A Mechanism for Adapting Theoretical Models of Trader Behavior to the Real-World

John Walsh

April 2025

Abstract

We adapt a class of theoretical models to a practical yet robust market environment that more closely resembles the real-world. We introduce a model for a market with many buyers and sellers each with a single valuation for an object drawn from a continous distribution and accommodate a model for bilateral trade between a single buyer and seller each with many possible discrete valuations for an object. The buyer and seller valuations are independent random variables unknown to the counterparty at the time of trade. We formalize a comprehensive algorithm capable of recreating the result that ex-post efficiency is generally impossible in a market where participants have private information. We later show that efficiency can be restored to the bilateral trade model through a market-maker who either maximizes payoffs to the buyer and seller, or to themselves.

1 Acknowledgements

Looking back on half a decade at Lehigh University, first as an undergraduate and now as a graduate student, I am forever grateful to the Lehigh community that welcomed me with open arms from the moment I first stepped on campus, and to the faculty who helped make my experience as transformative as possible. I would like to thank Professor Oleksandr Nikolsko-Rzhevskyy for being not only willing, but eager to accommodate my request to complete a thesis paper. Without hesitation he trusted me, allowing me to pursue my passion as a year long project. Finally, I would like to thank Professor James Dearden for all of his efforts over the past three years. It is because of him that this paper exists, but more importantly it is because of him that I pursued economics at all. My introduction to game theory came during his undergraduate course of the

same name, and his guidance during that semester gave me the confidence to strive for things, both academically and personally, that at one point I would not have felt were possible. Thank you for always pointing me in the right direction when I felt lost in my research, for always helping me to "build intuition", and for always seeing the potential in me even when I may not have seen it in myself.

2 Introduction

In any market, the behavior of buyers and sellers competing for ownership of an object is influenced not only by rules and regulations, but by human nature as well. Buyer's do not buy an object at any price higher than they value that object, and seller's never sell an object for any amount less than they value that object. This assumption can be taken even further when we assert that any trader may be reluctant to trade whenever they do not strictly profit (either monetarily or in the utility sense) from purchase or sale of an object. Thus, as proposed by Harsanyi (1967), the buyer or seller who knows only their valuation for an object at the time of trade, but who is unsure of their counterparty's valuation for the object, is tempted to bid or offer at a price lower or higher than their true value to maximize their payoff.

Markets are an ancient concept, but markets are also fundamental to modern life. Markets can take many forms. For example, an *auction* is characterized by many potential buyers bidding for a single object being offered by a single seller. In such a market environment, buyers have a dominant strategy to bid at a value no greater than their true valuation, and often bid lower to maximize their payoff when they win (Vickrey (1961)). But this is just one type of market. Consider now the more granular *call market*, once observed in the open outcry pits at the Chicago Board of Trade and the New York Stock Exchange. Such markets consist of many buyers and sellers each participating in multilateral trade for various objects (Wilson (1985)), and exhibit the convergence to price-taking behavior described in Williams (1991) and Satterthwaite and Williams (1989) as the number of market participants grows large. Finally, consider the many decentralized markets which arise in every day life. Informal (but nonetheless real) markets manifest themselves in such mundane environments as a yard-sale or an impromptu proposition from a stranger. In such cases, bilateral trade occurs between counterparties with sometimes large informational asynchronicities, which can result in trading prices that are much more favorable for the trader possessing an asymmetric information advantage (Chatterjee and Samuelson (1983) and Akerlof (1970)). Nonetheless, Riley and Samuelson (1981) showed that the expected results in a market with risk-neutral, profit maximizing traders are no different across market structures with similar characteristics.

Given the similarity of results across different markets, it should be possible to formulate a model to accurately describe the characteristics of a broad class of market environments. Earlier contributions to the study of game theory and mechanism design succeeded in formalizing models for understanding various market environments. However, a fundamental flaw plagues these mechanisms. These models assume that buyers and sellers draw their valuations for an object from a continuum of alternatives over a specified interval. This assumption does not hold however, under the taxonomy of real-world markets. Instead, the conventions of real-world trade require that

buyers and sellers value objects in discrete intervals, and in the event of trade a buyer makes a payment to a seller in discrete units of some commodity. It follows, that the main purpose of this paper will be to propose a method for bridging the gap between theory and reality, as it pertains to the set of possible valuations a buyer and seller can possess for an object. We formalize a complex algorithm, capable of performing the necessary transformations to fulfill the paper's main purpose. The remainder of this paper will proceed as follow. In Section 3 we present the theoretical model for the k-Double Auction from Rustichini et al. (1994). In particular, we outline the first-order-conditions for a buyer when selecting his equilibrium bid. In Section 4 we formalize a mechanism to generate a discrete set of realistic possible valuations for a buyer and a seller that are applicable to virtually any number of scenarios. Throughout the remainder of Section 4 we proceed to outline a mechanism capable of transforming the discrete valuation sets into an approximate continuous form, and conclude the section by replicating the results from Rustichini et al. (1994). In Section 5 we outline the theoretical model of a market-maker presented in Myerson and Satterthwaite (1983). Then, in Section 6 we define a mechanism capable of replicating the results from the paper, for the discrete values case. This paper will show that it is possible to recreate the results obtained by models with continuously distributed buyer and seller valuations, for the case of discrete values.

3 Theoretical Model of the k-Double Auction with Many Buyers and Sellers

Consider a market with m buyers and n sellers where $m, n \geqslant 2$ in which each market participant decides whether to trade 0 or 1 unit of a commodity. We follow the notation of Harsanyi (1967) and denote a buyer's value of 1 unit of the commodity as v_b and a seller's as v_s. Thus, a buyer's payoff is $v_b - p$ if he trades at price p and 0 if he does not. A seller's payoff is $p - v_s$ if she trades and 0 if she does not. We assume that buyer valuations are distributed on $[0, 1]$ according to the distribution function $F(\cdot)$, with a corresponding continuous density function $f(\cdot) > 0$. Valuations of the sellers are also distributed on $[0, 1]$ according to the distribution function $G(\cdot)$, with a corresponding continuous density function $g(\cdot) > 0$.

We define the k-Double Auction (henceforth, k-DA) mechanism according to the model in Chatterjee and Samuelson (1983), and later adapted by Rustichini et al. (1994). The buyers and the sellers submit bids and offers simultaneously. Let $B(\cdot)$ be a buyer's strategy function and let $S(\cdot)$ be a seller's strategy function. Let $\langle S(\cdot),\, B(\cdot)\rangle$ denote the strategy sets used by all buyers and sellers, where $\langle S(\cdot),\, B(\cdot)\rangle$ defines a symmetric equilibrium for an expected payoff maximizing buyer and seller when they assume that their opponent follows the strategy dictated by $B(\cdot)$ or $S(\cdot)$. A *strategy* $B(v_b)$ for a buyer, and $S(v_s)$ for a seller specifies a bid or offer as a function of v_b or v_s, respectively. To determine who trades sort the $m + n$ bids and offers as $s_{(1)} \leqslant s_{(2)} \leqslant \cdots \leqslant s_{(m+n)}$ where, for fixed $k \in [0, 1]$ we set $p = (1 - k)s_{(m)} + ks_{(m+1)}$. Because only the distribution of opponent valuations is known at the time of trade, a pair of strategies constitutes a *Bayesian Nash Equilibrium* in the trading game when each player's strategy maximizes their expected payoff conditional on expectations about their opponent's strategy.

Theorem 1. *For any equilibrium $\langle S(\cdot),\, B(\cdot)\rangle$: (Rustichini et al. (1994))*

(a) There exist values $\underline{v} < 1$, $\bar{c} > 0$ such that a seller with value v_s trades with positive probability if and only if $v_s < \bar{c}$, and a buyer with value v_b trades with positive probability if and only if $\underline{v} < v_b$;

(b) $S(\cdot)$ and $B(\cdot)$ are increasing over $[0, \bar{c})$ and $(\underline{v}, 1]$, respectively;

(c) $\lim_{v_b \downarrow \underline{v}} B(v) = \underline{v} = \lim_{v_s \downarrow 0} S(v_s)$, $\lim_{v_s \uparrow \bar{c}} S(v_s) = \bar{c} = \lim_{v_b \uparrow 1} B(v_b)$.

Point (a) defines the *serious* intervals for the buyers and sellers $(\underline{v}, 1]$ and $[0, \bar{c})$, which are the intervals over which a buyer with valuation v_b and seller with valuation v_s trade with positive probability. Because a seller with $v_s > \bar{c}$ never trades, she can costlessly submit erroneous offers. Similarly, a buyer with valuation $v_b < \underline{v}$ may bid erroneously, or even *a negative number* with no penalty. An implication of this logic is that misrepresentation by the buyers and sellers is *constrained* to the interval $[\underline{v}, \bar{c}]$ and cannot be bounded for bids in $[0, \underline{v}]$ or offers in $[\bar{c}, 1]$. It is only over serious bids and offers that equilibrium constrains $B(\cdot)$ and $S(\cdot)$. Point (b) implies that $S(\cdot)$ and $B(\cdot)$ are differentiable almost everywhere in $[0, \bar{c})$ and $(\underline{v}, 1]$. This allows for a model of convergence which employs the first-order conditions for the buyers. Finally, by tending $\underline{v}$ to the smallest serious offer and $\bar{c}$ to the largest serious bid, point (c) bounds the inefficiencies that non-serious bids/offers introduce into the market.

3.1 Main Results of the Theoretical Model

Following Rustichini et al. (1994) Consider an equilibrium $\langle S(\cdot), B(\cdot)\rangle$. Pick $v_b \in (\underline{v}, 1)$ and $v_s \in (0, \bar{c})$ such that $B'(v_b)$ and $S'(v_s)$ both exist and $B(v_b) = S(v_s) \equiv \lambda$. We formalize the first-order-conditions for a buyer as

$$\begin{aligned} 0 &= \frac{\partial \pi_b(v_b, \lambda)}{\partial \lambda} \\ &= (v_b - k)\left[nK_{n,m}(\lambda)\frac{f(v_s)}{S'(v_s)} + (m-1)L_{n,m}(\lambda)\frac{g(v_b)}{B'(v_b)} \right] \\ &\quad - kM_{n,m}(\lambda), \end{aligned} \tag{1}$$

where:

$\pi_b(v_b, k) \equiv$ A buyer's expected payoff when v_b is his value, λ is his bid, all sellers use $S(\cdot)$, and the other $m-1$ buyers use $B(\cdot)$

$K_{n,m}(\lambda) \equiv$ The probability that bid λ lies between $s_{(m-1)}$ and $s_{(m)}$ in a sample of $m-1$ buyers using strategy $B(\cdot)$ and $n-1$ sellers using $S(\cdot)$

$L_{n,m}(\lambda) \equiv$ The probability that bid λ lies between $s_{(m-1)}$ and $s_{(m)}$ in a sample of $m-2$ buyers using strategy $B(\cdot)$ and n sellers using $S(\cdot)$

$M_{n,m}(\lambda) \equiv$ The probability that bid λ lies between $s_{(m)}$ and $s_{(m+1)}$ in a sample of $m-1$ buyers using strategy $B(\cdot)$ and n sellers using $S(\cdot)$

A buyer's first-order-conditions equates his marginal expected gain from changing his bid with his marginal expected loss from receiving zero utility due to failure to trade at a price where he otherwise would have profited. For $v_b \in [\underline{v}, 1]$, the first-order conditions of a buyer may be invalid if either (i) $B(v_b)$ is outside of the range of $S(\cdot)$ or (ii) $S'(v_s)$ does not exist for the value of v_s that solves $S(v_s) = B(v_b)$. Nevertheless, as long as $B'(v_b)$ exists the inequality

$$(v_b - \lambda)(m-1)L_{n,m}(\lambda)\frac{g(v_b)}{B'(v_b)} - kM_{n,m}(\lambda) \leqslant 0 \tag{2}$$

holds because in equilibrium the marginal expected gain from passing another buyer (disregarding the possibility of passing a seller) as a result of raising one's bid cannot exceed the marginal expected loss from raising the bid.

4 Adaptation of the k-DA to a Single Buyer and Seller with Many Possible Valuations

The results from the previous section hold under fairly robust circumstances. The fundamental issue plaguing this analysis, however, is a disconnect between the model for traders with infinite possible valuations, and the relevance of this model to describing the real-world. The assumption of continuous distributions $F(\cdot)$ and $G(\cdot)$ from which buyers and sellers draw their valuations does not hold in real-world markets. In search of a better way to describe the behavior of real-world market participants, we construct a proprietary model for analyzing trade between a single buyer and a single seller, each with many possible discrete valuations for an object. In the real-world, trader's inherently value an object discretely, and in the event of trade, payments are made from the buyer to the seller in discrete intervals. Thus, adaptation of the model from Section 3 requires a mechanism capable of modeling strategic interaction between a buyer and seller across a broad range of possible scenarios. We formalize an intuitive algorithm capable of achieving results consistent with the earlier continuous model. We begin by discussing the data generation process for buyer and seller valuations and serious trading intervals. We then discuss the process used to transform a discrete valuation grid into continuous distributions. Finally, we outline the model for finding and verifying equilibrium trading conditions in the k-DA.

4.1 The Model, and Data Generating Process

Our model accommodates bilateral trade between a single buyer and seller, each having many possible discrete valuations. We take v_b to be the buyer's valuation and v_s to be the seller's valuation. We index the number of possible valuations for the buyer and the seller by $N = \{5, 6, \ldots, 1000\}$, and define $\mathcal{V}$ as the *discrete valuation grid*, which specifies an identical set of

uniformly distributed possible valuations for the buyer and seller on $[0, 1]$.[1]

Definition 1 (Discrete Valuation Grid)**.** *Let* $5 \leqslant N \leqslant 1000$ *be an integer. Then,*

$$\mathcal{V} \;=\; \left\{ \frac{i}{N-1} : i = 0, 1, \ldots, N-1 \right\} \tag{3}$$

Therefore, we denote $v_b,\, v_s \in \mathcal{V}$.[2] Restricting the number of possible valuations to $5 \leqslant N \leqslant 1000$ ensures that, in any case, there is sufficient data to generate informative conclusions, while limiting excessively long run-times when $\mathcal{V}$ grows large. In Section 3 continuous prices were implied, so here we denote the price of the object as p, with continuous distribution $P(\cdot) \in [0, 1]$ and density $\rho(\cdot) > 0$. Define k as the mechanism parameter, which is a measure of the influence that either the buyer or seller has on the final trading price in the k-DA. We restrict $k \in [0, .001, .002, \ldots, 1]$, where each different assignment of k comprises a separate mechanism in the k-DA. In the model our default values are $N = 5$ and $k = 0.5$, but we allow the researcher to select any integer N in the interval, and any real $0 \leqslant k \leqslant 1$ to three decimal places of precision.[3,4] This optionality allows the researcher to explore how both the discrete valuation grid $\mathcal{V}$, and the mechanism parameter k, affect equilibrium and trading-range outcomes.

We formalize another mechanism to recreate the serious trading range from Rustichini et al. (1994). Define the lower-bound on values where the buyer trades with positive probability as $\underline{v}$.

