

Technical Note: Parameterised-Response Zero-Intelligence (PRZI) Traders

Dave Cliff

Department of Computer Science

University of Bristol

Bristol BS8 1UB, UK

csdtc@bristol.ac.uk

Abstract—This brief technical note introduces PRZI (Parameterised-Response Zero Intelligence), a new form of zero-intelligence trader intended for use in simulation studies of auction markets such as those used in major financial markets around the world. Like Gode & Sunder’s classic Zero-Intelligence Constrained (ZIC) trader, PRZI generates quote-prices from a random distribution over some specified domain of discretely-valued allowable quote-prices. Unlike ZIC, which uses a uniform distribution to generate prices, the probability distribution in a PRZI trader is parameterised in such a way that its probability mass function (PMF) is determined by a real-valued control variable s in the range $[-1.0, +1.0]$ that determines the *strategy* for that trader. When s is zero, a PRZI trader behaves identically to the ZIC strategy, with a flat/rectangular PMF; but when s is close to plus or minus one the PRZI trader’s PMF becomes asymptotically maximally skewed to one extreme or the other of the price-range, thereby enabling the PRZI trader to act in the same way as the “Shaver” strategy (SHVR) or the “Giveaway” strategy (GVWY), both of which have recently been demonstrated to be surprisingly dominant over more sophisticated, and supposedly more profitable, trader-strategies that incorporate adaptive mechanisms and machine learning. Depending on the value of s , a PRZI trader will behave either as a ZIC, or as a SHVR, or as a GVWY, or as some hybrid strategy part-way between two of these three previously-reported strategies. The novel smoothly-varying strategy in PRZI has value in giving trader-agents plausibly useful “market impact” responses to imbalances in an auction-market’s limit-order-book, and also allows for the study of co-adaptive dynamics in continuous strategy-spaces rather than the discrete spaces that have traditionally been studied in the literature. Python source-code for a PRZI trader has been made publicly available on GitHub, as a reference implementation.

I. INTRODUCTION

In attempting to understand and predict the fine-grained dynamics of financial markets, there is a long tradition of studying simulation models of such markets. Simulation studies nicely complement the two primary alternative lines of enquiry: analysis of real market data recorded at fine-grained temporal resolution, as is studied in the branch of finance known as *market microstructure*; and running carefully planned experiments where human subjects interact in artificial markets under controlled laboratory conditions, an approach known as *experimental economics*. Simulation modelling of financial markets very often involves populating a market mechanism with some number of *trader-agents*: autonomous

entities that have “agency” in the sense that they are empowered to buy and/or sell items within the particular market mechanism that is being simulated: this approach, known as *agent-based computational economics* (ACE), has a history stretching back for more than 30 years. Over that multi-decade history, a small number of specific trader-agent algorithms, i.e. precise mathematical and procedural specifications of particular trading strategies, have been frequently used for modelling various aspects of financial markets, and the convention that has emerged is to refer to each such strategy via a short sequence of letters, reminiscent of a stock-market ticker-symbol. Notable trading strategies in this literature include (in chronological sequence): SNPR [1], ZIC [2], ZIP [3], GD [4], MGD [5], GDX [6], and AA [7]; several of which are explained in more detail later in this paper. Of these, ZIC [2] is notable for being both highly stochastic and extremely simple, and yet it gives surprisingly human-like market dynamics; GD and ZIP were the first two strategies to be demonstrated as superior to human traders, a fact first established in a landmark paper by IBM researchers [8] (see also: [9]–[11]), which is now commonly pointed to as initiating the rise of algorithmic trading in real financial markets; and until very recently AA was widely considered to be the best-performing strategy in the public domain. With the exception of SNPR and ZIC, all later strategies in this sequence are adaptive, using some kind of machine learning (ML) or artificial intelligence (AI) method to modify their responses over time, better-fitting their trading behavior to the specific market circumstances that they find themselves in, and details of these algorithms were often published in major AI/ML conferences and journals.

The supposed dominance of AA has recently been questioned in a series of publications [12]–[16] which demonstrated AA to have been less robust than was previously thought. Notably, [15], [16] report on trials where AA is tested against two novel algorithms that each involve no AI or ML at all: these two newcomer strategies are known as GVWY and SHVR [17], [18], and each share the pared-back minimalism of Gode & Sunder’s ZIC mechanism. In the studies that have been published thus far, depending on the circumstances, it seems (surprisingly) that GVWY and SHVR can each outperform not only AA but also many of the other AI/ML-based trader-agent strategies in the set listed above. Given this surprising recent result, there is an appetite for

further ACE-style market-simulation studies involving GVWY and SHVR. One compelling issue to explore is the co-adaptive dynamics of markets populated by traders that can choose to play one of the three strategies from GVWY, SHVR, and ZIC, in a manner similar to that studied by [19] who employed ‘replicator dynamics’ modelling techniques borrowed from theoretical evolutionary biology to explore the co-adaptive dynamics of markets populated by traders that could choose between SNPR, ZIP, and GD.

Although one way of studying coadaptive dynamics in markets where the traders can choose to either deploy GVWY, SHVR, or ZIC is to give each trader a discrete choice of one from that set of three strategies, so at any one time any individual trader is either operating according to GVWY or SHVR or ZIC, it is appealing to instead design experiments where the traders can continuously vary their trading strategy, exploring a potentially infinite range of differing trading strategies, where the space of possible strategies includes GVWY, SHVR, and ZIC; and that is the motivation for this paper. Here, I introduce a new trading strategy that has a *parameterised response*: that is, its trading behavior is determined by a strategy parameter $s \in [-1, +1] \in \mathbb{R}$. When $s = 0$, the trader behaves identically to ZIC, and when $s = \pm 1$ it behaves the same as GVWY or SHVR. As is explained in more detail later in this paper, GVWY, ZIC, and SHVR are each members of the class of *zero intelligence* (ZI) trading strategies, and hence I’ve named the new strategy described here as the Parameterized-Response Zero-Intelligence (PRZI) trading strategy.

While one motivation for devising PRZI was just described: to enable explorations of coadaptive dynamics in continuous strategy-spaces, that is not the only motivation. Two other compelling reasons for wanting a ZI-style trader with a variable response are as follows:

- Recent publications [20], [21] have described methods for making these simple trader-agents exhibit a form of sensitivity to temporary imbalances between supply and demand in the marketplace, and that imbalance-sensitivity then gives rise to so-called “market impact” effects, where the prices quoted by traders shift in the *anticipated* direction of future transaction prices, where the shift is anticipated from the degree of supply/demand imbalance instantaneously evident in the market. Market impact is a significant issue for traders in real markets who are looking to buy or sell an unusually large amount of some asset, and major exchange operators such as the London Stock Exchange have introduced specialised mechanisms to try to reduce the ill-effect of market impact. For markets populated by ZI trader-agents to be able to exhibit impact effects, the traders need to be able to modulate their trading activity according to the direction and degree of imbalance in the market, becoming either more “urgent” or more “relaxed” in their trading: varying PRZI’s strategy-parameter s implements exactly this kind of tuneable response.
- Nobel Laureate Robert Shiller has recently proposed [22], [23] that certain economic phenomena which defy easy

explanation via classical assumptions of individual economic rationality can be better understood by reference to the *narratives* (i.e., the stories) that economic agents tell themselves and each other about current and future economic factors. Shiller refers to this new approach as *narrative economics*. Noting that stories are merely externalisations of an agent’s internally-held *opinions*, a recent publication [24] described an agent-based modelling platform for studying issues in narrative economics, in which two types of ZI traders were extended to also each include a real-valued *opinion* variable (the value of which could be altered by interactions with other agents, thereby modelling the way in which an agent’s opinions are shifted by the narratives it is exposed to) and adapted so that their trading strategies alter as a function of their individual opinion: this also gives rise to a need for ZI traders that can smoothly vary the nature of their trading behavior, and PRZI was designed with the intention of being used in such opinion dynamics modelling work.

