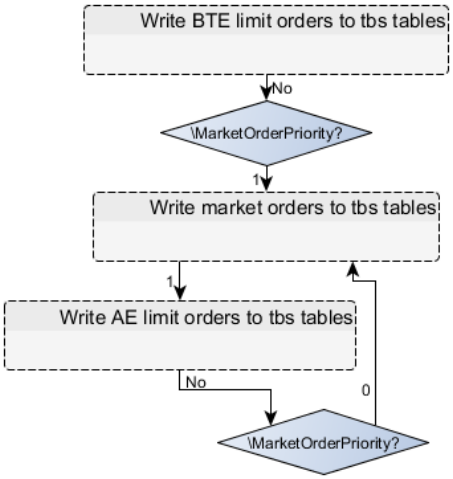

Figure 5 Trade settlement - Writing of tbs tables

5.2.2.1. Writing BTE limit orders to tbs tables

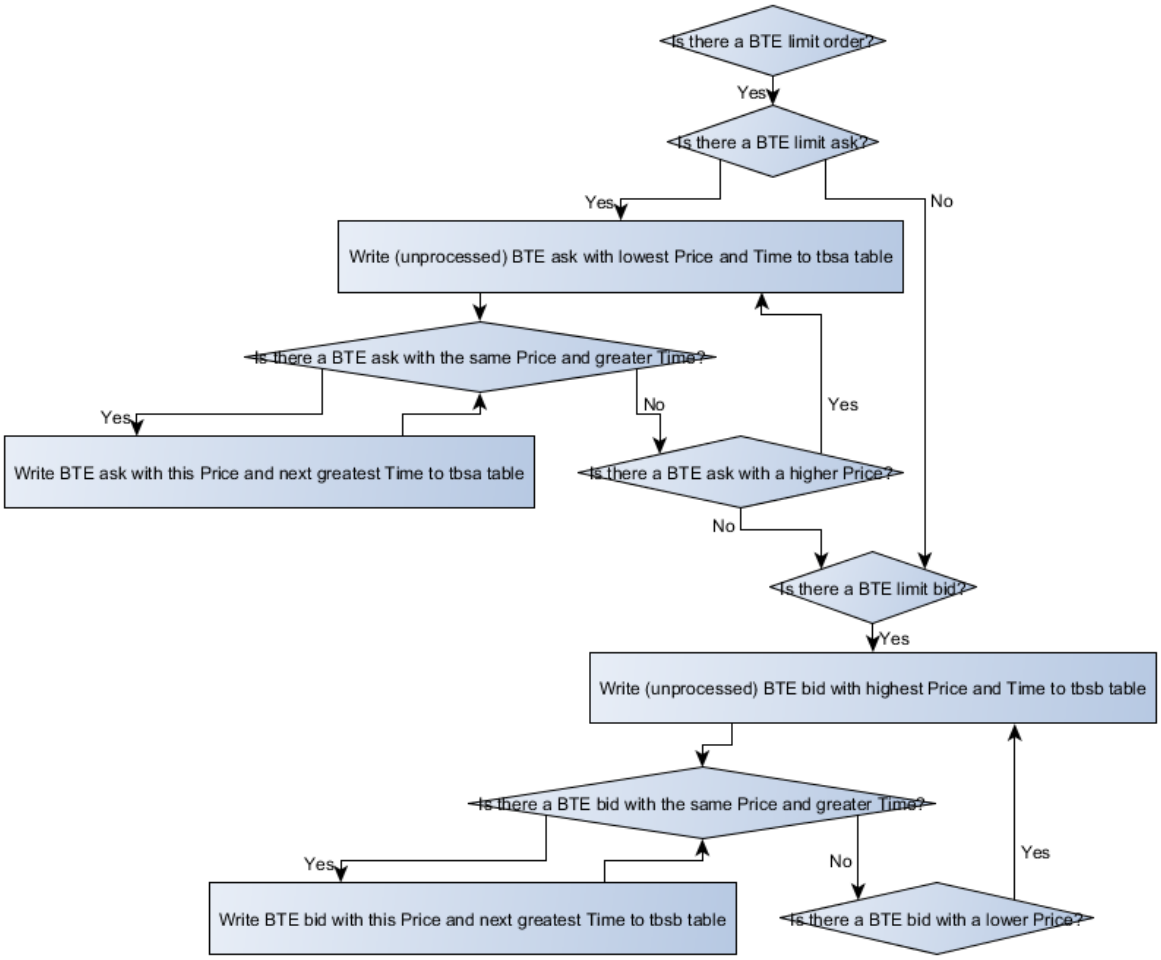

Figure 6 Price determination - Writing BTE limit orders to tbs tablesPrice determination overview