Definition 2 (Eligible Lower Bounds, $\underline{v}$)**.** *Denote* $\mathcal{E}$ *as the set of eligible valuations for* $\underline{v}$ *by*

$$\mathcal{E} \;=\; \{v_b \in \mathcal{V} : 0.1 \leqslant v_b \leqslant 0.25\}.$$

Then draw $\underline{v}$ *at random from* $\mathcal{E}$ *where each draw of* $\underline{v}$ *represents a different mechanism in the k-DA.*

The interval that defines the set of possible buyer valuations ($\mathcal{E}$), from which $\underline{v}$ is drawn, approximates a mean-preserving spread with stochastic uncertainty of .173.[5] Thus, define $v_{b,\text{ser}} \in [\underline{v}, 1]$ as any possible value for which the buyer trades with positive probability. Similarly, define $\bar{c} = 1 - \underline{v}$ as the symmetric upper-bound on the serious trading interval and any possible value $v_{s,\text{ser}} \in [0, \bar{c}]$ for which the seller trades with positive probability.[6] Denote any possible value for which the buyer and seller both trade with positive probability as $v_{\text{ser}} \in [\underline{v}, \bar{c}]$.[7] Therefore, $v_{b,\text{ser}},\, v_{s,\text{ser}},\, v_{\text{ser}} \in \mathcal{V}_{\text{ser}} \subset \mathcal{V}$. Borrowing from Rustichini et al. (1994), let $B(\cdot)$ be a buyer's strategy function and let $S(\cdot)$ be a seller's strategy function. A buyer's bid given their valuation v_b is thus $B(v_b)$, and a seller's offer given their valuation v_s is $S(v_s)$.

[1] Symmetric discrete valuation sets are chosen because they preserve strategic equivalence between buyers and sellers in bilateral trade, a key feature of the Rustichini et al. (1994) model.

[2] Reference Appendix A Figure 2(a) for the formal algorithm for parameter generation

[3] For example, if $N = 5$ then $v_b, v_s \in \mathcal{V} = \{0.0, 0.25, 0.50, 0.75, 1\}$.

[4] For the remainder of this paper all figures will display outputs for the case of $N = 30$ and $k = 0.5$

[5] .173 is chosen as the arbitrary lower-bound of the trading interval in Rustichini et al. (1994)

[6] Reference Appendix A Figure 3 for the formal algorithm to generate $\underline{v}$ and $\bar{c}$

[7] Henceforth, any valuation for the buyer and the seller within their respective serious intervals will be denoted by the subscript "ser".

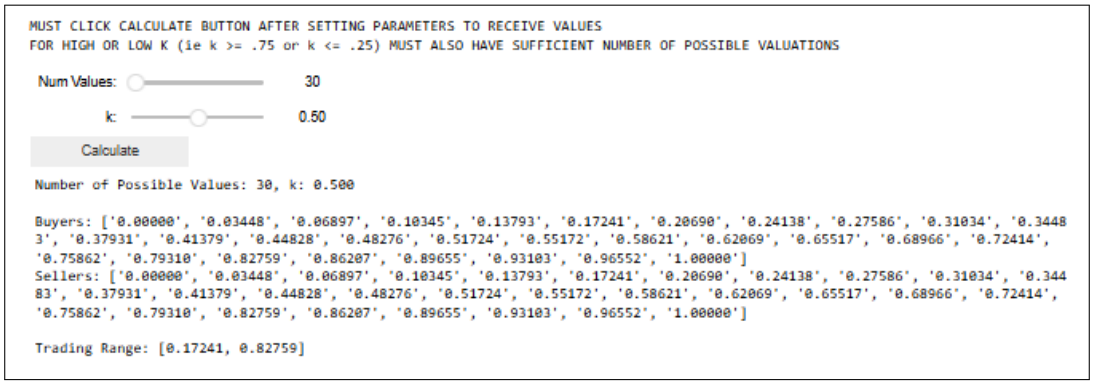


Figure 1: Parameter Selection Mechanism

Remark: Figure 1 shows the mechanism used to select the parameters N and k. After employing the mechanism from Equation 3, Figure 1 then showcases the number of possible values N. The mechanism also shows the set of possible valuations in the discrete valuation grid $\mathcal{V}$ for both the buyer and the seller, as well as the serious trading interval $[\underline{v}, \bar{c}]$.

Definition 3 (Eligible Bids and Offers)**.** *For any valuation $v_b, v_s \in \mathcal{V}$:*

$$B(v_b) = \begin{cases} \text{“No Serious Bid”}, & v_b < \underline{v}, \\ [\underline{v},\, v_b\,], & \underline{v} \leqslant v_b \leqslant 1, \end{cases}$$

$$S(v_s) = \begin{cases} \text{“No Serious Offer”}, & v_s > \bar{c}, \\ [\,v_s,\, \bar{c}\,], & 0 \leqslant v_s \leqslant \bar{c}. \end{cases}$$

In practice, we print an aligned table that lists the intervals over which the buyer and seller with a valuation v_b, v_s, $v_{b,\text{ser}}$, $v_{s,\text{ser}}$, v_{ser}, submit eligible bids and offers $B(v_b)$ and $S(v_s)$ respectively.[8]

[8]Reference Appendix A Figure 4 for the formal algorithm used to generate serious trading intervals

```
======================================================
============== SERIOUS TRADING INTERVALS ==============
======================================================
Buyer Serious Interval: [0.17241, 1.00000]
Seller Serious Interval: [0.00000, 0.82759]
Trading Overlap: [0.17241, 0.82759]
→ 65.5% of price continuum
```

```
======================================================
================ BUYER BIDDING RANGES ================
======================================================
Buyer 0.00000 | not in serious bids (below threshold)
Buyer 0.03448 | not in serious bids (below threshold)
Buyer 0.06897 | not in serious bids (below threshold)
Buyer 0.10345 | not in serious bids (below threshold)
Buyer 0.13793 | not in serious bids (below threshold)
Buyer 0.17241 | [0.17241] (Only Possible Bid)
Buyer 0.20690 | Bids: [0.17241, 0.20690]
Buyer 0.24138 | Bids: [0.17241, 0.24138]
Buyer 0.27586 | Bids: [0.17241, 0.27586]
Buyer 0.31034 | Bids: [0.17241, 0.31034]
Buyer 0.34483 | Bids: [0.17241, 0.34483]
Buyer 0.37931 | Bids: [0.17241, 0.37931]
Buyer 0.41379 | Bids: [0.17241, 0.41379]
Buyer 0.44828 | Bids: [0.17241, 0.44828]
Buyer 0.48276 | Bids: [0.17241, 0.48276]
Buyer 0.51724 | Bids: [0.17241, 0.51724]
Buyer 0.55172 | Bids: [0.17241, 0.55172]
Buyer 0.58621 | Bids: [0.17241, 0.58621]
Buyer 0.62069 | Bids: [0.17241, 0.62069]
Buyer 0.65517 | Bids: [0.17241, 0.65517]
Buyer 0.68966 | Bids: [0.17241, 0.68966]
Buyer 0.72414 | Bids: [0.17241, 0.72414]
Buyer 0.75862 | Bids: [0.17241, 0.75862]
Buyer 0.79310 | Bids: [0.17241, 0.79310]
Buyer 0.82759 | Bids: [0.17241, 0.82759]
Buyer 0.86207 | Bids: [0.17241, 0.86207]
Buyer 0.89655 | Bids: [0.17241, 0.89655]
Buyer 0.93103 | Bids: [0.17241, 0.93103]
Buyer 0.96552 | Bids: [0.17241, 0.96552]
Buyer 1.00000 | Bids: [0.17241, 1.00000]
```

(a) Eligible Bids $B(v_b)$, for Values v_b

```
======================================================
================ SELLER OFFERING RANGES ==============
======================================================
Seller 0.00000 | Offers: [0.00000, 0.82759]
Seller 0.03448 | Offers: [0.03448, 0.82759]
Seller 0.06897 | Offers: [0.06897, 0.82759]
Seller 0.10345 | Offers: [0.10345, 0.82759]
Seller 0.13793 | Offers: [0.13793, 0.82759]
Seller 0.17241 | Offers: [0.17241, 0.82759]
Seller 0.20690 | Offers: [0.20690, 0.82759]
Seller 0.24138 | Offers: [0.24138, 0.82759]
Seller 0.27586 | Offers: [0.27586, 0.82759]
Seller 0.31034 | Offers: [0.31034, 0.82759]
Seller 0.34483 | Offers: [0.34483, 0.82759]
Seller 0.37931 | Offers: [0.37931, 0.82759]
Seller 0.41379 | Offers: [0.41379, 0.82759]
Seller 0.44828 | Offers: [0.44828, 0.82759]
Seller 0.48276 | Offers: [0.48276, 0.82759]
Seller 0.51724 | Offers: [0.51724, 0.82759]
Seller 0.55172 | Offers: [0.55172, 0.82759]
Seller 0.58621 | Offers: [0.58621, 0.82759]
Seller 0.62069 | Offers: [0.62069, 0.82759]
Seller 0.65517 | Offers: [0.65517, 0.82759]
Seller 0.68966 | Offers: [0.68966, 0.82759]
Seller 0.72414 | Offers: [0.72414, 0.82759]
Seller 0.75862 | Offers: [0.75862, 0.82759]
Seller 0.79310 | Offers: [0.79310, 0.82759]
Seller 0.82759 | [0.82759] (Only Possible Offer)
Seller 0.86207 | not in serious offers (above threshold)
Seller 0.89655 | not in serious offers (above threshold)
Seller 0.93103 | not in serious offers (above threshold)
Seller 0.96552 | not in serious offers (above threshold)
Seller 1.00000 | not in serious offers (above threshold)
```

(b) Eligible Offers $S(v_s)$, for Values v_s

Figure 2: Eligible Bids and Offers, $B(v_b)$ and $S(v_s)$, for Values v_b and v_s

Remark: Figure 2 showcases the intervals over which the buyer and seller submit eligible bids and offers for each of their possible valuations. Because a buyer never bids at $B(v_b) > v_b$ and a seller never offers at $S(v_s) < v_s$, we can restrict the bidding interval to $[0, v_b]$ for all v_b and $[v_s, 1]$ for all v_s, respectively. Because a buyer and seller never bid or offer at a value outside the their respective serious intervals $[\underline{v}, v_b]$ for the buyer and $[v_s, \bar{c}]$ for the seller, we can denote these values as "not in serious bids" or "not in serious offers". Figure 2 also displays the percentage of valuations for the buyer and seller that overlap in the serious trading interval $[\underline{v}, \bar{c}]$.

4.2 Adaptation of Discrete v_b and v_s to Continuous Distributions

The discrete nature of $v_b, v_s \in \mathcal{V}$ still poses analytical challenges when constructing strategy functions for the buyer and seller. Specifically, the cumulative distribution functions $F(\cdot)$ and $G(\cdot)$, and probability density functions $f(\cdot)$, $g(\cdot) > 0$, manifest as "step" functions with n discrete steps over $[0, 1]$ according to Equation 3. Let $\ell = \underline{v}$ and $u = \bar{c}$ denote bounds in *any* serious trading interval where the buyer and/or seller trade with positive probability. We will describe two methods for transforming discrete valuation sets into continuous distributions that allow for differentiation. These methods produce *smoothed* distributions via kernel density methods and a process called

jittering. We define the kernel density method as an *adaptive bandwidth adjustment* which scales a sample standard deviation relative to the width of the interval, and we define jittering as a perturbation of the set of all interior values contained within the interval. The following definition is taken from Silverman (1986).

Definition 4 (Silverman's Rule of Thumb)**.** *For a Gaussian kernel density estimator applied to a dataset $X = \{x_1, \ldots, x_n\}$, the bandwidth h minimizing Asymptotic Mean Integrated Squared Error (AMISE) under normality is:*

$$h_{Silverman} = 0.9 \cdot \min\left(\hat{\sigma},\ \frac{\mathrm{IQR}(X)}{1.34}\right) \cdot n^{-1/5},$$

where $\hat{\sigma}$ is the sample standard deviation and $\mathrm{IQR}(X)$ is the inter-quartile range.

Adapting Silverman's Rule, we omit the IQR, as bounded data renders it unstable. For bounded support $[\ell, u]$ the IQR becomes degenerate when variation clusters near ℓ or u. This violates the normality assumption, making IQR bandwidths unreliable.

Method 1 (Adaptive Bandwidth for Bounded Support)**.** *Let $\mathcal{T}$ be any valuation sample in $[\ell, u]$ with positive probability of trade. Then, $\mathcal{T} \subseteq [\ell, u]$. When $[\ell, u] = [0, 1]$, $\mathcal{T} = \mathcal{V}$. Denote by $\hat{\sigma}$ the sample standard deviation of $\mathcal{T}$, and let $W = u - \ell$ be the interval width. The adaptive bandwidth is then given by:*

$$h = 1.06 \cdot \hat{\sigma} \cdot |\mathcal{T}|^{-0.2} \cdot \underbrace{\left(1 + \tfrac{\hat{\sigma}^2}{W^2}\right)^{0.2}}_{\textit{boundary factor}}, \tag{4}$$

which adjusts for boundary bias. To avoid over-smoothing, the bandwidth is constrained to:

$$h \in [0.05W,\ 0.5W].$$

We include a boundary factor $\left(1 + \frac{\hat{\sigma}^2}{W^2}\right)^{0.2}$ to explicitly address edge biases without requiring kernel reflection, which reduces runtime and complexity.[9] This adapts Silverman's rule by inflating h when variance $\hat{\sigma}^2$ is large relative to W.

Before applying our kernel density estimation from Equation 4, we randomly perturb (jitter) each interior valuation in the sample $\{v_b, v_s\} \in (0, 1)$ by appending $\epsilon \overset{\text{u}}{\sim} (-\delta, \delta)$ with $(\delta = 0.005)$, and leave the endpoints $\{0, 1\}$ unperturbed.[10] This prevents the kernel from over-weighting mass at 0 or 1 to produce a smoother, stabler density estimate. Below is the process used for jittering buyer valuations.[11,12] We refer to any value that has undergone bandwidth adjustment and jittering as a *processed* value.

[9] Reference Appendix B Figure 1 for the formal algorithm used to perform the adapted bandwidth adjustment

[10] Reference Appendix B Figure 2 for the formal algorithm used to apply boundary jittering

[11] We withhold a model of the jittering process for seller values on the interior of a given interval because it is analogous to that of the buyer.

[12] Superscript "proc" is short for processed, and denotes that the value has been processed.

Method 2 (Boundary Jittering). *For each valuation $v_b \in \mathcal{V}$, denote*

$$v_b^{\text{proc}} = \begin{cases} v_b, & \text{if } v_b \in \{0,1\}, \\ \text{clip}(v_b + \epsilon, 0, 1), & \text{otherwise.} \end{cases} \qquad \epsilon \overset{\text{u}}{\sim} (-\delta, \delta).$$

This prevents discretization artifacts at the boundaries in the subsequent KDE.