This paper is intended merely as a brief technical note introducing PRZI; it is beyond the scope of this document to provide a comprehensive and detailed literature review. Readers seeking further details of market microstructure are referred to [25]–[27], and for overviews of experimental economics see for example [28]–[30]. For reviews of ACE research, see [31]–[34]; and for discussions of ZI traders in finance research, see e.g. [35], [36]. Further details of the trader-algorithms discussed above are available as follows: SNPR is a present-day instantiation of Kaplan’s 1990 *Sniper* strategy described in [1]; ZIC was introduced by Gode & Sunder [2]; GD by Gjerstad & Dickhaut [4]; ZIP by Cliff [3]; MGD by Tesauro & Das [5]; GDX by Tesauro & Bredin [6]; AA by Vytelingum [7], [37]; and both GVWY and SHVR by Cliff [17], [18].

II. BACKGROUND: EXPERIMENTAL ECONOMICS

In a landmark 1962 paper [38] published in the prestigious *Journal of Political Economy* (JPE) the economist Vernon Smith described a seminal set of experiments in which human traders interacted in a *continuous double auction* (CDA), the mechanism embodied in most major real-world financial markets, under laboratory conditions. Smith’s 1963 JPE paper is now widely cited as marking the birth of experimental economics, and Smith’s work in this field led to him being awarded the 2002 Nobel Prize in Economics. In the simplest case, such experimental economics work involves a market with only one type of a tradeable asset (think of it as a stock market on which only one stock is listed), and each human trader can buy or sell a specific quantity of the asset by issuing one or more *quotes* to the market’s central exchange. A quote would specify: the trader’s desired *direction* (i.e., buy or sell) for the transaction; the quantity (number of units) that the trader is seeking to transact; and the price-per-unit that they want to pay or be paid. Each trader would be given instructions, referred to here as *assignments*, that are private and specific to that individual trader, and each trader is told to

keep these instructions secret. Some traders will be instructed to buy some quantity of the asset, paying no more than a trader-specific maximum price per unit (i.e., these traders are *buyers* each with a specific *limit price*); other traders will have been instructed to sell some quantity of the asset, accepting no less than some trader-specific minimum price per unit (so they are *sellers*, again each with their own limit-price). In this way, instructing some traders to be buyers and other traders to be sellers, and by varying the limit-prices in each trader's instructions, the market's underlying supply and demand curves could be controlled, along with that market's competitive equilibrium price (denoted by p_0) and equilibrium quantity (q_0). These two values, p_0 and q_0 , are significant because, cutting a long story short, economists care deeply about whether transaction-prices converge to p_0 or not, and if they do converge on p_0 then economists care about the speed and stability of that convergence. These *equilibration* properties of a market matter because, in very colloquial terms, if transactions are consistently taking place at prices either above or below the p_0 value then one side or the other, either the buyers or the sellers, are being consistently ripped off. Smith's 1962 paper (which reported on a sequence of experiments that he had commenced several years earlier) was notable for establishing a set of experiment methods that have since been reproduced and replicated by researchers around the world, and also for being the first empirical demonstration that CDA markets could show reliable equilibration even when populated with only very small numbers of buyers and sellers.

In many experimental economics studies, equilibration is not the only factor of interest. Another significant question is how much *surplus* is extracted from the market by the specific trading behaviors of the traders. In everyday language, surplus can be thought of as a seller's profit or a buyer's saving: if a seller has been given a limit-price of \$10 per unit, but manages to agree a transaction at \$15, then the \$5 difference between the seller's limit-price and the sale-price is that seller's surplus, her profit; similarly if a buyer is given a limit-price of \$10 but manages to instead buy for \$8, then her saving, her surplus, is \$2.

The initial experiments reported by Smith in 1962 were conducted with entirely manual issuing of assignments to traders, and with the traders simply shouting out their quote-prices in a laboratory version of an open-outcry trading-pit, a common sight on the trading floors of major exchanges for many decades prior to the arrival of automated market-trading technology. More recently, as in real financial exchanges, so in experimental economics: most experimental economics studies for the past 30 years or more have involved the human traders interacting with one another by each being sat at an individual trader-terminal (e.g. a PC running specialised trader-interface software, networked to a central exchange-server PC). The display-screen on a trader-terminal (whether in a real market or in a laboratory experiment) will often show a summary of all the currently outstanding bid-quotes received from buyers, and all the currently outstanding ask-quotes received from sellers, in a tabular form known as a *Limit Order Book* (LOB). Whole

books have been written on the LOB (see e.g. [39]–[41]) but for the purposes of this paper, all we need to know is that the LOB allows all traders in the market to see the best (lowest) ask-price from a seller and the best (highest) bid-price from a buyer. In the rest of this paper I'll use $p_{\text{ask}}^*(t)$ to denote the price of the best ask at time t , and $p_{\text{bid}}^*(t)$ to denote the price of the best bid.

Any study in experimental economics where human traders interact with one another, via some market mechanism, subject to the constraints imposed by the design of the experiment, gives rise to the question of just how big a role the intelligence of the human traders plays in determining the market's equilibration behavior. A genuinely shocking answer to that question was provided in 1993 by Gode & Sunder, also publishing in the JPE [2], who presented results which appeared to show that the answer was simple: zero. That is, Gode & Sunder showed that markets populated entirely by so-called *zero-intelligence* (ZI) traders could give rise to market dynamics, to equilibration behavior, which was statistically indistinguishable from that of human traders, when measured by the then-standard metric known as *allocative efficiency*, i.e. how much of the available surplus in the market was extracted by the traders. Gode & Sunder's ZI traders manifestly have no intelligence at all, and are discussed in more detail in the next section.

III. THREE ZI TRADING STRATEGIES

This section describes the three trading strategies *Zero-Intelligence-Constrained* (ZIC), *Shaver* (SHVR), and *Give-away* (GVWY). As will be seen, all three of these can very reasonably be described as ZI trading strategies.

In the following text, $\mathcal{U}(a, b)$ is used to denote draws from a uniform random distribution over the range $[a, b]$; δ_p denotes the market's *tick-size*, the minimum price change allowed in the market (very often one cent of the national currency in real financial exchanges); and $P_{bq(\text{STRAT})}(t)$ and $P_{sq(\text{STRAT})}(t)$ denotes the prices quoted by a buyer and a seller, respectively, at time t , by a trader of strategy-type STRAT. Note that a capital P is used to denote a price that is (or could be, in principle) randomly generated, a random variable, and the various lower-case p values are nonrandom. Time proceeds in discrete steps of Δ_t , and each strategy is summarised as an equation that specifies the price that will be quoted by a trader at time $t + \Delta_t$ on the basis of information available, the state of the market, at time t . The limit-price assigned to a buyer is denoted by λ_b , and the seller's limit-price is λ_s . The index i is used where necessary to distinguish values that are specific to trader i ; and finally the value $d_i(t) \in \{-1, +1\}$ is the *direction* of trader i at time t : when $d_i(t) = -1$, the trader is buying; when $d_i(t) = +1$, the trader is selling.

A. ZIC

In their seminal 1993 JPE paper [2], Gode & Sunder presented results from three sets of experimental economics studies: in the first, groups of human traders interacted in an electronic CDA, the mechanism found in real-world financial

markets, under laboratory conditions, as described above. As is commonplace in much experimental economics work, Gode & Sunder fixed the limit-quantity to one for all traders, so that transactions always involved a single unit of the asset changing hands, and hence the primary variable of interest was the transaction prices in the market. This first set of experiments established baseline data from the human traders, and Gode & Sunder recorded a key metric, α , first introduced in Smith's 1962 paper, which measures the variation of transaction prices around the the market's p_0 value, i.e. α is a measure of equilibration; and they also recorded the *allocative efficiency* for each market, which is a measure of how much of the total theoretically available surplus is actually extracted by the traders in that market.

In their second set of experiments, they replaced all the human traders with simple autonomous software agents that could electronically issue quotes to the market's central exchange mechanism: as with the human traders in the first experiments, the software agents were each assigned a direction (buy or sell) and given a limit price for their transactions. Gode & Sunder performed two sets of experiments with these software agents: one set with a type of trader-agent that they named *ZI-Unconstrained* (ZIU); and another set with a modified version of the ZIU strategy, one that involved imposition of an additional constraint, and so those traders were named *ZI-Constrained* (ZIC). If ever a seller ZI trader issued a quote-price below the current best bid-price (i.e., in the terminology of financial markets, if the quote is for a price that *crosses the spread*), that seller sold its unit the buyer who had issued that best bid; and similarly if ever a ZI buyer issued a quote-price that was higher than the current best ask-price issued by a seller, the buyer would buy from that seller, because the buyer's quote crossed the spread. (The *spread* is the difference between the current best ask price and the current best bid price).