5.2.2.2. Writing market orders to tbs tables

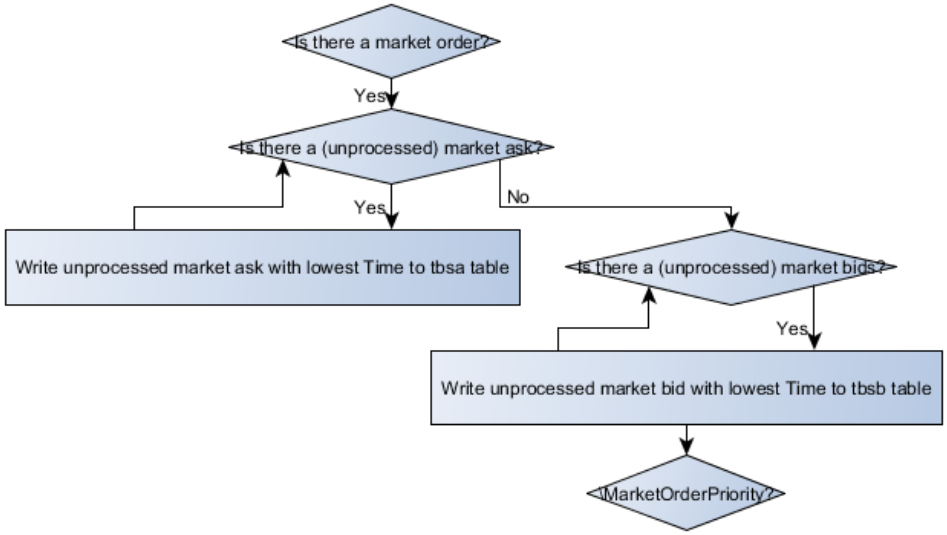

Figure 7 Price determination - Writing market orders to tbs tables

5.2.2.3. Writing AE limit orders to tbs tables

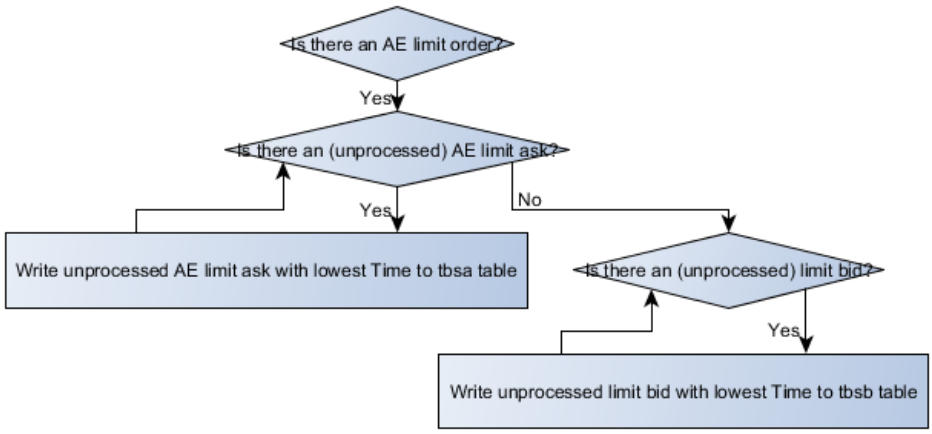

Figure 8 Price determination - Writing AE limit orders to tbs tables

5.2.3. Creation of settlement table

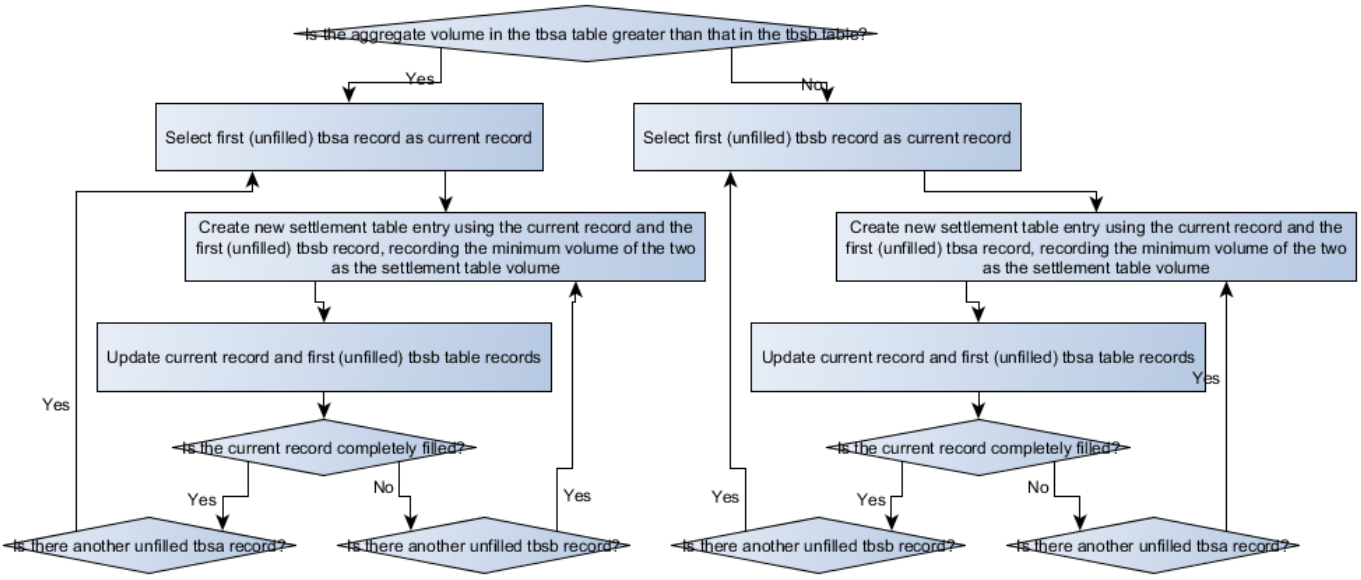

Figure 9 Trade settlement - Creation of settlement table

5.2.4. Bookkeeping

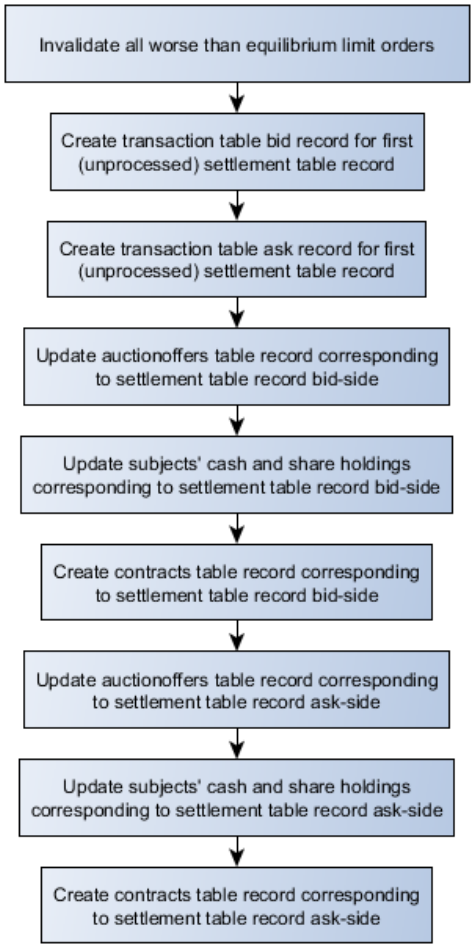

Figure 10 Trade settlement - Bookkeeping

6. Version history

The current GIMS version is v7.2.1. The current version of the questionnaires is v1.0. Note that previous versions of GIMS will be provided indefinitely for future reference. Furthermore, if you have suggestions for changes in GIMS, or if you have programmed additional functionality, please contact gims@palan.biz if you want them to be included in GIMS. Published versions are highlighted in green (all others are development versions which were not made public).

Version	Date	Changes