Our processed versions of buyer and seller values are thus: v_b^{proc}, v_s^{proc}, $v_{b,\text{ser}}^{\text{proc}}$, $v_{s,\text{ser}}^{\text{proc}}$, $v_{\text{ser}}^{\text{proc}}$.[13]

```
=== PROCESSED VALUE SETS ===

Processed Buyer Values (vb_processed) (30 items):
[0.00000, 0.03728, 0.07085, 0.10324, 0.13583, 0.16754, 0.20302, 0.24611, 0.27781, 0.31520, 0.34500, 0.37834, 0.41443, 0.44433,
0.48602, 0.51867, 0.55309, 0.58393, 0.62289, 0.65841, 0.69046, 0.72167, 0.75631, 0.79027, 0.82954, 0.86672, 0.90021, 0.93538,
0.96094, 1.00000]

Processed Seller Values (vs_processed) (30 items):
[0.00000, 0.03735, 0.07035, 0.10148, 0.13706, 0.16873, 0.20420, 0.24373, 0.27966, 0.30956, 0.34674, 0.37667, 0.41695, 0.44612,
0.48478, 0.51883, 0.55365, 0.58486, 0.61876, 0.65913, 0.69032, 0.72408, 0.75619, 0.79288, 0.82608, 0.86575, 0.89367, 0.93461,
0.96297, 1.00000]

Processed Serious Buyer Values (vb_serious_processed) (20 items):
[0.17129, 0.20253, 0.23887, 0.27416, 0.31157, 0.34184, 0.37795, 0.41253, 0.44930, 0.48100, 0.51706, 0.55657, 0.58635, 0.62562,
0.65481, 0.69004, 0.71994, 0.75606, 0.79370, 0.82922]

Processed Serious Seller Values (vs_serious_processed) (20 items):
[0.17682, 0.20396, 0.24357, 0.27180, 0.31131, 0.34869, 0.37852, 0.40967, 0.44634, 0.48030, 0.51653, 0.55381, 0.58759, 0.62431,
0.65837, 0.68646, 0.72037, 0.76108, 0.79245, 0.82712]

Processed Combined Serious Values (serious_processed) (20 items):
[0.17104, 0.20368, 0.24237, 0.27411, 0.31087, 0.34219, 0.37520, 0.41405, 0.44932, 0.48524, 0.52013, 0.55147, 0.58880, 0.62320,
0.65213, 0.68506, 0.72447, 0.75878, 0.78813, 0.82811]
```

Figure 3: Processed Values after "Jittering" and Bandwidth Adjustments.

Remark: Figure 3 displays all possible processed valuations in $\mathcal{V}^{\text{proc}}$ for the buyer and the seller following the application of Methods 1 and 2 to all of their respective original possible values v_b and v_s. The processed valuation sets are displayed in order of v_b^{proc}, v_s^{proc}, $v_{b,\text{ser}}^{\text{proc}}$, $v_{s,\text{ser}}^{\text{proc}}$, $v_{\text{ser}}^{\text{proc}}$.

The five samples are then fed through the same KDE pipeline. After computing the adaptive bandwidth h in Method 1, we form the raw kernel estimate and then re-normalize, so that on the finite support $[\ell, u]$ the PDF integrates to one and the CDF spans 0 to 1.[14]

Definition 5 (Empirical PDF and CDF). *Let $\mathcal{T} \subseteq [\ell, u]$ be any serious valuation set. Then define for any arbitrary value v, the empirical probability mass function*

$$f_{\text{emp}}(v) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{v\}, \quad v \in \mathcal{T},$$

then, the empirical cumulative distribution function is,

$$F_{\text{emp}}(v) = \sum_{u, \in \mathcal{T} : u \leqslant v} f_{\text{emp}}(u).$$

[13] We henceforth append superscript "proc" to any possible valuation, because in our algorithm any subsequent process utilizes only processed values.

[14] Reference Appendix B Figure 3(a) for the formal algorithm used to compute the empirical PDF and CDF, Reference Appendix B Figure 3(b) for the formal algorithm used to compute the smoothed PDF and CDF via bounded KDE.

Definition 6 (Bounded KDE with Normalization)**.** *Let $\mathcal{T}^{\text{proc}} \subseteq [\ell, u]$ be any processed serious valuation set, and let h be the adaptive bandwidth from Method 1. Then define for any arbitrary processed value v^{proc}, the (non-normalized) Gaussian kernel estimate*

$$\hat{f}(x) = \frac{1}{n\,h} \sum_{i=1}^{n} K\Big(\frac{x - v^{\text{proc}}}{h}\Big), \quad K(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}, \;\; x \in [\ell, u]. \tag{5}$$

Then compute the normalization constant

$$Z \;=\; \int_{\ell}^{u} \hat{f}(t)\,\mathrm{d}t,$$

and finally obtain the proper PDF/CDF pair

$$f_{\text{kde}}(x) = \frac{\hat{f}(x)}{Z} \text{ and } F_{\text{kde}}(x) = \int_{\ell}^{x} f_{\text{kde}}(t)\,\mathrm{d}t.$$

After formalizing the adaptive bandwidth adjustment and jittering processes in Methods 1 and 2, as well as $f_{\text{kde}}(\cdot)$ and $F_{\text{kde}}(\cdot)$ from Definition 6, we can now compute continuous distributions for buyer and seller valuations.

Method 3 (Smoothed PDF and CDF via Bounded KDE)**.** *Using the same jittered sample $\mathcal{T}^{\text{proc}}$ and bandwidth h as in Definition 6, define for any arbitrary processed value v^{proc}*

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\Big(\frac{x - v^{\text{proc}}}{h}\Big), \quad x \in [\ell, u], \tag{6}$$

and normalize to obtain

$$f_{\text{kde}}(x) = \frac{\hat{f}(x)}{\displaystyle\int_{\ell}^{u} \hat{f}(t)\,\mathrm{d}t}, \tag{7}$$

$$F_{\text{kde}}(x) = \int_{\ell}^{x} f_{\text{kde}}(t)\,\mathrm{d}t. \tag{8}$$

Remark: A larger h smooths out discontinuities but can over-smooth strategic thresholds when $\mathcal{V}$ grows large. The boundary factor in Method 1 rescales h relative to the support width $W = u - \ell$, preserving differentiability without incurring excessive boundary bias. Overall, the marginal efficiency losses at large $\mathcal{V}^{\text{proc}}$ are outweighed by the analytical flexibility of a differentiable density.

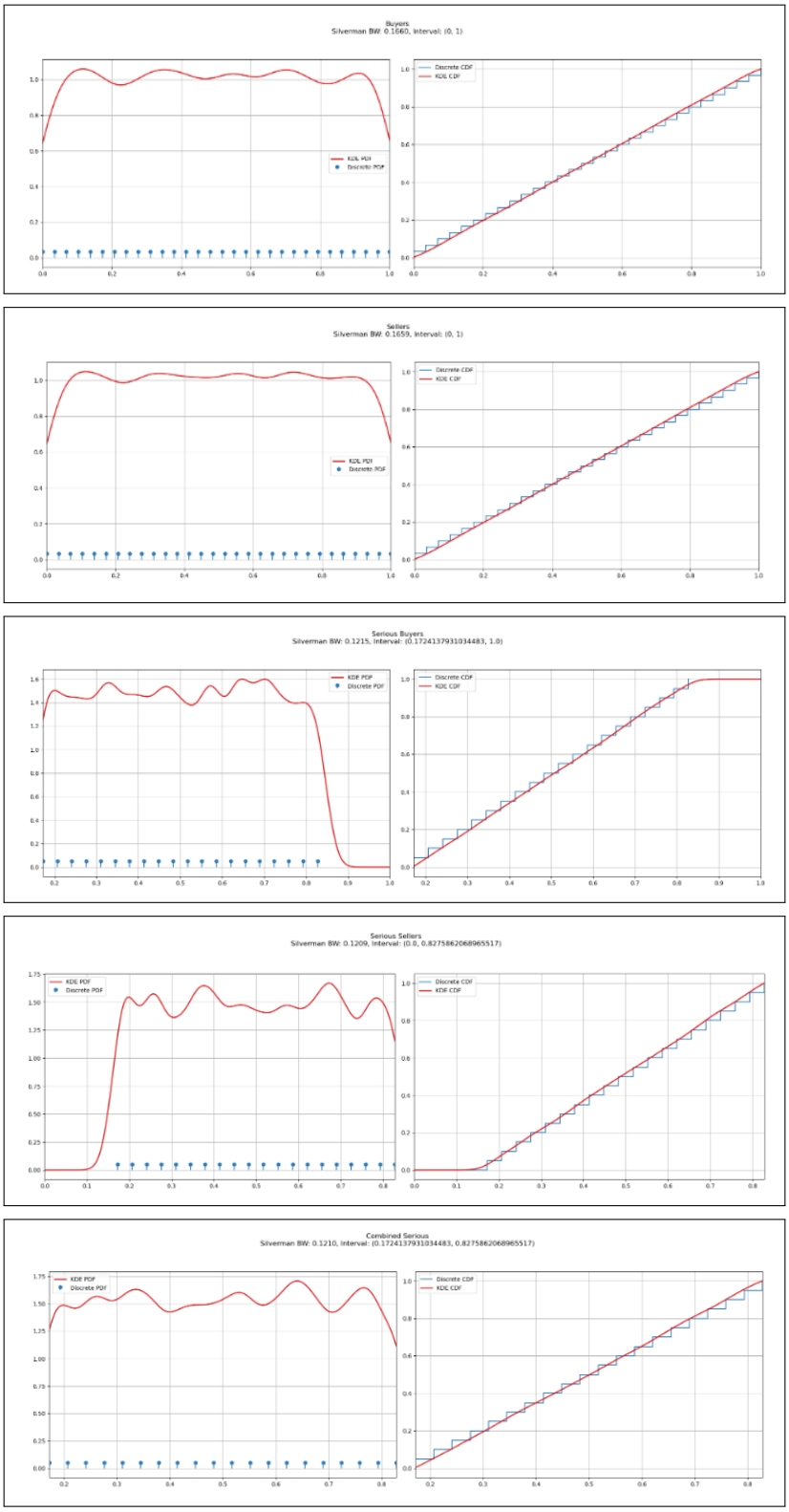


Figure 4: Processed CDFs and PDFs for v_b^{proc}, v_s^{proc}, $v_{b,\text{ser}}^{\text{proc}}$, $v_{s,\text{ser}}^{\text{proc}}$, $v_{\text{ser}}^{\text{proc}}$.

Remark: Figure 4 displays the PDFs and CDFs for all possible v_b^{proc}, v_s^{proc}, $v_{b,\text{ser}}^{\text{proc}}$, $v_{s,\text{ser}}^{\text{proc}}$, $v_{\text{ser}}^{\text{proc}}$ for the buyer and seller. The PDF and CDF graphs appear in the order that the possible value alternatives are listed above. For each case, the left-side grid displays the discrete PDF in blue

and the processed, continuous PDF in red. For each case, the right-side grid display the discrete CDF in blue and overlays the processed, continuous CDF in red.

Once we have a smoothed density over any processed sample $\mathcal{T}^{\text{proc}} \subseteq \mathcal{V}^{\text{proc}}$, we know by construction that the distributions $f_{\text{kde}}(\cdot)$ and $F_{\text{kde}}(\cdot)$ form a valid PDF/CDF pair. The following theorem provides the formal properties of the smoothed kernel density.

Remark: The pair $\big(f_{\text{kde}}(\cdot), F_{\text{kde}}(\cdot)\big)$ from Method 3 satisfy the properties:

(a) $\displaystyle\int_{\ell}^{u} f_{\text{kde}}(x)\,\mathrm{d}x = 1$;
(b) $F'_{\text{kde}}(x) = f_{\text{kde}}(x)$ for all $x \in (\ell, u)$.
(c) F_{kde} is non-decreasing with $F_{\text{kde}}(\ell) = 0$ and $F_{\text{kde}}(u) = 1$.

Points (a)-(c) are foundational requirements for any proper CDF and PDF and confirm that Method 3 produces a pair of valid distribution functions.

4.3 Robust Equilibrium Implementation

The previous section outlines the method for transforming discrete valuations into continuous distributions on $[\ell, u] \subseteq [0, 1]$. In this section we utilize these distributions to write the first-order-conditions stated in Equation 1, as it pertains to our model. We define the linear *theoretical baseline strategy functions* for bilateral trade as $B^0(\cdot)$ and $S^0(\cdot)$.[15] We then develop a revised version of Equation 1 using the smoothed distributions generated in Method 3.

Definition 7 (k-DA Theoretical Baseline Strategies)**.** *Given the interval $[\underline{v}, \bar{c}]$ where trade occurs with positive probability between a buyer and seller with valuations v_b and v_s respectively, and the mechanism parameter k, we define the theoretical baseline strategy functions for the buyer and seller as*

$$B^0(v^{\text{proc}}_{b,\text{ser}}) \equiv \underline{v} + (v^{\text{proc}}_{b,\text{ser}} - \underline{v})(1-k), \quad v^{\text{proc}}_{b,\text{ser}} \in [\underline{v}, 1] \tag{9}$$

$$S^0(v^{\text{proc}}_{s,\text{ser}}) \equiv \bar{c} - (\bar{c} - v^{\text{proc}}_{s,\text{ser}})(1-k), \quad v^{\text{proc}}_{s,\text{ser}} \in [0, \bar{c}] \tag{10}$$

Adaptation of the linear model for a buyer's bid double auction (BBDA) from Satterthwaite and Williams (1989) matches the framework of our model, because we consistently define buyer parameters first before later enforcing symmetry on the seller.[16] Furthermore, Definition 7 captures the idea that a more aggressive buyer (larger k) bids closer to his valuation, and the seller symmetrically offers further from her valuation.

A necessary step for adapting Equation 1 to our model, is computing the probability that any possible bid, defined as λ, wins the auction for the buyer. This involves the joint probability of two separate events.

(a) The seller submits an offer $S(v^{\text{proc}}_s) \leqslant \lambda$;

[15] Equations 9 and 10 taken directly from Satterthwaite and Williams (1989)
[16] Reference Appendix C Figure 1 for the formal algorithm used to define theoretical baseline strategies

(b) the buyer with at least $m-1$ alternative possible valuations bids $B(v_b^{\text{proc}}) \geqslant \lambda$.

These probabilities enter multiplicatively into the buyer's objective function, so to compute optimal bidding behavior their product must be evaluated at each possible valuation. Computation of these small probabilities in a market with many possible buyer and seller valuations (large $\mathcal{V}^{\text{proc}}$) can quickly lead to underflow when the resulting values approach machine epsilon.[17] Thus, we elect to logarithmically transform relevant terms from Equation 1 to improve stability. We accomplish this by:

(a) avoiding computation of near-zero probabilities;
(b) converting products of probabilities into sums of logs, which are more stable and efficient to compute; and
(c) allowing the use of a root-finding algorithm with bounded derivatives, since log-transformation smooths multiplicative sharpness.

Using this motivation, we define the following log-probability functions.[18,19]

Definition 8 (Numerical Probability Integration)**.** *Let $f_{kde}(v_{b,\text{ser}}^{\text{proc}})$ and $f_{kde}(v_{s,\text{ser}}^{\text{proc}})$ be the smoothed probability density functions for any possible processed buyer and seller valuation that trades with positive probability, obtained by Method 3. For any λ, the log-probabilities are*

$$\log K_{n,m}(\lambda) = n \cdot \log\Big(\int_{\ell}^{\min(\lambda,u)} f_{kde}(v_{s,\text{ser}}^{\text{proc}})\,\mathrm{d}(v_{s,\text{ser}}^{\text{proc}})\Big), \tag{11}$$

$$\log L_{n,m}(\lambda) = (m-1) \cdot \log\Big(\int_{\max(\lambda,\ell)}^{u} f_{kde}(v_{b,\text{ser}}^{\text{proc}})\,\mathrm{d}(v_{b,\text{ser}}^{\text{proc}})\Big), \tag{12}$$

$$\log M_{n,m}(\lambda) = \log K_{n,m}(\lambda) \;+\; \log L_{n,m}(\lambda). \tag{13}$$

Here,

(a) The integral in Equation 11 is the probability that the seller submits an offer below λ.
(b) The integral in Equation 12 is the probability that at any of his $m-1$ other possible valuations the buyer bids at least λ.
(c) Together, these yield the log-probability of the joint event in Equation 13 needed for the first-order conditions.