ZIU traders were very basic: they generated a randomly-selected quote-price drawn from $\mathcal{U}(\delta_p, p_{\max})$, where $p_{\max}$ is an arbitrarily-chosen maximum-allowable price in the market. That is, ZIU traders were designed to ignore their limit prices. Unsurprisingly, the time-series of transaction prices in markets populated by ZIU traders looked much like random noise, and ZIU traders would often enter into loss-making deals, because they were buying at prices above their limit price or selling at prices below their limit price.

Gode & Sunder then modified the ZIU strategy, giving the ZIC strategy, by introducing a just one constraint: to not quote prices that were potentially loss-making. ZIC traders still quote random prices drawn from a uniform distribution over some range, but now there is a difference depending on whether the ZIC trader is a buyer or a seller:

$$P_{bq}(\text{ZIC})(t + \Delta_t) = \mathcal{U}(\delta_p, \lambda_b); \quad (1)$$

$$P_{sq}(\text{ZIC})(t + \Delta_t) = \mathcal{U}(\lambda_s, p_{\max}). \quad (2)$$

That is, a ZIC buyer randomly generates its quote-prices equiprobably from $[\delta_p, \lambda_b]$, where λ_b is that buyer's limit price;

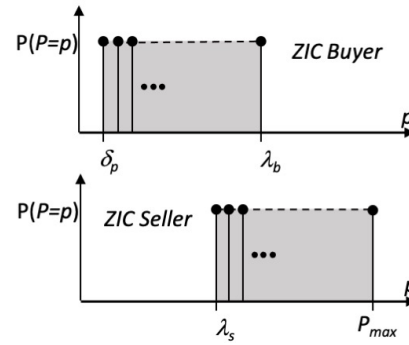


Fig. 1. Illustrative Probability Mass Function (PMF) for the uniformly-distributed discrete quote-prices generated by a ZIC buyer (upper figure) and seller (lower figure).

and a ZIC seller with limit-price λ_s generates its quote-prices equiprobably over the range $[\lambda_s, p_{\max}]$. Surprisingly, markets populated by ZIC traders showed equilibration behaviors (as measured by Smith's α metric) and allocative efficiency scores that were virtually indistinguishable from the comparable human-populated markets. This notable result quickly became highly-cited, as it seemed to demonstrate that if there was any 'intelligence' in the system at all, it was in the CDA market mechanism rather than residing in the traders. Gode & Sunder were also careful to note that a different measure of surplus-extraction, called *profit dispersion*, was worse for ZIC traders than for human traders, and they argued that their results demonstrated that allocative efficiency was no longer as useful a metric as had previously been thought.

Because quote-prices in any market are quantized by that market's tick-size δ_p , the prices quoted by a ZIC trader are samples of a discrete random variable, and the probability mass function (PMF) for that variable has a rectangular profile, because the distribution is uniform, as illustrated in Figure 1.

One problematic aspect of ZIC traders that becomes clear to anyone who actually implements them to run live in an operational market is that while ZIC buyers have a PMF domain that is bounded from below by the smallest nonzero price δ_p and bounded from above by the trader's limit price λ_b , the PMF domain for a ZIC seller is bounded from below by its limit price λ_s but the upper bound is an arbitrary exogenous system limit $p_{\max}$, the highest price allowed in the market. In theory, $p_{\max}$ might appear to be unimportant, but if its value is set to a large multiple of the largest buyer's limit price $\lambda_{b:\max}$ then the ZIC sellers will spend an awful lot of their time quoting at prices $P_{sq}(t) > \lambda_{b:\max}$, i.e. at prices that can never lead to a transaction, and so the market is flooded with unfulfillable ask-quotes, and you have to wait a long time before a ZIC seller just happens to randomly generate a plausible ask, i.e. one for which $P_{sq}(t) \leq \lambda_{b:\max}$. If the only issue of interest in the market is the temporally-ordered sequence of transaction prices, this is not a problem; but if you care about the actual time intervals between successive transactions, that can be greatly affected by setting $p_{\max}$ too

high. Experimenters also need to be careful to ensure that $p_{\max}$ is not set below the highest limit-price assignable to a seller in the market, which can be a problem in practice if the seller limit-prices are generated by an unbounded random-walk process such as geometric Brownian motion with positive drift. We'll return to these issues, the " $p_{\max}$ problem", in Section IV.

B. SHVR

Source-code for the *Shaver* strategy, abbreviated to SHVR, was made public in 2012 [17] when it was released as one of the several trader-agent strategies available in the *Bristol Stock Exchange* (BSE) which is a freely-available open-sourced simulation of a LOB-based financial exchange, written in Python. BSE was initially developed as a teaching resource, used by masters-level students studying on a Financial Technology module taught at the University of Bristol. In the years since it was first released, it has been used by hundreds of students and has increasingly also found use as a trusted and stable test-bed for exploring research questions in ACE.

Like ZIC, SHVR is minimally simple, involving no intelligence at all. Unlike ZIC, SHVR is entirely deterministic. SHVR can be explained very simply: a SHVR buyer sets its quote-price at time $t + \Delta_t$, denoted by $P_{bq(\text{SHVR})}(t)$, to be one tick (i.e., δ_p) more than the current best bid $p_{\text{bid}}^*(t)$, so long as that does not exceed its limit price λ_b , i.e.:

$$P_{bq(\text{SHVR})}(t + \Delta_t) = \min(p_{\text{bid}}^*(t) + \delta_p, \lambda_b); \quad (3)$$

and similarly a SHVR seller sets its quote-price to be:

$$P_{sq(\text{SHVR})}(t + \Delta_t) = \max(p_{\text{ask}}^*(t) - \delta_p, \lambda_s). \quad (4)$$

If the best bid or ask is undefined at time t , e.g. because no quotes have yet been issued, a SHVR buyer will start with a very low $P_{bq}(t)$, and a SHVR seller with a very high $P_{sq}(t)$. If there is no prior trading data available in the market, the low value could be δ_p and the high value $p_{\max}$, i.e. the same lower and upper bounds as for ZIC traders; if there is prior trading data, the low and high initial values could instead be the lowest and highest prices seen in prior trading over some recent time-window.

SHVR was introduced to BSE as something of a joke, as a tongue-in-cheek approximation to a high-frequency trading algorithm. It does serve the purpose of being a minimal illustrative implementation of a trader that actually uses information available from the LOB. The surprising result (discussed in more in [15], [16]) is that if the circumstances are in its favour then SHVR can out-perform well-known strategies like ZIP or AA, which had previously been hailed as examples of AI-powered super-human robot-trader systems. And, in one further surprise, the same study that revealed SHVR can outperform ZIP and AA also revealed that an even simpler strategy, called GVWY, can do just as well as SHVR and sometimes better.

C. GVWY

Like SHVR, source-code for the *Giveaway* strategy (GVWY) was made public when the first version of BSE was released as open-source in 2012 [17]. The correlates of Equations 3 and 4 for GVWY are as follows:

$$P_{bq(\text{GVWY})}(t + \Delta_t) = \lambda_b; \quad (5)$$

$$P_{sq(\text{GVWY})}(t + \Delta_t) = \lambda_s. \quad (6)$$

As can be seen, a GVWY trader simply sets its quote price to whatever its currently-assigned limit price is, regardless of the time. As the name implies, *prima facie* this trading strategy gives away any chance of surplus, because there is no difference between its quote price and its limit price: if its quote results in a transaction at that price, it yields zero surplus.

However, the spread-crossing rule (which is standard in most LOB-based markets) means that it is possible for a GVWY trader to enter into surplus-generating trades. For example, consider a situation in which a GVWY buyer has a limit price $\lambda_b = \$10$, and the current best ask is $p_{\text{ask}}^* = \$7$: when the GVWY buyer quotes its limit price (i.e., $P_{bq}(t) = \lambda_b$), the \$10 price on the quote crosses the spread and so the GVWY buyer is matched with whichever seller issued that best ask, and the transaction goes through at a price of \$7, yielding a \$3 surplus for the GVWY buyer (and yielding whatever surplus is arising for the seller, dependent on that seller's value of λ_s).