Known issues:		- The time display in the header bar does not work correctly due to a z-Tree bug, which was reported to the programmers of z-Tree on 03.12.2013. The problem can be circumvented by removing either the German or English language version of the header box in the background (whichever is not needed in a given experiment). - Due to z-Tree limitations, the parameter settings for output precision (of prices, volumes, etc.) cannot be implemented in all output items. Thus, some precision settings have to be changed in the code if modified precision settings are necessary.
Feature requests:		-
7.2.1	27.07.2015	Fixed: - Error in checking sell order feasibility in CallAution stage in multi-asset setting.
7.2	27.07.2015	Features: - Added practice period results screen to CallAuctionResults stage.
7.1.2	16.07.2015	Features: - Finished work on multi-asset trading in ContinuousDoubleAuction stage. Changed: - Price highlighting in ContinousDoubleAuction rewritten completely. Fixed: - No English language market results list display in CallAuctionResults. - Waiting screen in ContinuousDoubleAuctionResults could not be left using “Continue” button if \CallAuction==1.
7.1.1	15.07.2015	Features: - Continued work on multi-asset trading in ContinuousDoubleAuction stage.
7.1	13.07.2015	Features: - Started work on multi-asset trading in ContinuousDoubleAuction stage. - Prices are highlighted in case of price changes.
7.0.10	02.07.2015	Fixed: - Time when subject submitted orders in CallAuction stage was not being logged.
7.0.9	25.06.2015	Fixed: - Corrected double-counting of call auction orders in case of multiple markets.
7.0.8	13.05.2015	Fixed: - Spelling mistake.
7.0.7	12.05.2015	Features: - Implemented preset buyback value functionality.
7.0.6	08.05.2015	Features: - Testing of multi-market functionality. - Modifying screen-layout for multi-market functionality. - Implemented safety measure to prevent memory overrun in case of excessively short \IndicativePriceInterval. Fixed: - Signals container in the CallAuction and ContinuousDoubleAuction stages had English caption independent of \Language.
7.0.5	07.05.2015	Features: - Testing of multi-market functionality.