To compute a numerically stable derivative of a real-valued function f at point x, we apply the

[17] From Higham (2002), we compute probabilities in log-space to avoid underflow/overflow when multiplying small terms $P = \prod p_i$, since $\log P = \sum \log p_i$. Clamping exponents to ± 700 ensures $\exp(\cdot)$ stays within float64 limits.
[18] Reference Appendix C Figure 2 for the formal algorithm used to define the numerical probability integration
[19] Consult Equation 1 and subsequent analysis for the original forms and definition of variables $K_{n,m}$, $L_{n,m}$ and $M_{n,m}$ used in Rustichini et al. (1994).

complex-step method introduced by Martins et al. (2003). This technique avoids cancellation error and achieves machine-precision accuracy using an imaginary perturbation, defined as i.[20]

Definition 9 (Complex-Step Derivative Approximation)**.**

$$f'(x) \approx \frac{\Im\{f(x+i\varepsilon)\}}{\varepsilon}, \text{ where } \varepsilon = 10^{-20}.$$

To prevent numerical underflow or instability in later steps (e.g., when derivatives enter logarithms), we enforce a safe lower bound:

$$f'(x) \leftarrow \begin{cases} \max\big(D(x),\, \delta\big), & D(x) > 0, \\ \min\big(D(x),\, -\delta\big), & D(x) < 0, \end{cases}$$

where $\delta = 10^{-8}$, and $D(x) = \frac{\Im\{f(x+i\varepsilon)\}}{\varepsilon}$ is the raw complex-step estimate. This method yields stable gradients even in near-flat regions of $f(\cdot)$.

The complex-step derivative method in Definition 9 isolates derivatives in the imaginary component, avoiding subtractive errors inherent in finite differences and achieving machine precision. This stability is critical for accurate gradient computations in log-probability or optimization contexts.

Thus far, we have restricted our stabilization analysis to the intervals $[\ell, u]$ over which trade occurs with positive probability. It is also relevant to examine stabilization of those values $v_b^{\text{proc}} \in \mathcal{V}^{\text{proc}} \backslash \mathcal{V}_{\text{ser}}^{\text{proc}}$ and $v_s^{\text{proc}} \in \mathcal{V}^{proc} \backslash \mathcal{V}_{\text{ser}}^{\text{proc}}$ for the buyer and seller.[21]

Definition 10 (Stabilized PDF Extension)**.** *Let $\{\hat{x}_j\}_{j=1}^M \subset [\ell, u]$ be a fine grid on the serious interval and let $\{\hat{f}_j\}_{j=1}^M$ be the corresponding normalized KDE estimates from Method 3. We then define an extended PDF*

$$\tilde{f}_{kde}(x) = \begin{cases} \textit{(linear interpolation of } (\hat{x}_j, \hat{f}_j)\textit{)}, & x \in [\ell, u], \\ \varepsilon, & x \notin [\ell, u], \end{cases}$$

where $\varepsilon > 0$ is a tiny constant (e.g. 10^{-100}). We denote by $\tilde{f}_{kde}(v_b^{\text{proc}})$ and $\tilde{f}_{kde}(v_s^{\text{proc}})$ the two such extensions for buyer and seller densities.

The purpose of this construction is two-fold:

(a) On $[\ell, u]$, $\tilde{f}_{kde}(v_b^{\text{proc}})$ and $\tilde{f}_{kde}(v_s^{\text{proc}})$ recovers the original KDE values via a smooth, piecewise-linear interpolant.

(b) Outside $[\ell, u]$, setting $\tilde{f}_{kde}(v_b^{\text{proc}})$, $\tilde{f}_{kde}(v_s^{\text{proc}}) = \varepsilon$ prevents the density from vanishing, so that $\tilde{f}_{kde}(v_b^{\text{proc}})$ and $\tilde{f}_{kde}(v_s^{\text{proc}})$ remain finite in all subsequent log-probability computations.

[20]Reference Appendix C Figure 3 for the formal algorithm for the complex-step derivative approximation

[21]Reference Appendix C Figure 4 for the formal algorithm for stabilized PDF extension

We enforce a minimal density floor ($\varepsilon = 10^{-100}$) outside the support. This prevents numerical failures in root-finding when bids or offers near the boundaries are outside of the serious trading interval.

Prior to formalizing our stabilized first-order-conditions-solver, define Λ as a Log-clamp constant which prevents overflow in exponentiation.

Remark: Numerical Safeguards in Method 4

(a) *Log-clamping* at $\pm\Lambda$ prevents overflow in $\exp(\cdot)$.
(b) *Density flooring* $\tilde{f}_{kde,b}(\cdot),\ \tilde{f}_{kde,s}(\cdot) \geqslant \varepsilon$ ensures all integrals and logs remain finite.
(c) *Complex-step derivatives* (Definition 9) give machine-precision gradients without cancellation error (Martins et al. (2003)).
(d) *Bracketed root search* within $[\ell, u]$ guarantees a valid solution exists in the serious interval.

Method 4 (Stabilized First-Order-Conditions Solver)**.** *Following the model from Higham (2002):*

(a) Continuous, stabilized PDFs $\tilde{f}_{kde}(v_b^{\text{proc}})$, $\tilde{f}_{kde}(v_s^{\text{proc}})$ *on* $[\ell, u]$ *(via KDE; Definition 6),*
(b) Baseline strategy functions $B^0(v_b^{\text{proc}})$, $S^0(v_s^{\text{proc}})$ *(Definition 7),*
(c) Complex-step derivative approximation (Definition 9),
(d) Log-clamp constant $\Lambda = 700$*,*
(e) Positive floor $\varepsilon > 0$ *for all densities,*

we solve for each buyer's equilibrium bid $B^*(v_{b,\text{ser}}^{\text{proc}})$ *by finding the root* $\lambda \in [\ell, u]$ *of*

$$\mathcal{F}(v_{b,\text{ser}}^{\text{proc}}, \lambda) = (v_{b,\text{ser}}^{\text{proc}} - k)\Big[n\, K_{n,m}(\lambda)\, \frac{\tilde{f}_{kde,s}(\lambda)}{|S'(\lambda)|} + (m-1)\, L_{n,m}(\lambda)\, \frac{\tilde{f}_{kde,b}(v_{b,\text{ser}}^{\text{proc}})}{|B'(v_{b,\text{ser}}^{\text{proc}})|}\Big] - k\, M_{n,m}(\lambda) = 0, \quad (14)$$

where

$$\log M_{n,m}(\lambda) = \text{clip}\big(\log K_{n,m}(\lambda) + \log L_{n,m}(\lambda),\ -\Lambda, \Lambda\big), \quad (15)$$

$$M_{n,m}(\lambda) = \exp\big(\log M_{n,m}(\lambda)\big). \quad (16)$$

Finally, apply a robust 1-D root-finder (see Brent (1973); Judd (1998)) on $[\ell, u]$ *to obtain* $B^*(v_{b,\text{ser}}^{\text{proc}})$*.*

The solver (Method 4) operationalizes the equilibrium condition from Rustichini et al. (1994) through four key mechanisms.[22,23]

Proposition 1 (Review of Processes used in Method 4)**.**

1. ***Log-Domain Stabilization****:*

(a) Probabilities $K_{n,m}, L_{n,m}$ *decay exponentially with the number of bids and offers,* m*,* n

[22]Reference Appendix C Figure 5 for the formal algorithm for stabilized first-order-conditions solver
[23]This algorithmic method originated in Brent (1973), however the application from Judd (1998) is a modern approach more relevant to the class of mechanism design and optimization problems analyzed in this paper.

(b) Direct computation risks underflow (e.g., 10^{-300} becomes zero)

(c) Working in log-space preserves precision: $\log(10^{-300}) = -300$ remains valid

(d) Clipping at ± 700 prevents overflow in exponentiation ($\exp(700) \approx 10^{304}$ nears float64 max)

2. ***Derivative Regularization****:*

(a) Complex-step differentiation gives machine-precision gradients without subtractive cancellation:

$$B'(v_{b,\text{ser}}^{\text{proc}}) = \frac{\Im[B^0(v_{b,\text{ser}}^{\text{proc}} + i\varepsilon)]}{\varepsilon} \tag{17}$$

(b) Flooring derivatives at $\delta = 10^{-8}$ prevents division instability in flat regions

3. ***Density Containment****:*

(a) PDF extrapolation with $\tilde{f}_{kde}(v_{b,\text{ser}}^{\text{proc}})$, $\tilde{f}_{kde}(v_{s,\text{ser}}^{\text{proc}}) \geqslant 10^{-100}$ ensures:

$$\int_{\lambda}^{1} \tilde{f}_{kde}(v_{b,\text{ser}}^{\text{proc}}) \mathrm{d}v_{b,\text{ser}}^{\text{proc}} > 0 \quad \forall \lambda < 1 \tag{18}$$

$$\int_{\underline{v}}^{\lambda} \tilde{f}_{kde}(v_{s,\text{ser}}^{\text{proc}}) \mathrm{d}v_{b,\text{ser}}^{\text{proc}} > 0 \quad \forall \lambda > \underline{v} \tag{19}$$

(b) Guarantees $\log(K_{n,m})$ and $\log(L_{n,m})$ remain finite

4. ***Failure Fallbacks****:*

(a) Bracketed root search (Brent's method) always converges within $[\underline{v}, 1]$

(b) On solver failure, falls back to theoretical bid $B^0(v_{b,\text{ser}}^{\text{proc}})$ with warning

We are now ready to append Method 4 to computing the equilibrium bids and offers for the buyer and seller, defined as $B^*(v_{b,\text{ser}}^{\text{proc}})$ and $S^*(v_{s,\text{ser}}^{\text{proc}})$, respectively. All inputs have previously been defined, but we repeat them here for ease of analysis.[24]

Inputs:

(a) Processed serious valuation set $v_{b,\text{ser}}^{\text{proc}} \subseteq [\ell, u]$,
(b) Baseline strategy $B^0(v_{b,\text{ser}}^{\text{proc}})$ (Definition 7),
(c) First-order-condition function $\mathcal{F}(v_{b,\text{ser}}^{\text{proc}}, \lambda)$ (Equation 14),
(d) Interval bounds $\ell = \underline{v}$, $u = \bar{c}$

Output: Equilibrium bid mapping $B^*(v_{b,\text{ser}}^{\text{proc}}) : \mathcal{V}^{\text{proc}} \to \mathbb{R}$.

Method 5 (Buyer-Equilibrium Computation)**.**

[24]Reference Appendix C Figure 5 for the formal algorithm for part (a) of the buyer equilibrium computation, Reference Appendix C Figure 6 for the formal algorithm for buyer equilibrium calculation with "safe" brackets, Reference Appendix C Figure 7 for the formal algorithm of the extension of the buyer-equilibrium computation to all valuations

(a) For each $v_{b,\text{ser}}^{\text{proc}} \in \mathcal{V}^{\text{proc}}$*, set*

$$B^*(v_{b,\text{ser}}^{\text{proc}}) \;=\; \underset{\lambda\in[\underline{\lambda},\,\overline{\lambda}]}{\text{root}}\; \mathcal{F}(v_{b,\text{ser}}^{\text{proc}}, \lambda) \tag{20}$$

where the "safe bracket" is

$$\underline{\lambda} = \max\big(\ell,\; B^0(v_{b,\text{ser}}^{\text{proc}}) - 0.1\big), \quad \overline{\lambda} = \min\big(u,\; v_\wr b, \text{ser}^{\text{proc}}\big).$$

If the bracket collapses (i.e. $\underline{\lambda} \geqslant \overline{\lambda}$*), revert to* $[\ell, \min(u, v)]$*. Use a robust 1D solver (e.g. Brent's method) to find the unique root in that interval.*

(b) Extend to all valuations (Definition 10) *For any* $v_b^{\text{proc}} \in \mathcal{V}^{proc}\backslash\mathcal{V}_{\text{ser}}^{\text{proc}}$*, define*

$$B^*(v_b^{\text{proc}}) = \begin{cases} B^*(\ell) \;-\; (\ell - v_b^{\text{proc}}), & v_b^{\text{proc}} < \ell, \\ B^0(v_b^{\text{proc}}), & v_b^{\text{proc}} > u. \end{cases}$$

This projects the computed endpoint values outward with unit slope, and uses the baseline strategy as a fallback.

The seller's first-order-conditions for equilibrium are not defined explicitly within Rustichini et al. (1994) in the same manner as the first-order-conditions for the buyer (Equation 1). As such, we craft a clever adaptation of Method 5 which works as desired because of our symmetry requirements. Define a as any possible offer for the seller.[25] Define a function $\mathcal{G}(\cdot)$ as the root finding function for the seller, where $\mathcal{G}(\cdot)$ follows a symmetric implementation to the bracketed root search function for the buyer $\mathcal{F}(\cdot)$.[26]

Inputs:

(a) Serious seller set $v_{s,\text{ser}}^{\text{proc}} \subseteq [\ell, u]$,

(b) Baseline strategy $S(v_{b,\text{ser}}^{\text{proc}})$ (Definition 7),

(c) Seller first-order-condition $\mathcal{G}(v_{b,\text{ser}}^{\text{proc}}, a) = (a - v_{b,\text{ser}}^{\text{proc}}) \;-\; \dfrac{\int_a^u f_b(x)\,\mathrm{d}x}{f_b(a)},\;\; a \in [v_{b,\text{ser}}^{\text{proc}}, u]$,

(d) Interval bounds $\ell = \underline{v}$, $u = \bar{c}$.

Output: Equilibrium ask mapping $S^*(v_{b,\text{ser}}^{\text{proc}}) : \mathcal{V}^{\text{proc}} \to \mathbb{R}$.

Method 6 (Seller-Equilibrium Computation)**.**

(a) Solve within the serious interval. *For each* $v_{s,\text{ser}}^{\text{proc}} \in \mathcal{T}^{\text{proc}}$*, find*

$$S^*(v_{s,\text{ser}}^{\text{proc}}) \;=\; \underset{a\in[\underline{a},\,\overline{a}]}{\text{root}}\; \mathcal{G}(v_{s,\text{ser}}^{\text{proc}}, a)$$

[25] Reference Appendix C Figure 8 for the formal algorithm of the seller-equilibrium computation, Reference Appendix C Figure 9 for the formal algorithm for the extension of seller-equilibrium computation to all valuations

[26] This formalization for a follows the same logic as the formalization of λ in the buyer's first-order-conditions for equilibrium.

where the "safe bracket" is

$$\underline{a} = \max\big(v^{\text{proc}}_{s,\text{ser}},\ \ell\big), \quad \overline{a} = \min\big(u,\ v^{\text{proc}}_{s,\text{ser}} + 0.1\,(u-\ell)\big).$$

If $\underline{a} \geqslant \overline{a}$*, fall back to* $[\ v^{\text{proc}}_{s,\text{ser}},\ \min(u, v^{\text{proc}}_{s,\text{ser}})\ ]$*. Use a robust 1D solver to locate the unique root.*

(b) Extend to all valuations. *For any non-serious* $v^{\text{proc}}_s \in \mathcal{V}^{\text{proc}} \backslash \mathcal{V}^{\text{proc}}_{\text{ser}}$*, define*

$$S^*(v^{\text{proc}}_s) = \begin{cases} S^*(\ell)\ -\ (\ell - v_s), & v^{\text{proc}}_s < \ell, \\ \\ S^0(v^{\text{proc}}_s), & v^{\text{proc}}_s > u, \end{cases}$$

i.e. project the endpoint asks outward with unit slope and use the theoretical baseline as fallback.