In the next section, I explain how PRZI traders have a continuously-variable strategy space that includes GVWY, ZIC, and SHVR: by setting a strategy-parameter s to an appropriate value, PRZI traders will act either as one of those three ZI strategies, or as some kind of novel hybrid, intermediate between two of them; that is, a PRZI trader's response is a parameterised version of one or more of these three ZI trading strategies, hence the name.

IV. PARAMETERISED-RESPONSE ZI TRADERS

A. Motivation

The initial motivation for developing PRZI came from a desire to give ZI traders a sense of *urgency*, of how keen they are to find a counter-party for a transaction. Intuitively, there is a tradeoff between time-to-transact and the expected surplus (profit or saving) on that transaction. In human-populated markets, over time, while working a trade, an individual buyer will typically announce increasing bid prices, in the hope that the better prices increase the chance of finding a counterparty seller to transact with, but each of those increases in the bid-price reduces the surplus that the transaction will generate for that buyer; the situation is the same for sellers, gradually reducing their ask prices, again in the hope that each price-cut makes it more likely that a buyer will come forward, but each cut slices away at the seller's final profit for this transaction. In both cases, if the trader is urgent to get a deal done they can increase their chances of finding a counterparty by cutting

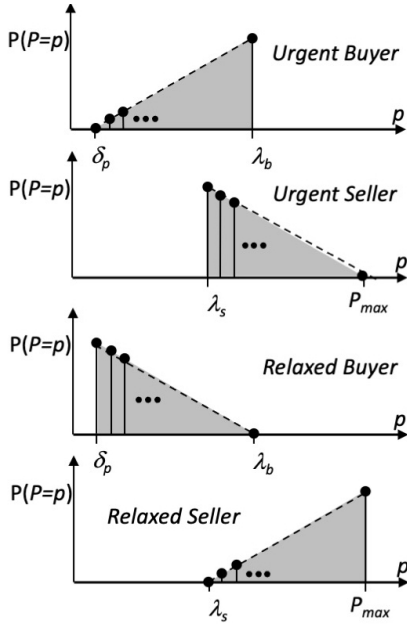


Fig. 2. Illustrative PMF for the variants of ZIC traders that are either *urgent* (upper two figures) or *relaxed* (lower two figures).

their potential surplus, i.e. by making bigger step-changes in the prices that they're quoting.

Lacking the intelligence of human traders, ZIC traders just issue their next quote price by drawing from a uniform distribution over a specified range of prices; for an individual ZIC trader, the change in price from one quote to the next may be positive or may be negative, and there is no control over the step-size. ZIC traders have no sense of urgency.

In contrast, a SHVR agent *can* be given some approximation to urgency: in [20], SHVR was extended by applying a multiplying coefficient $k \in \mathbb{Z}^+$ to the δ_p term in Equations 3 and 4, such that when the market circumstances dictate it, a value of $k > 1$ allows SHVR to make larger step-changes in its quote prices. Work on PRZI started as a response to asking: how can we do something similar for ZIC?

Figure 2 illustrates the reasoning underlying PRZI, when viewed as a way of making ZIC traders quote in a way that is more or less likely to lead to a transaction within any particular time-window: the rectangular PMF from the uniform distribution of the original ZIC can be replaced by a PMF that has either a right- or left-handed right-angled-triangle; depending on which way the triangle is oriented, the ZIC trader is either more or less likely to generate a quote-price that leads to a transaction. Let's refer to these two right-triangle PMFs as the *urgent* and *relaxed* variants of ZIC.

But why stop at these variants? Figure 3 illustrates two more extreme ZIC variants, where the PMF profile is nonlinear: let's refer to these as the *super-urgent* and *super-relaxed* variants of ZIC.

Clearly, the degree of nonlinearity in the variant-ZIC PMFs shown in Figure 3 could be made even more extreme, and eventually when the degree of curvature is at its most extreme

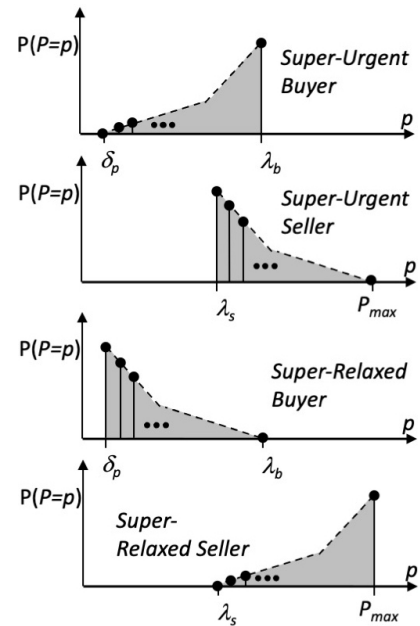


Fig. 3. Illustrative PMF for the variants of ZIC traders that are either *super-urgent* (upper two figures) or *super-relaxed* (lower two figures).

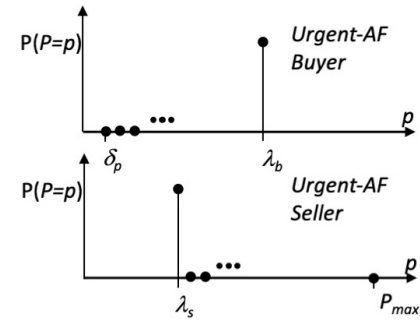


Fig. 4. Illustrative PMF for the *urgent-AF* variant of ZIC: a single price (the trader's limit-price) is quoted with probability one; the probability for all other prices is zero. This is identical to GVWY, as discussed in the text.

the PMF would have only one price at nonzero probability: the price that had the highest probability in the triangular PMF. And, as there is only one price with nonzero probability, that probability must be one (i.e., 100%, total certainty). Let's call these absolute extremes the *urgent-AF* and *relaxed-AF* ('AF' might stand for *asymptotic form*); they're illustrated in Figures 4 and 5.

But the shapes of the urgent-AF PMFs in Figures 4 and 5 are familiar: the urgent-AF PMFs are simply a probabilistic representation of GVWY, because equations 5 and 6 can be expressed in (redundantly) probabilistic language as: *with probability one, the price quoted by a GVWY trader is its current limit-price*.

This then prompts the question: can a useful link be made from the relaxed-AF form of ZIC to SHVR? The PMFs of SHVR and relaxed-AF ZIC have essentially the same shape, the difference is in where they occur on the number-line of

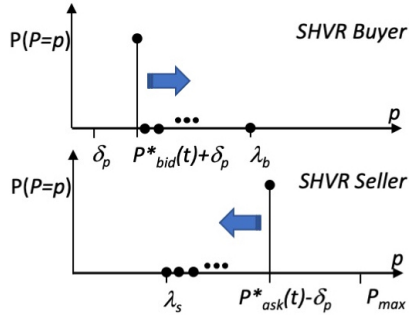


Fig. 5. Illustrative PMF for SHVR which can be considered as the equilibrating *relaxed-AF* variant of ZIC. The only price with a nonzero probability is a one-tick improvement on the best bid or ask at time t , and that price has a probability of one. The arrow pointing right (left) indicates the direction of travel of the lower (upper)-bound on the PMF domain; the upper (lower) bound is given by the trader's limit-price λ .

prices: SHVR as specified in Equations 3 and 4 quotes a price that is a one-tick (i.e. $1 \times \delta_p$) improvement on the current best price on the LOB; whereas a relaxed-AF ZIC would quote the most extreme price available, either the lowest nonzero price (i.e., δ_p) for a buyer, or the arbitrary system maximum P_{max} for a seller, neither of which is going to do anything at all to help equilibrate the market's transaction prices. Because the relaxed-AF versions of ZIC would not equilibrate, it makes most sense to move the relaxed-AF price away from the most extreme value, and toward the best price on the LOB, as the degree of nonlinearity in the variant-ZIC PMF increases. This would mean that by the time the nonlinearity is maximally extreme, i.e. when the PMF has the -AF shape, the variant ZIC is doing the same thing as a SHVR.

And that is the motivation for creating PRZI: now all that is needed is a function that smoothly varies from the urgent-AF variant that is GVWY, on through the nonlinear super-urgent variant, on through the triangular-PMF of the urgent variant, then on to the original ZIC, then onwards to the triangular-PMF of the relaxed variant, and then on through the super-relaxed nonlinear PMF to the relaxed-AF PMF that implements SHVR: this progression is controlled by PRZI's strategy parameter, denoted by $s \in [-1, +1] \in \mathcal{R}$. Recall that the trader's current *direction* (denoted by $d_i(t)$) is $+1$ for a seller and -1 for a buyer: when $d.s = -1$, PRZI implements GVWY; when $d.s = 0$, it implements ZIC; and when $d.s = +1$, SHVR. All of this is illustrated in Figure 6.