Version	Date	Changes
		- Added check to make sure subjects cannot submit market buy orders which, if executed at the minimum permissible price, combined with the subject's other outstanding buy orders exceed the subject's cash endowment. Fixed: - Subjects can submit orders which collectively exceed their cash endowment in CallAuction. - Subjects can submit orders which collectively exceed their asset endowment in CallAuction.
7.0.4	06.05.2015	Features: - Testing of multi-market functionality. Fixed: - CallAuctionResults screen showed no auction price if \IndicativePriceInterval > 0.
7.0.3	05.05.2015	Features: - Continued work on multi-market functionality.
7.0.2	04.05.2015	Features: - Continued work on multi-market functionality.
7.0.1	30.04.2015	Features: - Continued work on multi-market functionality. Fixed: - Experimenter saw signals in CallAuction stage iff ContinuousDoubleAuctionSignals is set to 1.
7.0	28.04.2015	Features: - GIMS now runs on z-Tree 3.4.7. - Started work on multi-market functionality.
6.7	11.12.2014	Features: - Added profit table to allow for recording information for each subject and period and for paying randomly chosen period.
6.6.4	17.11.2014	Fixed: - Signals container in the CallAuction stage could (only) be turned off by using globals.ContinuousDoubleAuctionSignals.
6.6.3	30.04.2014	Fixed: - Doubled status time recording when cancelling orders. - Offers cannot be cancelled without entering a number in the volume field.
6.6.2	28.04.2014	Fixed: - Time displayed in continuous double auction stage chart starts at program start, not at CDA start. - "minutes" instead of "Minuten" in CRT DE question 2.
6.6.1	16.04.2014	Fixed: - Duplicate line on FinLit results screen. - Experimenter has to click "Continue to CRT and FinLit" after call auction even if the continuous double auction is also set to run. - Role to prevent submitting bids with prices greater than the lowest ask price (and vice versa) does not work in call auction. - Bids and asks are, in some settings, sorted in reverse order of submission time.
6.6	15.04.2014	Features: - Added market displays to experimenter screen in CallAuction and ContinuousDoubleAuction stages. Fixed: - Wording of dialog box upon submission of more bids/asks than allowed in call auction and continuous double auctions are unsuitable in case of limit of only 1 bid and ask.

Version	Date	Changes
6.5.2	14.04.2014	Fixed: - Display of vote to end period early text does not look good in 1024x768.
6.5.1	13.04.2014	Features: - Redefined some layout elements from fixed to table-customizable. Other: - Restructured programs containing parameter settings.
6.5	10.04.2014	Features: - Completely redesigned interface, optimized for greater clarity and also “nicer” look. - Introduced central layout table to hold color definitions for all screen elements. - Introduced possibility to pay a final buyback value for units of the asset. - Added option to let subjects vote to end the period in the continuous double auction.
6.4	09.04.2014	Features: - Added option to select whether subjects should be paid based on the cash balance at the end of every period, or only based on the cash balance at the end of the last period. Fixed: - Buy and sell offer lists scroll to end instead of beginning. Other: - Generated exit questionnaire versions 1.0 in English and German.
6.3	08.04.2014	Features: - Added option to limit the number of offers displayed in the order book. Experimenter can thus limit the number of bids (asks) displayed to a fixed number. Orders beyond this number are only displayed for the subject having submitted them. Fixed: - Error in offer acceptance check allows offer volume to go negative if volume of the offer was previously partially invalidated and subject offers to accept more than the available volume. - Time priority is not strictly enforced for offers of equal price in continuous double auction.
6.2	07.04.2012	Features: - Added capacity to short assets and cash (buy on margin).
6.1.2	03.04.2014	Fixed: - Subjects can accept worse than the best bids and asks. - In continuous double auction, upon accepting a buy offer, the system checks the offer’s total volume instead of the accepter’s desired volume against the accepter’s asset balance. - Chart is not being displayed in continuous double auction after call auction.
6.1.1	02.04.2014	Fixed: - Buy/sell offer columns in both market types are not labelled and thus not readily distinguishable. - Overlapping fields in call auction order entry with multi-unit trading enabled. - Error in determination of indicative price.
6.1	20.03.2014	Features: - Added option to empty the order book after each trade (“classical” double auction). Fixed: - Invalidated orders are not displayed in order history table.
6.0.6	19.03.2014	Features: - Dynamic chart resizing extended to values between 5,000 and ~100,000. Fixed: - Endowment table only filled in first period. - Chart in ContinuousDoubleAuction stage displays transactions from CallAuction stage if both are enabled.