Proposition 2 (Boundary Consistency)**.** *The extended strategies* $B(v^{\text{proc}}_b)$ *and* $S(v^{\text{proc}}_s)$ *satisfy*

$$\lim_{v_b \uparrow \ell} B(v^{\text{proc}}_b) = B^*(\ell), \quad \lim_{v_s \downarrow u} S(v^{\text{proc}}_s) = S^*(u),$$

and both are linear with slope 1 outside $[\ell, u]$*.*

```
NOTE: THESE ERROR MESSAGES ARE THEORETICALLY SOUND, DUE TO MISREPRESENTATION NO STABLE EQUILIBRIUM SHOULD BE POSSIBLE
Using theoretical bid for v=0.2025: f(a) and f(b) must have different signs
Warning: Seller root-finding failed for c=0.1710. Using theoretical offer. Error: f(a) and f(b) must have different signs

================================================================
========== STRATEGIC EQUILIBRIUM (RWS 1994 COMPATIBLE) ==========
================================================================
Trading Interval: [0.1724, 0.8276]
Number of Possible Buyer Values: 20, Number of Possible Seller Values: 20
Mechanims Parameter k: 0.50

------------------------------------------------------------
---------- BUYER BIDS (B(vb)) ----------
Valuation    Computed          Theoretical       Δ
0.00000    0.00000 (0.00%)   0.08621       -0.0862
0.03448    0.03448 (33.33%)  0.10345       -0.0690
0.06897    0.06897 (57.14%)  0.12069       -0.0517
0.10345    0.10345 (75.00%)  0.13793       -0.0345
0.13793    0.13793 (88.89%)  0.15517       -0.0172
0.17241    0.17241 (100.00%)  0.17241       +0.0000
0.20690    0.18966 (100.00%)  0.18966       +0.0000
0.24138    0.20690 (100.00%)  0.20690       +0.0000
0.27586    0.22414 (100.00%)  0.22414       +0.0000
0.31034    0.24138 (100.00%)  0.24138       +0.0000
0.34483    0.25862 (100.00%)  0.25862       +0.0000
0.37931    0.27586 (100.00%)  0.27586       +0.0000
0.41379    0.29310 (100.00%)  0.29310       +0.0000
0.44828    0.31034 (100.00%)  0.31034       +0.0000
0.48276    0.32759 (100.00%)  0.32759       +0.0000
0.51724    0.34483 (100.00%)  0.34483       +0.0000
0.55172    0.36207 (100.00%)  0.36207       +0.0000
0.58621    0.37931 (100.00%)  0.37931       +0.0000
0.62069    0.39655 (100.00%)  0.39655       +0.0000
0.65517    0.41379 (100.00%)  0.41379       +0.0000
0.68966    0.43103 (100.00%)  0.43103       +0.0000
0.72414    0.44828 (100.00%)  0.44828       +0.0000
0.75862    0.46552 (100.00%)  0.46552       +0.0000
0.79310    0.48276 (100.00%)  0.48276       +0.0000
0.82759    0.50000 (100.00%)  0.50000       +0.0000
0.86207    0.53448 (103.33%)  0.51724       +0.0172
0.89655    0.56897 (106.45%)  0.53448       +0.0345
0.93103    0.60345 (109.38%)  0.55172       +0.0517
0.96552    0.63793 (112.12%)  0.56897       +0.0690
1.00000    0.67241 (114.71%)  0.58621       +0.0862
```

(a) Equilibrium Buyer Bids $B^*(v^{\text{proc}}_b)$

```
------------------------------------------------------------
---------- SELLER OFFERS (S(vs)) ----------
Cost         Computed          Theoretical       Δ
0.00000    0.32759 (79.17%)  0.41379       -0.0862
0.03448    0.36207 (84.00%)  0.43103       -0.0690
0.06897    0.39655 (88.46%)  0.44828       -0.0517
0.10345    0.43103 (92.59%)  0.46552       -0.0345
0.13793    0.46552 (96.43%)  0.48276       -0.0172
0.17241    0.50000 (100.00%)  0.50000       +0.0000
0.20690    0.51724 (100.00%)  0.51724       +0.0000
0.24138    0.53448 (100.00%)  0.53448       +0.0000
0.27586    0.55172 (100.00%)  0.55172       +0.0000
0.31034    0.56897 (100.00%)  0.56897       +0.0000
0.34483    0.58621 (100.00%)  0.58621       +0.0000
0.37931    0.60345 (100.00%)  0.60345       +0.0000
0.41379    0.62069 (100.00%)  0.62069       +0.0000
0.44828    0.63793 (100.00%)  0.63793       +0.0000
0.48276    0.65517 (100.00%)  0.65517       +0.0000
0.51724    0.67241 (100.00%)  0.67241       +0.0000
0.55172    0.68966 (100.00%)  0.68966       +0.0000
0.58621    0.70690 (100.00%)  0.70690       +0.0000
0.62069    0.72414 (100.00%)  0.72414       +0.0000
0.65517    0.74138 (100.00%)  0.74138       +0.0000
0.68966    0.75862 (100.00%)  0.75862       +0.0000
0.72414    0.77586 (100.00%)  0.77586       +0.0000
0.75862    0.79310 (100.00%)  0.79310       +0.0000
0.79310    0.81034 (100.00%)  0.81034       +0.0000
0.82759    0.82759 (100.00%)  0.82759       +0.0000
0.86207    0.86207 (102.04%)  0.84483       +0.0172
0.89655    0.89655 (104.00%)  0.86207       +0.0345
0.93103    0.93103 (105.88%)  0.87931       +0.0517
0.96552    0.96552 (107.69%)  0.89655       +0.0690
1.00000    1.00000 (109.43%)  0.91379       +0.0862
```

(b) Equilibrium Seller Offers $S^*(v^{\text{proc}}_s)$

Figure 5: Equilibrium Bids and Offers, $B^*(v^{\text{proc}}_b)$ and $S^*(v^{\text{proc}}_s)$

Remark: Figure 5 showcases the equilibrium bids and offers $B^*(v^{\text{proc}}_b)$ and $S^*(v^{\text{proc}}_s)$ for each possible processed buyer and seller valuation in $[0, 1]$. The mechanism showcases the "Computed" bids and offers obtained by Methods 5 and 6, and compares them (in both percentage and numerical terms) against the "Theoretical" bids and offers $B^0(v^{proc}_b)$ and $S^0(v^{\text{proc}}_s)$ dictated by equations 9 and 10. Within the serious trading interval, the computed results align with the theoretical results, while outside the serious trading interval the buyer and seller misrepresent their bids by a greater margin than is dictated by the theoretical model.

4.4 Equilibrium Condition Verification

To verify the equilibrium condition for the buyer in a k-DA, we adapt Equation 2 in our model. Having computed equilibrium bids $B^*(v_b^{\text{proc}})$ and offers $S^*(v_s^{\text{proc}})$ via Method 5 and 6, we now verify that they satisfy the first-order inequality conditions within numerical tolerance.[27] We define γ as a fixed parameter that scales the predictor's effect in the model. We define functions $R_b(\cdot)$ and $R_s(\cdot)$ as the residual functions for the buyer and the seller, where $R_b(v_b^{\text{proc}})$ is the difference between the equilibrium bid and the computed value evaluated at v_b, and $R_s(v_s^{\text{proc}})$ is the difference between the equilibrium offer and the computed value evaluated at v_s.[28]

Method 7 (Stabilized Buyer-Inequality Verification)**.** *Let $v_{b,\text{ser}}^{\text{proc}} \subseteq [\ell, u]$ be the set of serious buyer valuations. For each $v_{b,\text{ser}}^{\text{proc}} \in \mathcal{T}^{\text{proc}}$, confirm*

$$(v_{b,\text{ser}}^{\text{proc}} - B^*(v_{b,\text{ser}}^{\text{proc}})) \, \frac{(m-1)\, L_{n,m}\big(B^*(v_{b,\text{ser}}^{\text{proc}})\big)\, \tilde{f}_{kde}(v_{b,\text{ser}}^{\text{proc}})}{\big|B'(v_{b,\text{ser}}^{\text{proc}})\big|} \;\leqslant\; k \cdot M_{n,m}\big(B^*(v_{b,\text{ser}}^{\text{proc}})\big) \;-\; \gamma, \tag{21}$$

where:

(a) $B^*(v_{b,\text{ser}}^{\text{proc}})$ from Method 4;
(b) $\tilde{f}_{kde}(v_{b,\text{ser}}^{\text{proc}})$, $L_{n,m}$, $M_{n,m}$;
(c) $B'(v_{b,\text{ser}}^{\text{proc}})$ via complex-step (Definition 9), floored at ε;
(d) probabilities log-clamped to $[-\Lambda, \Lambda]$ and densities floored to ε.

Loop over $v_{b,\text{ser}}^{\text{proc}} \in \mathcal{T}^{\text{proc}}$, compute the residual

$$R_b(v_{b,\text{ser}}^{\text{proc}}) = (v_{b,\text{ser}}^{\text{proc}} - B^*(v_{b,\text{ser}}^{\text{proc}}))(m-1)\, \frac{L \cdot \tilde{f}_{kde}(v_{b,\text{ser}}^{\text{proc}})}{|B'(v_{b,\text{ser}}^{\text{proc}})|} - k \cdot M, \tag{22}$$

We craft a clever symmetric equilibrium verification condition for the seller as well.[29]

Method 8 (Stabilized Seller-Inequality Verification)**.** *Let $v_{s,\,\text{ser}}^{\text{proc}} \subseteq [\ell, u]$. For each $v_{s,\text{ser}}^{\text{proc}} \in \mathcal{T}^{\text{proc}}$, confirm*

$$\big(S^*(v_{s,\text{ser}}^{\text{proc}}) - v_{s,\text{ser}}^{\text{proc}}\big) \, \frac{(n-1)\, K_{n,m}\big(S^*(v_{s,\text{ser}}^{\text{proc}})\big)\, \tilde{f}_{kde}(v_{s,\text{ser}}^{\text{proc}})}{\big|S'(v_{s,\text{ser}}^{\text{proc}})\big|} \;\leqslant\; k \cdot N_{n,m}\big(S^*(v_{s,\text{ser}}^{\text{proc}})\big) \;-\; \gamma, \tag{23}$$

where we swap (m, n) and use $\tilde{f}_{\text{kde}}(v_{s,\text{ser}}^{\text{proc}})$, $K_{n,m}$, $N_{n,m}$. Compute

$$R_s(v_{s,\text{ser}}^{\text{proc}}) = (S^*(v_{s,\text{ser}}^{\text{proc}}) - v_{s,\text{ser}}^{\text{proc}})(n-1)\, \frac{K \cdot \tilde{f}_{kde}(v_{s,\text{ser}}^{\text{proc}})}{|S'(v_{s,\text{ser}}^{\text{proc}})|} - k \cdot N, \tag{24}$$

[27] For numerical tolerance We fix constants $\varepsilon = 10^{-12}, \Lambda = 700$ and $\gamma = 10^{-6}$.
[28] Reference Appendix D Figure 1 for the formal algorithm for stabilized buyer-inequality verification
[29] Reference Appendix D Figure 2 for the formal algorithm for stabilized seller-inequality verification

```
======================================================
============== VERIFYING BUYER CONDITIONS ============
======================================================

Market Parameters:
 Number of Possible Buyer Values: 20, Number of Possible Seller Values: 20
- k value: 0.50
- Trading interval: [0.1724, 0.8276]

Valuation    Bid          LHS          Threshold    Status
------------------------------------------------------------------
0.17129    | 0.17241    | -4.27e-14  | 1.00e-06   | OK
0.20253    | 0.18747    | 5.79e-01   | 1.00e-06   | VIOLATED
0.23887    | 0.20564    | 7.22e-01   | 1.00e-06   | VIOLATED
0.27416    | 0.22329    | 6.44e-01   | 1.00e-06   | VIOLATED
0.31157    | 0.24199    | 5.33e-01   | 1.00e-06   | VIOLATED
0.34184    | 0.25713    | 4.12e-01   | 1.00e-06   | VIOLATED
0.37795    | 0.27518    | 2.69e-01   | 1.00e-06   | VIOLATED
0.41253    | 0.29247    | 1.77e-01   | 1.00e-06   | VIOLATED
0.44930    | 0.31085    | 1.12e-01   | 1.00e-06   | VIOLATED
0.48100    | 0.32671    | 6.90e-02   | 1.00e-06   | VIOLATED
0.51706    | 0.34474    | 3.51e-02   | 1.00e-06   | VIOLATED
0.55657    | 0.36449    | 1.94e-02   | 1.00e-06   | VIOLATED
0.58635    | 0.37938    | 1.17e-02   | 1.00e-06   | VIOLATED
0.62562    | 0.39902    | 5.80e-03   | 1.00e-06   | VIOLATED
0.65481    | 0.41361    | 3.47e-03   | 1.00e-06   | VIOLATED
0.69004    | 0.43123    | 1.72e-03   | 1.00e-06   | VIOLATED
0.71994    | 0.44618    | 8.93e-04   | 1.00e-06   | VIOLATED
0.75606    | 0.46424    | 3.54e-04   | 1.00e-06   | VIOLATED
0.79370    | 0.48306    | 1.39e-04   | 1.00e-06   | VIOLATED
0.82922    | 0.50082    | 4.66e-05   | 1.00e-06   | VIOLATED

============================================================
WARNING: Potential violations detected
- Possible causes:
  1. Numerical approximations in density estimation
  2. Boundary effects near trading interval edges
  3. Discontinuities in strategic bidding functions
============================================================
```