B. Details

The various seller PMFs shown in Figure 6 have a domain that is bounded from below (the left-hand limit of the PMF) by the individual seller's limit-price λ_s , but the upper bound on that domain, the right-hand limit of the PMF, brings us back to "the p_{max} problem" discussed previously in Section III-A. PRZI sellers address this problem by treating the range $s \leq 0$ and the range $s > 0$ in two different but related ways which will now be described in that order.

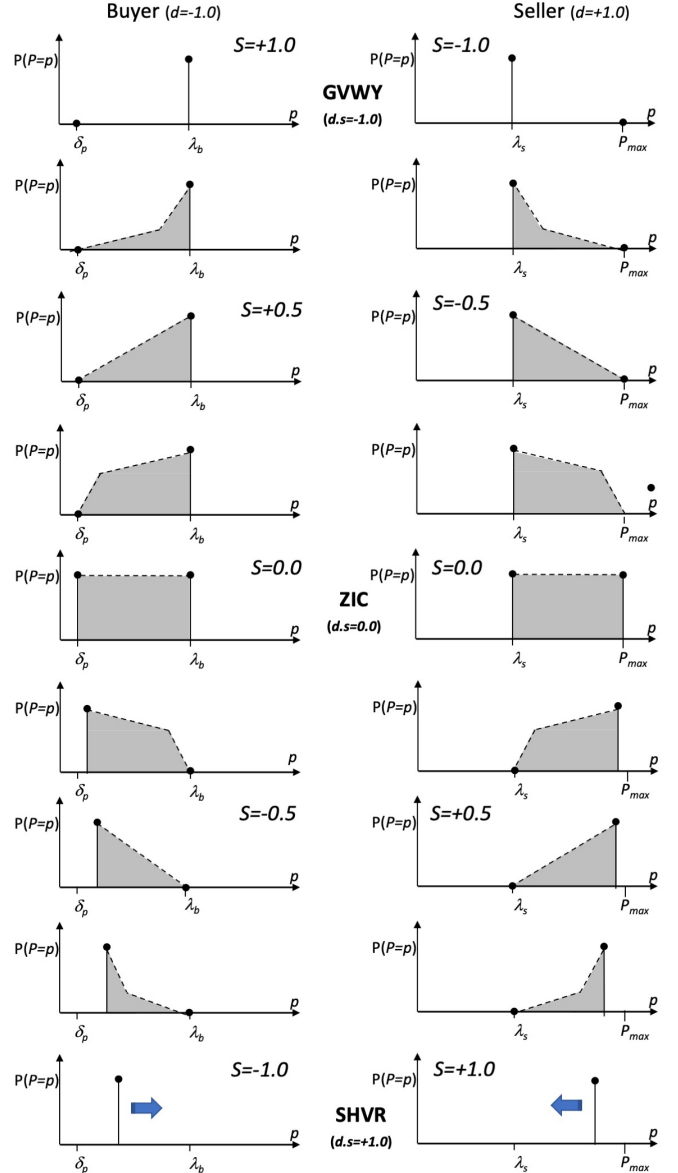


Fig. 6. The full spectrum of quote-price PMFs for PRZI: left-hand column is Buyer PMFs; right-hand column is Seller PMFs. Buyers have a *direction* value d of -1.0 ; sellers $+1.0$. Top row (where $d.s = -1.0$) is the *urgent-AF* PMF, equivalent to GVWY; then on each successive row the PMF envelope warps to become closer to the shape of the original ZIC PMF, which is in the middle row ($d.s = 0.0$); after that, each lower row warps closer to the *relaxed-AF* PMF that implements SHVR (at $d.s = +1.0$). The strategy-control parameters s is as shown: in this diagram seller strategies are arranged top-to-bottom from from $s = -1$ to $s = +1$, and buyer strategies are arranged from $s = +1$ to $s = -1$.

When $s \leq 0$ each PRZI seller i determines its own private estimate of the highest plausible price, denoted by $p_{i:\max}$, according to the following set of criteria:

- If the PRZI seller has no other information available to it (e.g., it is the start of the first session in a market experiment, and the LOB is empty) then the only available information it has is the highest limit price it has been assigned thus far, denoted by $\lambda_{s(i:\max)}$ (and if it has not been assigned a limit price, it is unable to participate in the market). In this situation, the PRZI seller sets $p_{i:\max} = c_i \lambda_{s(i:\max)}$ for some coefficient $c_i > 1$. Informally, this models a naive and uninformed seller making a wild guess at the highest price that the market might tolerate. Different sellers can have different values of c_i – that is, some might guess cautiously while others might be much more optimistic.
- If ever another seller issues an ask-quote at a price $P_{sq} > p_{i:\max}$ then seller i sets $p_{i:\max} = P_{sq}$. Informally, this models a seller realising that her current best guess of the highest price the market will tolerate was too low, because there is some other seller in the market who has quoted a higher price.

In principle, if reliable price information is available from an earlier market session, then that information could be used instead. However, in any one previous market session there are likely to be multiple potential candidate values (e.g: the highest ask-price quoted in that session, or the highest transaction-price in that session, or the final transaction price, or the linear-regression prediction of the next transaction price in the market, or a nonlinear prediction, and so on), and that would soon take us away from the appealing minimalism of ZI traders.

This approach, of letting each trader form its own private estimate of the highest tolerable ask-price when $s \geq 0$, could be criticised as merely swapping one arbitrary system parameter (the global constant $p_{\max}$) for a bunch of new arbitrary exogenously-imposed parameters (the set of individual c_i values, one per trader). If all we care about is minimising parameter-count, that would be a valid criticism. However, this approach has the advantage that only uses information that is locally available to an individual trader (i.e., it does not require knowledge of a system-wide constant) and it never requires manual re-calibration to the highest price in the market's supply curve. In practice setting the c_i values at random over a moderate range is sufficient to generate useful results – we have found $c_i = \mathcal{U}(1, 10)^{0.5}$ to be sufficient.

As described thus far, we have a method for setting the upper limit of the PRZI seller PMF when $s \leq 0$, i.e., for the PRZI range of strategies from ZIC to GVWY. However, for the $s = +1$ case that implements SHVR, we need the upper limit of the PMF to be set such that $p_{i:\max} = p_{\text{ask}}^* - \delta_p$, and we need to get there smoothly from the $s = 0$ case where PRZI is implementing ZIC and $p_{i:\max}$ is set by the method just described. The simplest way of doing that is to have $p_{i:\max}$ be a linear combination of the two. For a PRZI seller, let $p_{i:\max:\text{ZIC}}$

denote the $p_{i:\max}$ value at $s = 0$, then for $s > 0$ we use:

$$p_{i:\max} = (1 - s)P_{i:\max:\text{ZIC}} + s(p_{\text{ask}}^* - \delta_p). \quad (7)$$

and the same form of linear combination for PRZI buyers, but with inverted sign on s , to pull the lower limit on the buyer PMFs progressively away from δ_p and toward $p_{\text{bid}}^* + \delta_p$.

Next, let s_i denote the strategy-value for trader i ; and let $p_{i:\min} \in \mathbb{Z}^+$ and $p_{i:\max} \in \mathbb{Z}^+$ be the bounds on trader i 's discrete-valued price-range $[p_{i:\min}, p_{i:\max}]$, with $p_{i:\min} < p_{i:\max}$ and let $r_i = p_{i:\max} - p_{i:\min}$.