Version	Date	Changes
		- Alignment of summary information items (Cash, Assets vs. Last Price) in continuous double auction differs.
6.0.5	13.03.2014	Features: - Added specific message if subject does not enter any offer in call auction.
6.0.4	12.03.2014	Features: - New option to display (or not) call auction transactions on continuous double auction screens (switch \ConnectedMarketMechanisms).
6.0.3	11.03.2014	Features: - Introduced parameters AssetPrecision, CashPrecision, PricePrecision and VolumePrecision for asset number, cash, price and volume input and output display precision. Fixed: - German language price display in call auction. - Missing button to leave call auction stage.
6.0.2	05.03.2014	Features: - Created option to carry over (or not) wealth from previous period.
6.0.1	04.03.2014	Features: - Created parameter to set signal display precision. Fixed: - Experimenter-induced forced exit from experiment only forced exit from period.
6.0	27.02.2014	Features: - Signal module completed.
5.7.1	22.01.2014	Features: - Call auction module completed. Fixed: - Corrected bug in CDA history display (German despite English setting)
5.7	14.01.2014	Call auction without market order submission
5.6	19.12.2013	Call auction indicative price determination implemented
5.5		Call auction settlement v2
5.4		Call auction settlement v1
5.3		Call auction price determination v2
5.2		Call auction price determination v1
5.1		Call auction order accounting completed
5.0		Multi-Language support completed
4.0		Holt Laury module completed
3.0		FinLit module completed
2.0		CRT module completed
1.0		Continuous double auction module completed

7. License information and disclaimer

8. Publication bibliography

Fischbacher, Urs (2007): z-Tree: Zurich toolbox for ready-made economic experiments. In *Experimental Economics* 10, pp. 171–178. DOI: 10.1007/s10683-006-9159-4.

Fischbacher, Urs; Kendrick, Katharine; Schmid, Stefan (2015): z-Tree 3.4. Tutorial and Reference Manual. University of Zurich, updated on 1/6/2015.

Frederick, Shane (2005): Cognitive Reflection and Decision Making. In *Journal of Economic Perspectives* 19 (4), pp. 25–42.

Haruvy, Ernan; Lahav, Yaron; Noussair, Charles N. (2007): Traders' Expectations in Asset Markets. Experimental Evidence. In *American Economic Review* 97 (5), pp. 1901–1920.

Holt, Charles A.; Laury, Susan K. (2002): Risk Aversion and Incentive Effects. In *American Economic Review* 92 (5), pp. 1644–1655.

Kirchler, Michael; Huber, Jürgen; Kleinlercher, Daniel (2011): Market microstructure matters when imposing a Tobin tax. Evidence from the lab. In *Journal of Economic Behavior & Organization* 80 (3), pp. 586–602. DOI: 10.1016/j.jebo.2011.06.001.

Kirchler, Michael; Huber, Jürgen; Stöckl, Thomas (2012): Thar She Bursts - Reducing Confusion Reduces Bubbles. In *American Economic Review* 102 (2), pp. 865–883.

Noussair, Charles N.; Robin, Stéphane; Ruffieux, Bernard (2001): Price Bubbles in Laboratory Asset Markets with Constant Fundamental Values. In *Experimental Economics* 4, pp. 87–105.

Smith, Vernon Lomax; Suchanek, Gerry L.; Williams, Arlington W. (1988): Bubbles, Crashes, and Endogenous Expectations in Experimental Spot Asset Markets. In *Econometrica* 56 (5), pp. 1119–1151.

Smith, Vernon Lomax; van Boening, Mark V.; Wellford, Charissa P. (2000): Dividend timing and behavior in laboratory asset markets. In *Economic Theory* 16, pp. 567–583.

van Rooij, Maarten; Lusardi, Annamaria; Alessie, Rob (2011): Financial literacy and stock market participation. In *Journal of Financial Economics* 101 (2), pp. 449–472. DOI: 10.1016/j.jfineco.2011.03.006.