(a) Validation of Equilibrium Bids $B^*(v_{b,\text{ser}}^{\text{proc}})$

```
======================================================
============ VERIFYING SELLER CONDITIONS =============
======================================================

Market Parameters:
- Number of Buyer Values: 20, Number of Seller Values: 20
- k value: 0.50
- Trading interval: [0.1724, 0.8276]

Valuation    Offer        LHS          Threshold    Status
------------------------------------------------------------------
0.17104    | 0.49931    | 2.57e-05   | 1.00e-06   | VIOLATED
0.20368    | 0.51563    | 7.78e-05   | 1.00e-06   | VIOLATED
0.24237    | 0.53498    | 1.95e-04   | 1.00e-06   | VIOLATED
0.27411    | 0.55085    | 3.97e-04   | 1.00e-06   | VIOLATED
0.31087    | 0.56923    | 8.20e-04   | 1.00e-06   | VIOLATED
0.34219    | 0.58489    | 1.76e-03   | 1.00e-06   | VIOLATED
0.37520    | 0.60139    | 3.77e-03   | 1.00e-06   | VIOLATED
0.41405    | 0.62082    | 7.29e-03   | 1.00e-06   | VIOLATED
0.44932    | 0.63845    | 1.37e-02   | 1.00e-06   | VIOLATED
0.48524    | 0.65641    | 2.67e-02   | 1.00e-06   | VIOLATED
0.52013    | 0.67386    | 4.97e-02   | 1.00e-06   | VIOLATED
0.55147    | 0.68953    | 8.82e-02   | 1.00e-06   | VIOLATED
0.58880    | 0.70819    | 1.58e-01   | 1.00e-06   | VIOLATED
0.62320    | 0.72539    | 2.46e-01   | 1.00e-06   | VIOLATED
0.65213    | 0.73986    | 3.64e-01   | 1.00e-06   | VIOLATED
0.68506    | 0.75632    | 5.04e-01   | 1.00e-06   | VIOLATED
0.72447    | 0.77529    | 5.64e-01   | 1.00e-06   | VIOLATED
0.75878    | 0.79236    | 6.70e-01   | 1.00e-06   | VIOLATED
0.78813    | 0.80725    | 6.47e-01   | 1.00e-06   | VIOLATED
0.82811    | 0.82759    | -2.01e-14  | 1.00e-06   | OK

============================================================
WARNING: Seller inequality violations detected
- Potential causes:
  1. Boundary effects in seller offer strategies
  2. Discontinuities in seller virtual valuations
  3. Numerical instability in high-density regions
============================================================
```

(b) Validation of Equilibrium Offers $S^*(v_{s,\text{ser}}^{\text{proc}})$

Figure 6: Validation of Equilibrium Bids and Offers, $B^*(v_{b,\text{ser}}^{\text{proc}})$ and $S^*(v_{s,\text{ser}}^{\text{proc}})$

Remark: Figure 7 showcases the results of the stabilized buyer and seller verification equations (Equations 21 and 22) for all possible processed buyer and seller valuations for which trade occurs with positive probability. The reduced inequality equation outlined at the end of Method 6 is $R_b(v_{b,\text{ser}}^{\text{proc}}) > \gamma$ and $R_s(v_{s,\text{ser}}^{\text{proc}}) > \gamma$ for the buyer and seller respectively, where $R_b(v_{b,\text{ser}}^{\text{proc}})$ and $R_s(v_{s,\text{ser}}^{\text{proc}})$ are denoted as "LHS" in the visual, and $\gamma = 10^{-6}$ is the fixed parameter denoted as "Threshold". In the specific case of $N = 30$ and $k = .5$ there are many observed violations. This does not immediately oppose the theoretical conclusions from Rustichini et al. (1994), however. An important aspect of the paper excluded from our model, is improved efficiency as $\mathcal{V}$ grows large.[30]

4.5 Equilibrium Strategy Preservation

After formalizing the model for buyer and seller equilibrium bids and values $B^*(v_{b,\text{ser}}^{\text{proc}})$ and $S^*(v_{s,\text{ser}}^{\text{proc}})$, we formalize a mechanism to verify the individual rationality of the buyer and seller. We define individual rationality (IR) as the assumption that no buyer or seller receives a lower payoff from participating in the mechanism than they otherwise would have by non-participation. We define functions $\mathcal{U}_b(\cdot)$ and $\mathcal{U}_s(\cdot)$ which give the profit (utility) received by the buyer and seller from participation in the mechanism. In the event that the buyer or seller trades at their equilibrium bid

[30]Reference the link the GitHub repository located in Appendix H to see how for cases where the discrete valuation grid is large, the number of instances where equilibrium bids and offers violate rationality and incentive compatibility gradually decreases.

or offer $B^*(v_{b,\text{ser}}^{\text{proc}})$ and $S^*(v_{s,\text{ser}}^{\text{proc}})$, they profit an amount $\mathcal{U}_b(B^*(v_{b,\text{ser}}^{\text{proc}}))$ and $\mathcal{U}_s(S^*(v_{s,\text{ser}}^{\text{proc}}))$, respectively. We introduce a serialization process which verifies that *all* possible bids and offers for the buyer and seller satisfy individual rationality. We then organize all possible buyer and seller values v_b and v_s, bids and offers at these values $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$, and profit from trade at these bids and offers $\mathcal{U}_b(B(v_b^{\text{proc}}))$ and $\mathcal{U}_s(S(v_s^{\text{proc}}))$. We formulate tables, defined as $\mathcal{D}_b$ and $\mathcal{D}_s$ where the output from $\mathcal{D}_b$ and $\mathcal{D}_s$ is a complete, ordered data set of the form $[v_b^{\text{proc}}, B(v_b^{\text{proc}}), \mathcal{U}_b(B(v_b^{\text{proc}}))]$ and $[v_s^{\text{proc}}, S(v_s^{\text{proc}}), \mathcal{U}_s(S(v_s^{\text{proc}}))]$ for the buyer and seller, respectively.[31] While this process is useful for analysis, it serves another important function for the researcher by allowing them to easily export the compact $\mathcal{D}_b$ and $\mathcal{D}_s$ data sets.[32,33]

Method 9 (Strategy Serialization)**.**

(a) Validate equilibrium consistency:

$$\forall\, v_b^{\text{proc}}, v_s^{\text{proc}} \in \mathcal{V}^{\text{proc}} : \begin{cases} B(v_b^{\text{proc}}) \in [\ell, u] \\ S(v_s^{\text{proc}}) \in [\ell, u] \end{cases} \tag{25}$$

(b) Encode strategies as ordered pairs:

$$\mathcal{D}_b = \{(v_{bi}^{\text{proc}}, B(v_{bi}^{\text{proc}}), \mathcal{U}_b(v_{bi}^{\text{proc}}))\}_{i=1}^{m} \tag{26}$$

$$\mathcal{D}_s = \{(v_{sj}^{\text{proc}}, S(v_{sj}^{\text{proc}}), \mathcal{U}_s(v_{sj}^{\text{proc}}))\}_{j=1}^{n} \tag{27}$$

where, $\mathcal{U}_b(B(v_b^{\text{proc}})) = v_b^{\text{proc}} - B(v_b^{\text{proc}})$ *and* $\mathcal{U}_s(Sv_s^{\text{proc}})) = S(v_s^{\text{proc}}) - v_s^{\text{proc}}$.

Theorem 2 (Strategy Conservation)**.** *The serialization process preserves:*

(a) Monotonicity: $B(v_{b1}^{\text{proc}}) \leqslant B(v_{b2}^{\text{proc}})$ *for* $v_{b1}^{\text{proc}} \leqslant v_{b2}^{\text{proc}}$
(b) Boundary compliance: $B(\underline{v}) = \underline{v} + \epsilon$
(c) Utility coherence: $\mathcal{U}_b(B(v_b^{\text{proc}})) + \mathcal{U}_s(S(v_s^{\text{proc}})) \leqslant v_b^{\text{proc}} - v_s^{\text{proc}}$

[31] To keep prose relatively accessible for a reader unfamiliar with the software used to generate the algorithms in this paper, we denote $\mathcal{D}_b$ and $\mathcal{D}_s$ here as "tables". More formally, $\mathcal{D}_b$ and $\mathcal{D}_s$ are compact "data-frames".

[32] Reference Appendix D Figure 2 for the formal algorithm for stabilized seller-inequality verification

[33] Specifically, in our case the serialized data sets $\mathcal{D}_b$ and $\mathcal{D}_s$ are exported from the workbook housing the algorithm used to generate the results from this section, and are imported into the workbook housing the algorithm for Section 6.

Constrained Equilibrium Strategies:

Buyer Strategies B(vb) | 30 agents

Buyer Valuation	Bid B(vb)	Buyer Utility
0.00000	0.00000	0.00000
0.03448	0.03448	-0.00000
0.06897	0.06897	0.00000
0.10345	0.10345	0.00000
0.13793	0.13793	0.00000
0.17241	0.17241	0.00000
0.20690	0.18966	0.01724
0.24138	0.20690	0.03448
0.27586	0.22414	0.05172
0.31034	0.24138	0.06897
0.34483	0.25862	0.08621
0.37931	0.27586	0.10345
0.41379	0.29310	0.12069
0.44828	0.31034	0.13793
0.48276	0.32759	0.15517
0.51724	0.34483	0.17241
0.55172	0.36207	0.18966
0.58621	0.37931	0.20690
0.62069	0.39655	0.22414
0.65517	0.41379	0.24138
0.68966	0.43103	0.25862
0.72414	0.44828	0.27586
0.75862	0.46552	0.29310
0.79310	0.48276	0.31034
0.82759	0.50000	0.32759
0.86207	0.53448	0.32759
0.89655	0.56897	0.32759
0.93103	0.60345	0.32759
0.96552	0.63793	0.32759
1.00000	0.67241	0.32759

(a) IR Verification for all Bids $B(v_b^{\text{proc}})$

Seller Strategies S(vs) | 30 agents

Seller Valuation	Offer S(vs)	Seller Utility
0.00000	0.32759	0.32759
0.03448	0.36207	0.32759
0.06897	0.39655	0.32759
0.10345	0.43103	0.32759
0.13793	0.46552	0.32759
0.17241	0.50000	0.32759
0.20690	0.51724	0.31034
0.24138	0.53448	0.29310
0.27586	0.55172	0.27586
0.31034	0.56897	0.25862
0.34483	0.58621	0.24138
0.37931	0.60345	0.22414
0.41379	0.62069	0.20690
0.44828	0.63793	0.18966
0.48276	0.65517	0.17241
0.51724	0.67241	0.15517
0.55172	0.68966	0.13793
0.58621	0.70690	0.12069
0.62069	0.72414	0.10345
0.65517	0.74138	0.08621
0.68966	0.75862	0.06897
0.72414	0.77586	0.05172
0.75862	0.79310	0.03448
0.79310	0.81034	0.01724
0.82759	0.82759	0.00000
0.86207	0.86207	0.00000
0.89655	0.89655	0.00000
0.93103	0.93103	0.00000
0.96552	0.96552	0.00000
1.00000	1.00000	0.00000

(b) IR Verification for all Offers $S(v_s^{\text{proc}})$

Figure 7: Individual Rationality Verification for all Bids and Offers, $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$

Remark: Figure 8 showcases the formal results from the serialization process, formalized by $\mathcal{D}_b$ and $\mathcal{D}_s$ respectively. For both cases, buyers on the left and sellers on the right, we showcase all possible processed buyer and seller valuations v_b^{proc} and v_s^{proc} on the interval $[0, 1]$, and compare these to the computed bids and offers $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$ from the previous section. We then display the profit to the buyer and seller in the event of trade at these bids and offers $\mathcal{U}_b(B(v_b^{\text{proc}}))$ and $\mathcal{U}_s(S(v_s^{\text{proc}}))$ respectively. From Figure 8, we verify that while the buyer and seller with valuations outside of the serious trading interval receive zero utility from their bids and offers by participating in the mechanism, over the interval where the buyer and seller trade with positive probability, both buyer and seller receive positive profit from participation in the mechanism.

5 A Model with a Market-Maker

For a single buyer and seller behaving strategically in the market for a single object, responses to rational beliefs about an opponent's actions destroy the possibility of trade at any fixed-point

equilibrium. To reintroduce efficiency into the market, we allow for the presence of a market-maker who can be either a net source or sink of money, but who cannot himself own the object. A trading mechanism augmented by the market-maker is characterized by three outcome functions (p, x_1, x_2), where $p(v_1, v_2)$ is the probability that the object is transferred from the seller to the buyer, $x_1(v_1, v_2)$ is the expected payment from the market-maker to the seller, and $x_2(v_1, v_2)$ is the expected payment from the buyer to the market-maker when v_1 and v_2 are the reported valuations of the buyer and seller. Consider $\bar{x_1}(v_1)$, $\bar{x_2}(v_2)$, $\bar{p_1}(v_1)$, $\bar{p_2}(v_2)$, $U_1(v_1)$ and $U_2(v_2)$.[34]

$$\bar{x}_1(v_1) = \int_{a_2}^{b_2} x_1(v_1, t_2) f_2(t_2)\,\mathrm{d}t_2,$$
$$\bar{x}_2(v_2) = \int_{a_1}^{b_1} x_2(t_1, v_2) f_1(t_1)\,\mathrm{d}t_1.$$

In addition, we let U_0 denote the expected net profit to the market-maker, such that

$$U_0 = \int_{a_2}^{b_2}\int_{a_1}^{b_1} \Big(x_2(t_1, t_2) - x_1(t_1, t_2)\Big) f_1(t_1) f_2(t_2)\,\mathrm{d}t_1\,\mathrm{d}t_2 \tag{28}$$

Neither buyer nor seller should ever expect to gain by lying about their valuation, and both traders should receive non-negative expected gains from trade in the mechanism. For any v_1 and v_2, let

$$C_1(v_1) = c_1(v_1, 1) = v_1 + \frac{F_1(v_1)}{f_1(v_1)},$$
$$C_2(v_2) = c_2(v_2, 1) = v_2 - \frac{1 - F_1(v_1)}{f_1(v_2)}.$$

With this we can formalize the following theorem:

Theorem 3. *For any individually rational mechanism with a market-maker,*

$$\begin{aligned} U_0 + U_1(b_1) + U_2(a_2) &= U_0 + \min_{v_1 \in [a_1, b_1]} U_1(v_1) + \min_{v_2 \in [a_2, b_2]} U_2(v_2) \\ &= \int_{a_2}^{b_2}\int_{a_1}^{b_1} \big(C_2(v_2) - C_1(v_1)\big) p(v_1, v_2)\, f_1(v_1)\, f_2(v_2)\,\mathrm{d}v_1\,\mathrm{d}v_2 \end{aligned} \tag{29}$$

Extending this result to any ex-post efficient mechanism with a market-maker,

$$U_0 + U_1(b_1) + U_2(a_2) = -\int_{a_2}^{b_1} (1 - F_2(t)) F_1(t)\mathrm{d}t.$$

Thus, the minimum expected subsidy required for the market-maker to achieve ex-post efficiency is,

$$\int_{a_2}^{b_1} (1 - F_2(t) F_1(t)\mathrm{d}t,$$

[34] Defined in the section "Trading with a Broker" in Myerson and Satterthwaite (1983)

even if the subsidy is not lump-sum.

Another interesting concept is the mechanism which maximizes the expected profit to the market-maker. Given the revealed valuations and equilibrium bids and offers for the buyer and seller, if the buyer and seller are restricted to trading through the market-maker then we can define the market-maker's optimal mechanism.

Theorem 4. *Suppose $C_1(\cdot)$ and $C_2(\cdot)$ are monotone increasing functions on $[a, b]$ and $[a_2, b_2]$, respectively. Then among all incentive-compatible, individually-rational mechanisms, the market maker's expected profit is maximized by a mechanism in which the object is transferred to the buyer if and only if $C_2(\tilde{V_2}) \geqslant C_1(\tilde{V_1})$.*

Theorem 5. *From Theorem 5 we get*

$$U_0 = \iint (C_2(v_2) - C_1(v_1))p(v_1, v_2)f_1(v_1)f_2(v_2)dv_1dv_2 - U_1(b_1) - U_2(a_2), \tag{30}$$

for any incentive-compatible mechanism. To maximize this expression, we want

$$p(v_1, v_2) = 1 \text{ if } C_2(v_2) \geqslant C_1(v_1) = 0 \text{ if } C_2(v_2) < C_1(v_1) \text{ and } U_1(b_1) = U_2(a_2) = 0. \tag{31}$$

It only remains to construct x_1 and x_2 such that (p, x_1, x_2) satisfies these conditions. If there is trade, then the market-maker charges the buyer the lowest possible valuation he could have quoted and still gotten the object (given the seller's valuation), and the broker pays the seller the highest valuation which he could have quoted and still sold the object (given the buyer's valuation). If there is no trade, then the market-maker receives no payments.