Then over the domain $p \in [0, 1] \in \mathbb{R}$, this function $\mathcal{P}$ has the right profile, makes the right shapes, for the PMFs that we want (as were illustrated in Figure 6):

$$\mathcal{P}(p, s_i) = \begin{cases} \frac{e^{cp} - 1}{e^c - 1} & \text{if } s_i > 0 \\ \frac{1}{r_i} & \text{if } s_i = 0 \\ 1 - \frac{e^{cp} - 1}{e^c - 1} & \text{if } s_i < 0 \end{cases} \quad (8)$$

where

$$c = \theta(m \tan(\pi(s_i + \frac{1}{2}))) \quad (9)$$

and $\theta(x)$ is the linear-rectifier threshold function symmetrically bounded by a cutoff constant θ_0 , which also (to avoid divide-by-zero errors) clips near-zero values at $\pm\epsilon$ for some sufficiently small value of ϵ (e.g. $\epsilon = 10^{-6}$):

$$\theta(x) = \begin{cases} \max(-\theta_0, \min(\theta_0, x)) & \text{if } |x| > \epsilon \\ \epsilon & \text{if } 0 < x < \epsilon \\ -\epsilon & \text{if } -\epsilon < x < 0 \end{cases} \quad (10)$$

After a little trial-and-error exploration, it was found that values for the constants $m = 4$ and $\theta_0 = 100$ give the desired shapes, i.e. similar to the qualitative PMF envelopes illustrated in Figure 6: these are illustrated in Figure 7. And, while $\mathcal{P}$ does generate curves of the desired shape, note that for it to do so when trader i is selling, $\mathcal{P}(p, s_i)$ should be used; but when i is buying then $\mathcal{P}(p, -s_i)$ should be used. Also, turning the $\mathcal{P}$ envelope into a usable PMF requires scaling and normalisation such that:

$$\mathbb{P}(P = p) = \frac{\mathcal{P}(\frac{p - p_{i:\min}}{r_i}, s_i)}{\sum_{j=0}^{r_i} \mathcal{P}(j/r_i, s_i)} : p \in [p_{i:\min}, p_{i:\max}] \quad (11)$$

From this we can then compute the cumulative distribution function (CDF) for trader i as $F_P(p) = P(P \leq p)$ which is defined over the domain $[p_{i:\min}, p_{i:\max}]$ as:

$$F_P(p) = \sum_{k=0}^{p - p_{i:\min}} \mathbb{P}(P = p_{i:\min} + k) \quad (12)$$

The $F_P(p)$ CDF in Equation 8 maps from its domain $[p_{i:\min}, p_{i:\max}]$ to cumulative probabilities in the interval $[0, 1] \in \mathbb{R}$: for each of the r_i discrete points in the domain, exact values of $F_P(p)$ can be computed and stored in a look-up table (LUT). It is then simple to use reverse-lookup on the LUT to give an inverse CDF function $\widehat{F_P^{-1}}(c)$ such that:

$$\widehat{F_P^{-1}}(c) : [0, 1] \in \mathbb{R} \mapsto [p_{i:\min}, p_{i:\max}] \in \mathbb{Z} \quad (13)$$

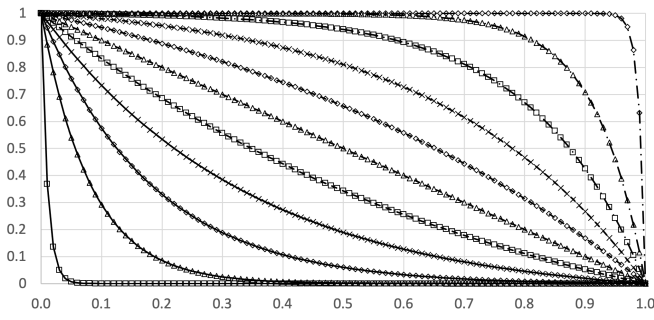


Fig. 7. Plots for $\mathcal{P}(p, s_i)$ (Equation 8) for values of s_i over the range $[0, 1]$ in steps of 0.1: horizontal axis is p in steps of 0.01; vertical axis is $\mathcal{P}$. The $\mathcal{P}$ curves have the desired shape, but require normalisation before use as PMF envelopes.

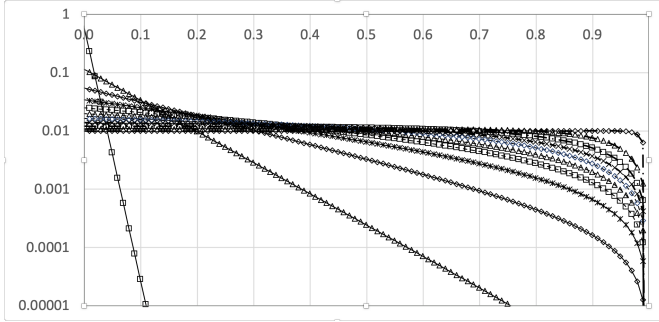


Fig. 8. Log-linear plots for $\mathbb{P}(P = p)$ (Equation 11) for values of s_i varying from 0 to 1 in steps of 0.1: horizontal axis is the interval $p \in [0, 1]$ in steps of 0.01; vertical axis is $\mathbb{P}$. The $\mathbb{P}$ curves are normalised to have a discrete definite integral over $p = [0, 1]$ equal to 1.0, and hence are valid as PMF envelopes.

Separate versions of $\widehat{F_P^{-1}}(c)$ need to be generated for buyers and sellers, which will be denoted by subscripted parenthetic s and b characters, and can then be fed samples from a uniformly distributed pseudo-random number generator to produce quote-prices for PRZI traders:

$$P_{bq}(\text{PRZI})(t + \Delta_t) = \widehat{F_{P(b)}^{-1}}(\mathcal{U}(0, 1)) \quad (14)$$

$$P_{sq}(\text{PRZI})(t + \Delta_t) = \widehat{F_{P(s)}^{-1}}(\mathcal{U}(0, 1)) \quad (15)$$

And note that in the special case when $s = 0$, $\widehat{F_P^{-1}}$ reduces to the identity function. That completes the definition of PRZI traders. The next section discusses implementation issues, and presents illustrative results.

V. PRZI IMPLEMENTATION

A reference implementation of PRZI written in *Python 3.8* has been added to the BSE sourcecode repository on GitHub [17]. For ease of intelligibility, the implementation follows the mathematics laid out in the previous section of this paper, computing an individual LUT for each trader: this approach is conceptually simple, but is manifestly inefficient in space and in time. To illustrate this, consider the case where all traders in the market have the same value s for their PRZI strategy parameter, and all sellers are assigned the same limit price λ_s

and all buyers the same λ_b : in such a situation, only two LUTs are needed: one to be shared among all buyers, and another to be shared among all sellers; but the current implementation wastes time and space by blindly computing an entire LUT for each trader. A more efficient implementation could be built around compiling any one LUT for a particular instance of an $(s, p_{\min}, p_{\max})$ triple only once, when it is first required, and storing it in some central shared key-value store or document database where the triple is the key and the LUT is the associated value or document. This would be considerably less inefficient, but would add considerably to the complexity of the code. As the Python code in BSE is intended to be simple, as an illustrative aid for non-expert programmers (and because I am lazy), the current BSE implementation does not use this approach. Further discussion of the BSE implementation of PRZI is given in the Appendix at the end of this paper.

VI. CONCLUSION

Having explained the motivation for designing PRZI, explained the maths of its internal mechanisms, described its implementation, and shown illustrative sample output from the reference implementation that is now freely available as Python source-code on GitHub, we are done.

ACKNOWLEDGMENT

I am grateful to Nik Alexandrov and Kamen Bachvarov for their comments on an early draft of this paper.

REFERENCES

- [1] J. Rust, J. Miller, and R. Palmer, “Behavior of trading automata in a computerized double auction market,” in *The Double Auction Market: Institutions, Theories, and Evidence*, D. Friedman and J. Rust, Eds. Addison-Wesley, 1992, pp. 155–198.
- [2] D. Gode and S. Sunder, “Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality,” *Journal of Political Economy*, vol. 101, no. 1, pp. 119–137, 1993.
- [3] D. Cliff, “Minimal-intelligence agents for bargaining behaviours in market-based environments,” HP Labs Technical Report, Tech. Rep. HPL-97-91, 1997.
- [4] S. Gjerstad and J. Dickhaut, “Price formation in double auctions,” *Games and Economic Behavior*, vol. 22, no. 1, pp. 1–29, 1998.
- [5] G. Tesauro and R. Das, “High-performance bidding agents for the continuous double auction,” in *Proceedings of the 3rd ACM Conference on Electronic Commerce*, 2001, pp. 206–209.
- [6] G. Tesauro and J. Bredin, “Sequential strategic bidding in auctions using dynamic programming,” in *Proceedings AAMAS 2002*, 2002.
- [7] K. Vytelingum, D. Cliff, and N. Jennings, “Strategic bidding in continuous double auctions,” *Artificial Intelligence*, vol. 172, no. 14, pp. 1700–1729, 2008.
- [8] R. Das, J. Hanson, J. Kephart, and G. Tesauro, “Agent-human interactions in the continuous double auction,” in *Proceedings IJCAI-2001*, 2001, pp. 1169–1176.
- [9] M. De Luca and D. Cliff, “Agent-human interactions in the continuous double auction, redux: Using the OpEx lab-in-a-box to explore ZIP and GDx,” in *Proceedings of the 2011 International Conference on Agents and Artificial Intelligence (ICAART2011)*, 2011.
- [10] —, “Human-agent auction interactions: Adaptive-Aggressive agents dominate,” in *Proceedings IJCAI-2011*, 2011, pp. 178–185.
- [11] M. De Luca, C. Szostek, J. Cartledge, and D. Cliff, “Studies of interaction between human traders and algorithmic trading systems,” UK Government Office for Science, London, Tech. Rep., Sep. 2011.
- [12] D. Vach, “Comparison of double auction bidding strategies for automated trading agents,” Master’s thesis, Charles University in Prague, 2015.

- [13] D. Cliff, “Exhaustive testing of trader-agents in realistically dynamic continuous double auction markets: AA does not dominate,” in *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, A. Rocha, L. Steels, and J. van den Herik, Eds. ScitePress, 2019, pp. 224–236.
- [14] D. Snashall and D. Cliff, “Adaptive-Aggressive traders don’t dominate,” in *Agents and Artificial Intelligence: Selected papers from ICAART2019*, J. van Herik, A. Rocha, and L. Steels, Eds. Springer, 2019.
- [15] M. Rollins and D. Cliff, “Which trading agent is best? using a threaded parallel simulation of a financial market changes the pecking-order,” in *Proceedings of the 32nd European Modeling and Simulation Symposium (EMSS2020)*, 2020.
- [16] D. Cliff and M. Rollins, “Methods matter: A trading algorithm with no intelligence routinely outperforms AI-based traders,” in *Proceedings of IEEE Symposium on Computational Intelligence in Financial Engineering (CIFE2020)*, 2020.
- [17] D. Cliff, *Bristol Stock Exchange: open-source financial exchange simulator*. <https://github.com/davecliff/BristolStockExchange>, 2012.
- [18] —, “BSE : A Minimal Simulation of a Limit-Order-Book Stock Exchange,” in *Proceedings of the 30th European Modeling and Simulation Symposium (EMSS2018)*, F. Bruzzone, Ed., 2018, pp. 194–203.
- [19] W. Walsh, R. Das, G. Tesauero, and J. Kephart, “Analyzing complex strategic interactions in multiagent systems,” in *Proceedings of the AAAI Workshop on Game-Theoretic and Decision-Theoretic Agents*, 2002.
- [20] G. Church and D. Cliff, “A simulator for studying automated block trading on a coupled dark/lit financial exchange with reputation tracking,” in *Proceedings of the 31st European Modelling and Simulation Symposium (EMSS2019)*, M. Affenzeller, A. Bruzzone, F. Longo, and G. Pereira, Eds., 2018, pp. 284–293.
- [21] Z. Zhang and D. Cliff, “Market impact in trader-agents: Adding multi-level order-flow imbalance-sensitivity to automated trading systems,” in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART2021)*, 2021.
- [22] R. Shiller, “Narrative Economics,” Cowles Foundation, Yale University, Tech. Rep. 2069, January 2017.
- [23] —, *Narrative Economics: How Stories Go Viral & Drive Major Economic Events*. Princeton University Press, 2019.
- [24] K. Lomas and D. Cliff, “Exploring narrative economics: An agent-based modeling platform that integrates automated traders with opinion dynamics,” in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART2021)*, 2021.
- [25] M. O’Hara, *Market Microstructure Theory*. Wiley, 1998.
- [26] L. Harris, *Trading and Exchanges: Market Microstructure for Practitioners*. Oxford University Press, 2002.
- [27] C.-A. Lehalle and S. Laruelle, *Market Microstructure In Practice (Second Edition)*. World Scientific, 2018.
- [28] J. Kagel and J. Roth, *The Handbook of Experimental Economics*. Princeton University Press, 1997.
- [29] V. Smith, Ed., *Bargaining and Market Behavior: Essays in Experimental Economics*. Cambridge University Press, 2000.
- [30] C. Plott and V. Smith, Eds., *Handbook of Experimental Economics Results, Volume 1*. North-Holland, 2008.
- [31] L. Tesfatsion and K. Judd, Eds., *Handbook of Computational Economics Vol.2: Agent-Based Computational Economics*. North-Holland, 2006.
- [32] S. H. Chen, “Varieties of agents in agent-based computational economics: A historical and an interdisciplinary perspective,” *Journal of Economic Dynamics and Control*, 2011.
- [33] —, *Agent-based computational economics: How the idea originated and where it is going*. Routledge, 2018.
- [34] Hommes, C. and LeBaron, B., Ed., *Computational Economics: Heterogeneous Agent Modeling*. North-Holland, 2018.
- [35] J. D. Farmer, P. Patelli, and I. Zovko, “The Predictive Power of Zero Intelligence in Financial Markets,” *Proceedings of the National Academy of Sciences*, vol. 102, no. 6, pp. 2254–2259, 2005.
- [36] D. Ladley, “Zero Intelligence in Economics and Finance,” *The Knowledge Engineering Review*, vol. 27, no. 2, pp. 273–286, 2012.
- [37] P. Vytelingum, “The structure and behavior of the continuous double auction,” Ph.D. dissertation, University of Southampton, 2006.
- [38] V. Smith, “An Experimental Study of Competitive Market Behaviour,” *Journal of Political Economy*, vol. 70, no. 2, pp. 111–137, 1962.
- [39] J. Osterrieder, *Arbitrage, Market Microstructure, and the Limit Order Book*. Suedwestdeutscher Verlag fuer Hochschulschriften, 2009.
- [40] I. Nolte, M. Salmon, and C. Adcock, Eds., *High Frequency Trading and Limit Order Book Dynamics*. Routledge, 2014.
- [41] F. Abergel, M. Anane, A. Chakraborti, A. Jedidi, and I. Toke, *Limit Order Books*. Cambridge University Press, 2016.

APPENDIX

Here, for reference, is a brief overview of the Python3.8 source-code for a PRZI trader within BSE [17]. BSE declares a generic Trader class, and a PRZI trader inherits various parameters from that in its `__init__` constructor function, which also creates each PRZI trader’s individual values of the strategy parameter $s_i(t)$, two LUTs for when it is buying ($d_i(t) = -1$) and selling ($d_i(t) = +1$), and the parameters m and θ_0 used in the calculation of $\mathcal{P}$. All BSE Trader objects have a `getorder()` function which returns the trader’s order (i.e. the quote it will send to the exchange) at time $t = \text{time}$, and which has access to the BSE LOB via its parameter `lob`. The parameter `countdown` tells the trader how long the current market session has left to run (this is used in SNPR, but not in SHVR or PRZI) and the parameter `verbose` controls how much narrative text the code writes to the standard output channel when executing. Figure 9 shows the `getorder()` function for the original SHVR, illustrating how the `lob` parameter is structured and accessed: the BSE LOB is divided into a bid-side and an ask-side, and each side stores its worst possible price in `worstp`: for the ask side of the LOB, `worstp` = $p_{\max}$; for the bid side `worstp` = δ_p . The bid-side `bestp` = $p_{\text{bid}}^*(t)$ and the ask-side `bestp` = $p_{\text{ask}}^*(t)$.