Remark: When comparing the mechanisms for a welfare maximizing and profit maximizing market-maker, there are always less trades under the profit maximizing mechanism.

6 Beyond Good and Evil: Restoring Market Efficiency

The theoretical model from Myerson and Satterthwaite (1983) provides insight into the efficiency gains from trading with a market-maker. However, the model in Myerson and Satterthwaite (1983) presents a similar problem to a real-world researcher as Rustichini et al. (1994), because both models assume that the buyer and seller draw v_b and v_s from a continuous distribution. We first adapt the model from Myerson and Satterthwaite (1983) to a "good" market-maker, who subsidizes the traders to facilitate transfer of the object and maximize the total welfare sum between the buyer and the seller. We then adapt the model to an "evil" market-maker, who uses their knowledge of the buyer and seller valuations and equilibrium bids and offers to profit from the spread between the price at which the buyer and seller with perfect information would trade, and their misrepresented bids and offers. Through this mechanism, the "evil" market-maker maximizes their own profit.

Before proceeding to the "good" and "evil" market-maker mechanisms, processed data must be imported as $\mathcal{D}_b$ and $\mathcal{D}_s$ from Section 4.5. In our case, the algorithm used to produce the results in

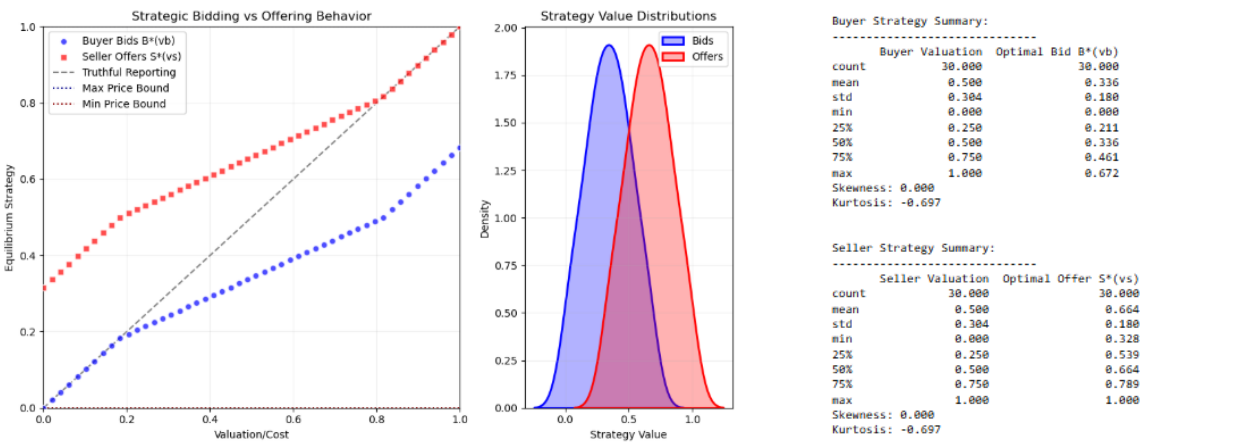
this section relies on the results from the algorithm outlined in Section 4. Therefore, we synthesize all relevant results for the researcher to verify before conducting additional analysis.

Method 10 (Equilibrium Data Importing)**.** *The import process validates:*

(a) Buyer and seller value ranges, $v_b, v_s \in [0, 1]$*;*
(b) Monotonicity of $B(v_b^{\text{proc}})$ *and* $S(v_s^{\text{proc}})$

The equations used to generate range validation visualizations are as follows.[35]

$$f_b(B) = \frac{1}{n_b h_b} \sum_{i=1}^{n_b} K\left(\frac{B - B_i^*}{h_b}\right), \quad f_s(S) = \frac{1}{n_s h_s} \sum_{j=1}^{n_s} K\left(\frac{S - S_j^*}{h_s}\right) \tag{32}$$



(a) Visualization of Equilibrium Strategies (b) Strategies Validation

Figure 8: Equilibrium Strategies, Visualization and Validation

Remark: Figure 8 gives a demonstration of the imported bids and values $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$ for all possible buyer and seller valuations v_b^{proc} and v_s^{proc} on [0,1]. Visualization in this manner is an important tool for verifying monotonicity of $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$.

6.1 A "Good" Market-Maker Mechanism

The "good" market-maker operates under the Myerson-Satterthwaite efficiency constraints, implementing subsidy mechanisms to maximize social welfare. We define $\zeta_{\min}$ as the optimal subsidy for the market-maker in the mechanism. Using $\zeta_{\min}$, the market-maker introduces the lowest subsidy necessary to achieve ex-post efficiency in the market once buyer and seller valuations v_b and v_s are known, and bids and offers $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$ have been revealed. We define that

[35]The fundamental strategic asymmetry is shown in the "Strategic Bidding vs Offering Behavior" graphic, while the "Strategic Value Distributions" graphic reveals strategic clustering patterns.

buyer valuations v_b^{proc} and v_s^{proc} are distributed on the intervals $F(v_b^{\text{proc}})$ and $G(v_s^{\text{proc}})$, respectively. We define $\bar{v}_b$ as the highest possible buyer valuation in his set of possible bids. This ensures that for a buyer and seller with equivalent valuations $v_b = v_s$, the buyer's bid $B(v_b)$ is always at least as big as the seller's offer $S(v_s)$.[36]

Theorem 6 (Myerson-Satterthwaite Subsidy Bound)**.** *For any bilateral trading environment with:*

(a) Buyer valuations $v_b^{\text{proc}} \sim F(v_b^{\text{proc}})$
(b) Seller valuations $v_s^{\text{proc}} \sim G(v_s^{\text{proc}})$
(c) Strategic bids $B(v_b^{\text{proc}})$ *and offers* $S(v_s^{\text{proc}})$

For $\bar{v}_b = \sup\{v_b^{\text{proc}} \in [0,1] : B^*(v_b^{\text{proc}}) \geqslant S^*(v_s^{\text{proc}})\}$*, the minimal required subsidy* $\zeta_{\min}$ *to achieve ex-post efficiency satisfies:*

$$\zeta_{\min} = \int_{v_s}^{\bar{v}_b} \int_{0}^{v_b} \max(0, S(v_s^{\text{proc}}) - B(v_b^{\text{proc}}))\, \mathrm{d}F(v_b^{\text{proc}}) \mathrm{d}G(v_s^{\text{proc}}) \tag{33}$$

We define for the market-maker, candidate pairs $\mathcal{C}$ which are the instances for which the buyer has a valuation v_b that is higher than the seller's valuation v_s, (ie. $v_b > v_s$). It is for these pairs that the market-maker can subsidize the traders to a market-clearing price in the mechanism. Bifurcate the set of candidate pairs $\mathcal{C}$ into natural candidate pairs $\mathcal{C}_{natural}$ and subsidized candidate pairs $\mathcal{C}_{subsidized}$, where natural candidate pairs are the instances where $v_b > v_s$ without the need of a subsidy, and the subsidized candidate pairs only achieve $v_b > v_s$ after intervention by the market-maker. Define by $\mathcal{S}$ the total amount of subsidy given to the traders across all matched candidate pairs by the market-maker in the mechanism.

Method 11 (Efficiency-Maximizing Matching)**.** *The subsidy-optimal matching process proceeds as:*

(a) Generate candidate pairs $\mathcal{C} = \{(i,j) | v_b^i \geqslant v_s^j\}$
(b) Compute required subsidies:

$$\zeta_{ij} = \max(0, S_j^* - B_i^*)$$

(c) Partition candidates:

$$\mathcal{C}_{natural} = \{(i,j) \in \mathcal{C} | B_i^* \geqslant S_j^*\}, \quad \mathcal{C}_{subsidized} = \mathcal{C} \backslash \mathcal{C}_{natural}$$

Let buyers' processed valuations and bids be

$$\{(v_{b,i},\, B_i)\}_{i=1}^{m},$$

and sellers' processed valuations and offers be

$$\{(v_{s,j},\, S_j)\}_{j=1}^{n}.$$

[36] Reference Appendix F Figure 1 for the formal algorithm for a "good" market-maker mechanism

Define the candidate set

$$\mathcal{C} = \big\{(i,j)\colon v_{b,i} \geqslant v_{s,j}\big\}, \quad s_{ij} = \max\{0,\ S_j - B_i\}.$$

Partition into

$$\mathcal{C}_{\text{natural}} = \big\{(i,j) \in \mathcal{C}\colon B_i \geqslant S_j\big\}, \quad \mathcal{C}_{\text{subsidized}} = \mathcal{C} \backslash \mathcal{C}_{\text{natural}},$$

and sort $\mathcal{C}_{\text{subsidized}}$ *in ascending order of* $\int_{ij}$.
Initialize empty match set P *and used-sets* U_b, U_s. *Then perform greedy matching:*

$$P = \big\{(i,j) \in \mathcal{C}_{\text{natural}} \ \cup \ sort(\mathcal{C}_{\text{subsidized}}) \ \big|\ i \notin U_b,\ j \notin U_s\big\},$$

where each selected (i,j) *is appended to* P *and* i *is added to* U_b, j *to* U_s. *The total subsidy disbursed is*

$$S_{\text{total}} = \sum_{(i,j)\in P} \int_{ij}.$$

```
"Good" Market Maker Results:
======================================================
Total Trades: 26
Total Subsidy: 1.582

First 10 Trades:
------------------------------------------------------
Trade 1: B(v=0.469, B=0.327) ↔ S(c=0.000, S=0.316)
Price: 0.321 | Subsidy: 0.000

Trade 2: B(v=0.510, B=0.347) ↔ S(c=0.020, S=0.337)
Price: 0.342 | Subsidy: 0.000

Trade 3: B(v=0.551, B=0.367) ↔ S(c=0.041, S=0.357)
Price: 0.362 | Subsidy: 0.000

Trade 4: B(v=0.592, B=0.388) ↔ S(c=0.061, S=0.378)
Price: 0.383 | Subsidy: 0.000

Trade 5: B(v=0.633, B=0.408) ↔ S(c=0.082, S=0.398)
Price: 0.403 | Subsidy: 0.000

Trade 6: B(v=0.673, B=0.429) ↔ S(c=0.102, S=0.418)
Price: 0.423 | Subsidy: 0.000

Trade 7: B(v=0.714, B=0.449) ↔ S(c=0.122, S=0.439)
Price: 0.444 | Subsidy: 0.000

Trade 8: B(v=0.755, B=0.469) ↔ S(c=0.143, S=0.459)
Price: 0.464 | Subsidy: 0.000

Trade 9: B(v=0.776, B=0.480) ↔ S(c=0.163, S=0.480)
Price: 0.480 | Subsidy: 0.000

Trade 10: B(v=0.837, B=0.520) ↔ S(c=0.184, S=0.500)
Price: 0.510 | Subsidy: 0.000
```

(a) First 10 Trades

```
Last 10 Trades:
------------------------------------------------------
Trade 12: B(v=0.980, B=0.663) ↔ S(c=0.327, S=0.571)
Price: 0.617 | Subsidy: 0.000

Trade 13: B(v=1.000, B=0.684) ↔ S(c=0.347, S=0.582)
Price: 0.633 | Subsidy: 0.000

Trade 14: B(v=0.816, B=0.500) ↔ S(c=0.367, S=0.592)
Price: 0.546 | Subsidy: 0.092

Trade 15: B(v=0.796, B=0.490) ↔ S(c=0.388, S=0.602)
Price: 0.546 | Subsidy: 0.112

Trade 16: B(v=0.735, B=0.459) ↔ S(c=0.408, S=0.612)
Price: 0.536 | Subsidy: 0.153

Trade 17: B(v=0.694, B=0.439) ↔ S(c=0.429, S=0.622)
Price: 0.531 | Subsidy: 0.184

Trade 18: B(v=0.653, B=0.418) ↔ S(c=0.449, S=0.633)
Price: 0.526 | Subsidy: 0.214

Trade 19: B(v=0.612, B=0.398) ↔ S(c=0.469, S=0.643)
Price: 0.520 | Subsidy: 0.245

Trade 20: B(v=0.571, B=0.378) ↔ S(c=0.490, S=0.653)
Price: 0.515 | Subsidy: 0.276

Trade 21: B(v=0.531, B=0.357) ↔ S(c=0.510, S=0.663)
Price: 0.510 | Subsidy: 0.306

======================================================
```

(b) Last 10 Trades

Figure 9: First 10 and Last 10 trades for the "Good" Market-Making Entity

Remark: Figure 9 shows the "First 10" trades and the "Last 10" trades within the "good" market-maker mechanism. In accordance with Method 11, the trades which are subsidized by the market-maker are sorted, and listed in ascending order according to the total subsidy required for

the market to clear between a candidate pair $\mathcal{C}$. Figure 9 lists the total number of trades achieved within the mechanism, and also the total subsidy disbursed by the market-maker, $\mathcal{S}_{total}$. It is important to note that in this case, only 20 trades could be achieved in the mechanism, so all 20 trades are listed in Figure 9. This is not likely to be the case when the buyer and seller have more than 30 possible valuations.