Figure 10 then shows the `getorder()` function for PRZI: this is very simple because, regardless of the value of s , it simply uses $d_i(t)$ (i.e., whether the current order is to be a bid or an ask) to select a LUT. The LUT is a Python list, i.e. an ordered sequence of LUT-entries. Each LUT entry is a two-element Python dictionary (i.e., a type of associative array, or key-value datastore), with entries for cumulative probability

```
# Trader subclass Shaver
# shaves one penny off the best price
class Trader_Shaver(Trader):

    def getorder(self, time, countdown, lob, verbose):

        if verbose:
            print("SHVR getorder:")

        if len(self.orders) < 1:
            order = None
        else:
            if verbose:
                print(" self.orders[0]=%s" % str(self.orders[0]))
            limitprice = self.orders[0].price
            otype = self.orders[0].atype
            ostyle = self.orders[0].astyle
            if otype == 'Bid':
                if lob['bids']['n'] > 0:
                    quoteprice = lob['bids']['bestp'] + 1
                    if quoteprice > limitprice:
                        quoteprice = limitprice
            else:
                quoteprice = lob['bids']['worstp']
        else:
            if lob['asks']['n'] > 0:
                quoteprice = lob['asks']['bestp'] - 1
                if quoteprice < limitprice:
                    quoteprice = limitprice
            else:
                quoteprice = lob['asks']['worstp']
        order = Order(self.lei, self.tid, otype, ostyle,
                    quoteprice, self.orders[0].qty, time, None, -1)
        self.lastquote = order
        return order
```

Fig. 9. Python3 source-code for SHVR’s `getorder()` function, illustrating how the LOB is accessed by a trader-agent in BSE.

```

def getorder(self, time, countdown, lob, verbose):
    # what are the extremes of the price interval?
    def price_extremes(lob):...

    # shvr_price tells us what price a SHVR would quote in these circs
    def shvr_price(otype, limit, lob):...

    # calculate cumulative distribution function (CDF) look-up table (LUT)
    def calc_cdf_lut(strat, t0, m, dirn, pmin, pmax):...

    if len(self.orders) < 1:
        # no orders: return NULL
        order = None
    else:
        # unpack the assignment-order
        limit = self.orders[0].price
        otype = self.orders[0].atype
        ostyle = self.orders[0].astyle

        # get extreme prices
        (minprice, maxprice) = price_extremes(lob)

        # what price would a SHVR quote?
        p_shvr = shvr_price(otype, limit, lob)

        # the LUTs are re-computed if any of the details have changed
        if otype == 'Bid':

            # direction * strat
            dxs = -1 * self.strat # the minus one multiplier is the "buy" direction

            p_max = limit
            if dxs <= 0:
                p_min = minprice # this is delta_p for BSE, i.e. ticksize =1
            else:
                # shade the lower bound on the interval
                # away from minprice and toward shvr_price
                p_min = int(0.5 + (dxs * p_shvr) + ((1.0-dxs) * minprice))

            if (self.cdf_lut_bid is None) or \
                (self.cdf_lut_bid['strat'] != self.strat) or \
                (self.cdf_lut_bid['pmin'] != p_min) or \
                (self.cdf_lut_bid['pmax'] != p_max):
                # need to compute a new LUT
                self.cdf_lut_bid = calc_cdf_lut(self.strat, self.theta0,
                                                self.m, -1, p_min, p_max)

            lut = self.cdf_lut_bid

        else: # otype == 'Ask'

            dxs = self.strat

            p_min = limit
            if dxs <= 0:
                p_max = maxprice
            else:
                # shade the upper bound on the interval
                # away from maxprice and toward shvr_price
                p_max = int(0.5 + (dxs * p_shvr) + ((1.0-dxs) * maxprice))

            if (self.cdf_lut_ask is None) or \
                (self.cdf_lut_ask['strat'] != self.strat) or \
                (self.cdf_lut_ask['pmin'] != p_min) or \
                (self.cdf_lut_ask['pmax'] != p_max):
                # need to compute a new LUT
                self.cdf_lut_ask = calc_cdf_lut(self.strat, self.theta0,
                                                self.m, +1, p_min, p_max)

            lut = self.cdf_lut_ask

        # do inverse lookup on the LUT to find the price
        u = random.random()
        for entry in lut['cdf_lut']:
            if u < entry['cum_prob']:
                quoteprice = entry['price']
                break

        order = Order(self.lei, self.tid, otype, ostyle,
                    quoteprice, self.orders[0].qty, time, None, -1)
        self.lastquote = order

    return order

```

Fig. 10. Python3 source-code for PRZI's `getorder()` function in BSE: the bulk of the code is taken up with deciding if a new CDF LUT needs to be computed or not; after that, it simply does reverse table-lookup on the relevant LUT.

(i.e., $F_P(p)$, defined in Equation 12), labelled with the key `cum_prob`; and price p , labelled with the key `price`. As would be expected from Equations 14 and 15, there are separate LUTs for buying and selling: the appropriate one is selected for lookup, and a new LUT is computed if any of $s, p_{\min:i}$, or $p_{\max:i}$ have changed since the current LUT was first calculated. The code for calculating the LUT is shown in Figure 11 which takes as parameters $s_i, \theta_0, m_i, d_i, p_{\min:i}$, and $p_{\max:i}$. The nested function `get_shvr_price()` within the PRZI `getorder()` is just a cut-and-paste of code shown in Figure 9. Inverse

```

# calculate cumulative distribution function (CDF) look-up table (LUT)
def calc_cdf_lut(strat, t0, m, dirn, pmin, pmax):
    # set parameter values and calculate CDF LUT
    # dirn is direction: -1 for buy, +1 for sell

    # the threshold function used to clip
    def threshold(theta0, x):
        t = max(-1*theta0, min(theta0, x))
        return t

    epsilon = 0.000001 #used to catch DIV0 errors

    if (strat > 1.0) or (strat < -1.0):
        # out of range
        sys.exit('FAIL: PRZI.getorder() self.strat out of range\n')

    if (dirn != 1.0) and (dirn != -1.0):
        # out of range
        sys.exit('FAIL: PRZI.calc_cdf() bad dirn\n')

    dxs = dirn * self.strat

    p_range = p_max - p_min
    if p_range < 1:
        sys.exit('Fail in Trader_PRZI.calc_cdf(): bad pmin pmax\n')

    c = threshold(t0, m * math.tan(math.pi * (strat + 0.5)))

    # catch div0 errors here
    if abs(c) < epsilon:
        if c > 0:
            c = epsilon
        else:
            c = -epsilon

    e2cm1 = math.exp(c) - 1

    # calculate the discrete calligraphic-P function over interval [pmin, pmax]
    # (i.e., this is Equation 8 in the PRZI Technical Note)
    calp_interval = []
    calp_sum = 0
    for p in range(pmin, pmax):
        p_r = (p - pmin) / p_range # p_r in [0.0, 1.0]
        if self.strat == 0.0:
            # special case: this is just ZIC
            cal_p = 1 / p_range
        elif self.strat > 0:
            cal_p = (math.exp(c * p_r) - 1.0) / e2cm1
        else: # self.strat < 0
            cal_p = 1.0 - ((math.exp(c * p_r) - 1.0) / e2cm1)
        if cal_p < 0:
            cal_p = 0 # just in case
        calp_interval.append({'price':p, 'cal_p':cal_p})
        calp_sum += cal_p

    cdf = []
    cum_prob = 0
    # now go thru interval summing and normalizing to give the CDF
    for p in range(pmin, pmax):
        price = calp_interval[p-pmin]['price']
        cal_p = calp_interval[p-pmin]['cal_p']
        prob = cal_p / calp_sum
        cum_prob += prob
        cdf.append({'price': p, 'cum_prob': cum_prob})

    return {'strat':strat, 'dirn':dirn, 'pmin':pmin, 'pmax':pmax, 'cdf_lut':cdf}

```

Fig. 11. Python3 source-code for PRZI's `calc_cdf_lut()` function in BSE: this calculates the $\widehat{F_P^{-1}}$ inverse-CDF LUT values.

table lookup in the PRZI `getorder()` is very simple: a value $u = \mathcal{U}(0, 1)$ is generated via the call to `random.random()`. The for loop then “walks” along the LUT until it encounters an entry with `cum_prob` value less than u , at which point the price value for that LUT entry gives the value p which the PRZI trader will use in this order.

Note that the code shown here has been written for clarity rather than for efficiency. Depending on how many discrete price-points there are in the LUT, it may be more efficient to detect the $s = 0$ (i.e., ZIC) special case and generate a quote-price for the next order directly, by calls to a uniform random number generator such as Python's `random.randint()`, rather than to compute a LUT for the uniform distribution over some price range. Similarly, when the strategy-parameter is set to its extreme values of $s = \pm 1$, it may be quicker to call the GVWY or SHVR functions directly rather than to retrieve the relevant value from the LUT. However, as soon as the value of $s \notin \{-1, 0, +1\}$, using the LUT is the path of least resistance.