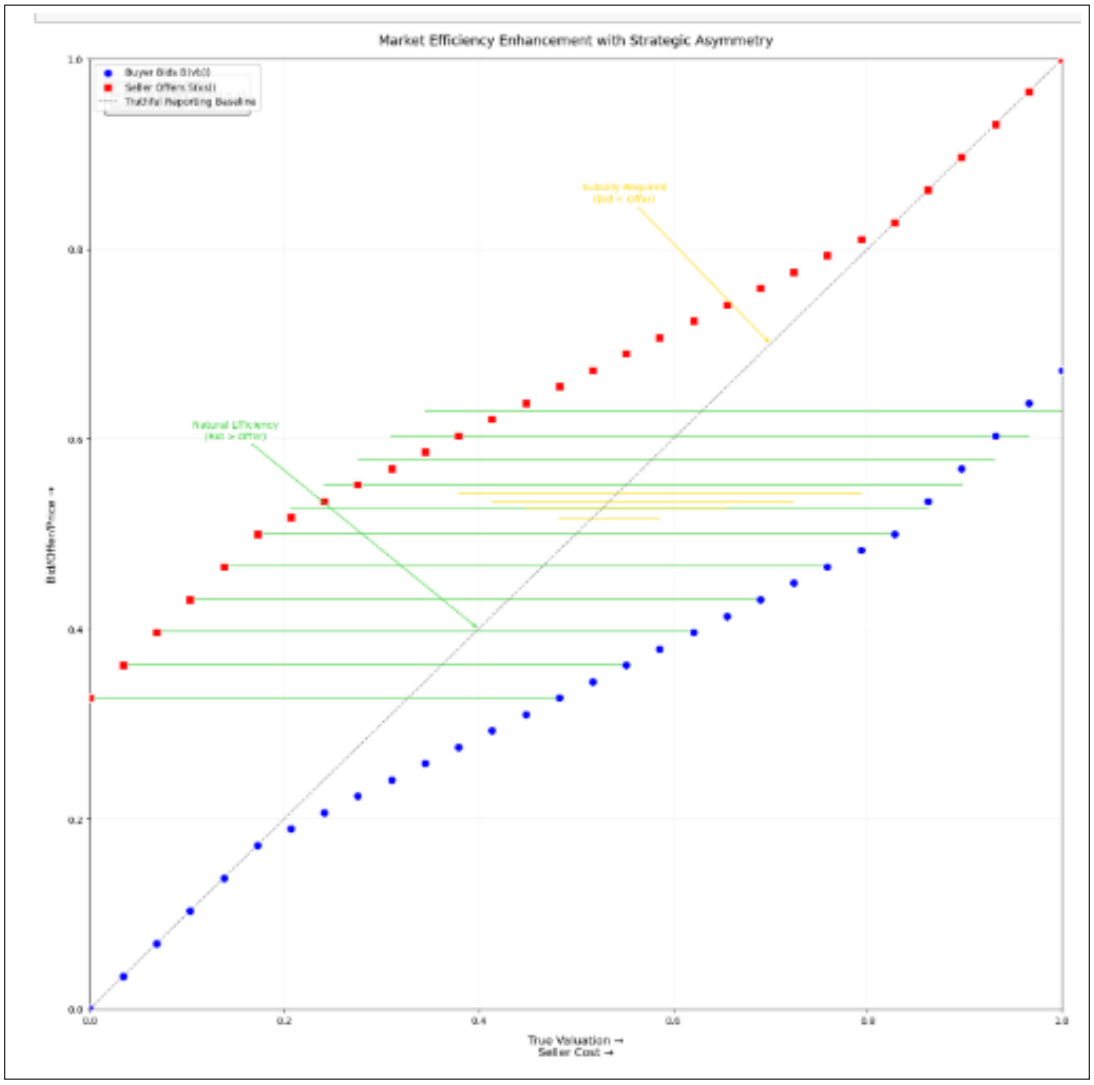


Figure 10: Visualization of the "Good" Market Maker

Remark: Figure 10 generates a visualization of the matched trades occurring over all bids and offers $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$ on the interval $[0, 1]$. It is a straight-forward and useful tool for analyzing the effectiveness of the "good" market-maker mechanism, and for ensuring no "bugs" have occured in the algorithm during the candidate pair matching process.

6.2 An "Evil" Market-Maker Mechanism

The "evil" market-maker uses their knowledge of equilibrium bids and offers, and the distribution of buyer and seller valuations to impose a *manipulation factor* on the buyer and seller, denoted by α. Here, α is used to present the buyer and the seller with the furthest possible price from their true valuations at which they still trade. The "evil" market-maker then maximizes its profits from this spread. We define for the "evil" market-maker a π which signifies the "evil" market-maker's profit from manipulation of an individual trade. Denote for the "evil" market-maker, the value Π which is the total profit they obtain within the mechanism from all trades. We use the same notation to denote empty sets P and used sets U_i as in the previous section.[37]

Method 12 (Strategic Matching Process)**.** *Let adjusted bids/offers be*

$$\tilde{B}_i = \min\big(B_i + \alpha v_b^i,\ v_b^i\big), \quad \tilde{S}_j = \max\big(S_j - \alpha v_s^j,\ v_s^j\big).$$

Define the net payoff for each feasible pair:

$$\pi_{ij} = \begin{cases} \tilde{B}_i - \tilde{S}_j, & \tilde{B}_i \geqslant \tilde{S}_j, \\ -\infty, & \textit{otherwise.} \end{cases}$$

(a) ***Candidate ranking.*** *Form*

$$\mathcal{C} = \big\{(i,j) \colon \pi_{ij} > -\infty\big\},$$

and sort $\mathcal{C}$ so that $\pi_{i_1 j_1} \geqslant \pi_{i_2 j_2} \geqslant \cdots$.

(b) ***Greedy matching.*** *Initialize $U_b = U_s = P = \varnothing$. Then*

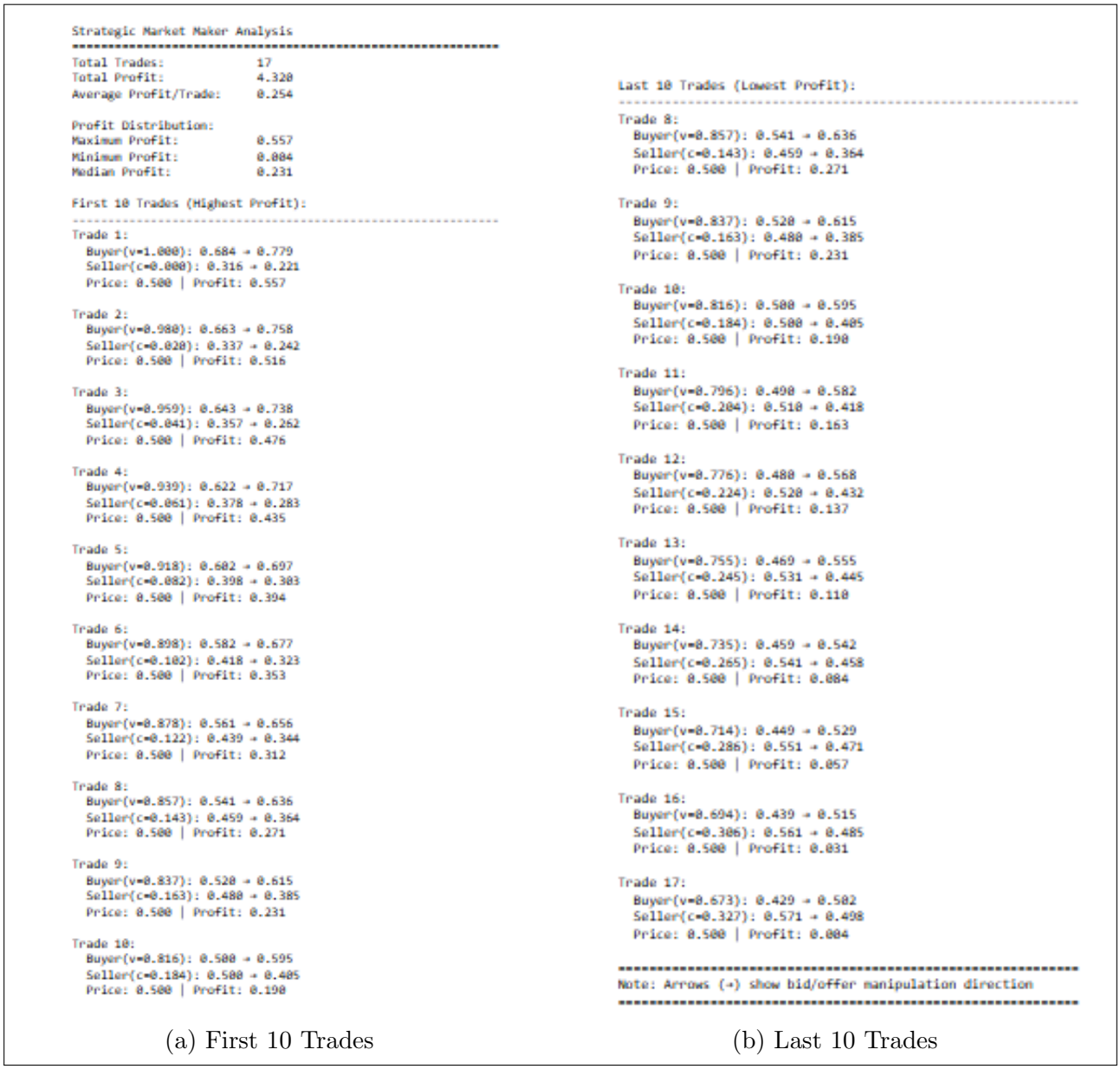
$$P \ = \ \big\{(i,j) \in \mathcal{C} : i \notin U_b,\ j \notin U_s\big\},$$

where each selected (i,j) is appended to P and we update $U_b \leftarrow U_b \cup \{i\}$, $U_s \leftarrow U_s \cup \{j\}$.

(c) ***Total profit.*** *Finally,*

$$\Pi = \sum_{(i,j)\in P} \pi_{ij}.$$

[37] Reference Appendix G Figure 1 for the formal algorithm for an "evil" market-maker mechanism



(a) First 10 Trades (b) Last 10 Trades

Figure 11: First 10 and Last 10 trades for the "Evil" Market-Making Entity

Remark: Figure 11 displays the "First 10" and "Last 10" trades achieved within the "evil" market-maker mechanism. As dictated by Method 12, trades are sorted in descending order for the "evil" market-maker according to the profit π_i obtained from that trade. Note, that here there are not 20 total trades to showcase, so some trades are shown multiple times. This is not likely to be the case when the buyer and seller have more than 30 possible valuations. Notice that the number of total trades in the "evil" market-maker mechanism is 17, this is less than the number of trades in the "good" market-maker mechanism for the same number of possible buyer and seller values. This aligns with the theoretical result from the end of Section 5.

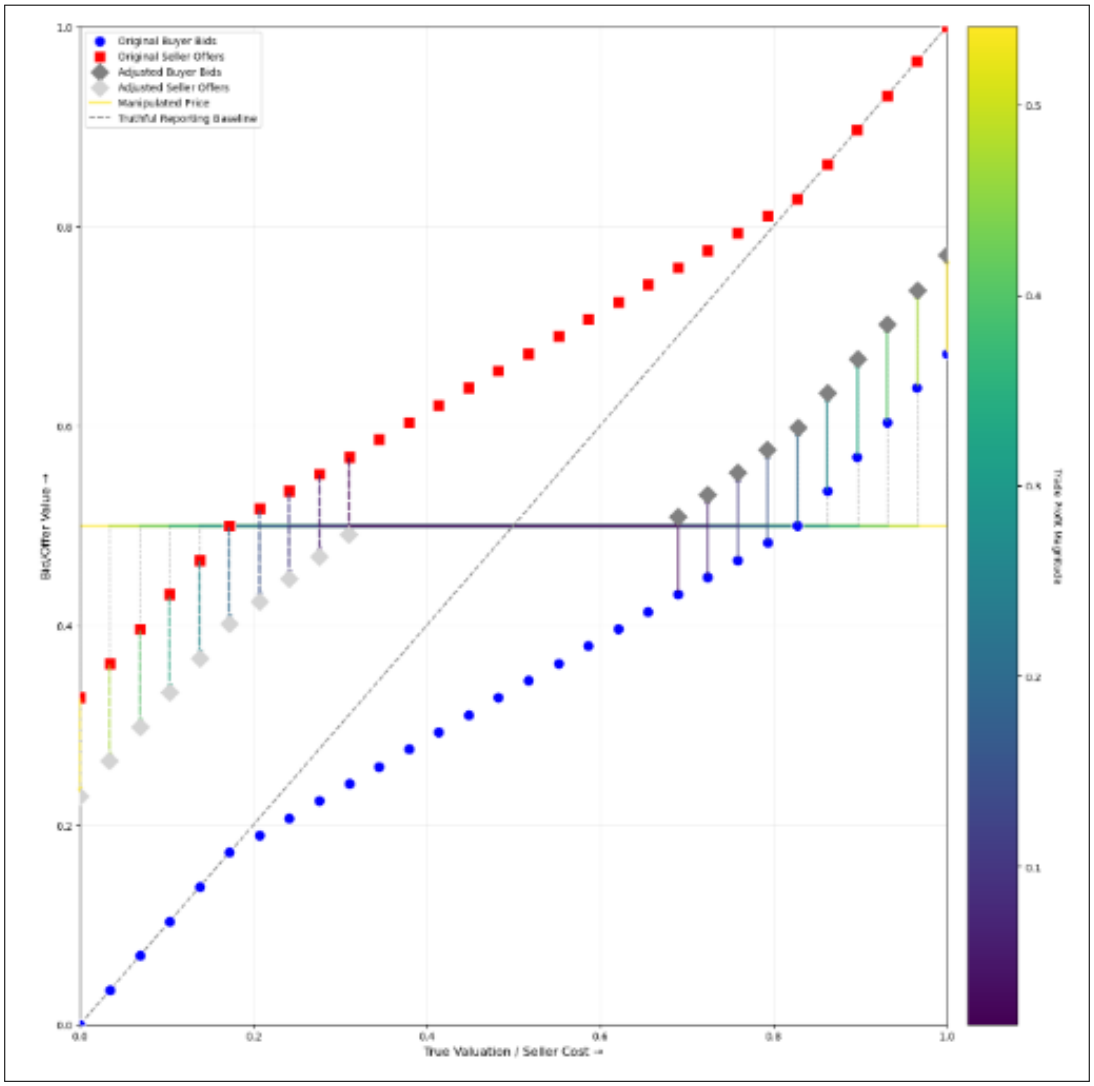


Figure 12: Visualization of the "Evil" Market Maker

Remark: Remark: Figure 12 generates a visualization of the manipulated trades occurring over all bids and offers $B(v_b^{\text{proc}})$ and $S(v_s^{\text{proc}})$ on the interval $[0, 1]$. It is a straight-forward and useful tool for analyzing the effectiveness of the "evil" market-maker mechanism, and for ensurimg no "bugs" have occured in the algorithm during the candidate pair matching process.

7 Conclusion

Theoretical models for understanding trade in the k-Double Auction are not relevant to the real world because a buyer and seller in a given market do not possess continuously distributed values for an object. Instead, we have proposed a comprehensive mechanism capable of adapting robust data for a large class of real-world scenarios to models employing continuous distributions. We first showed in Section 4, that with a combination of adaptive bandwidth adjustments and jittering at the boundaries of a given trading interval, we can transform a discrete valuation grid into a

continuous set of possible values for which differentiation is possible. Following this, we succeeded in recreating the results from Rustichini et al. (1994) using our computed distributions, proving the validity of our transformation methods. Our model for the buyer's first-order-conditions, and symmetric model for the seller's first-order-conditions succeeded in recreating the fundamental result that equilibrium in a market for an object is generally impossible when traders possess private information. We succeeded in showcasing that although the model violates the first-order inequality conditions for the buyer and the seller when the number of possible values in the discrete valuation grid is relatively small, in cases where the buyer and seller draw from many possible discrete valuations inefficiency decreases.[38] We succeeded in showing that in our model, the buyer and seller satisfy the individual rationality constraint, where neither trader ever trades at a price where their expected payoff is less than zero (the payoff received from non-participation in the mechanism). Finally, we succeeded in showing that the model from Myerson and Satterthwaite (1983) for a market-maker mechanism can be adapted to the case of bilateral trade between a buyer and a seller with discrete valuations. Our market-maker mechanism also succeeded in reproducing the fundamental result that there are strictly more trades in the welfare maximizing "good" market-maker mechanism, than in the "evil" market-maker mechanism. The results from this analysis are encouraging, and demonstrate that our novel mechanism may be capable of adapting other models with the continuous values distributions assumption, to recreate their fundamental results.

[38]Reference the link to the GitHub repository in Appendix H for access to examples of the model with many possible valuations.

8 Appendix A: Formal Algorithms from Section 4.1

```
# ------------------------------------------------------------------------------------
# Appendix of Libraries/Packages in Order of Use: Equilibrium Bids and Offers Algorithm
# ------------------------------------------------------------------------------------

from ipywidgets import widgets, VBox, Button, Output
from IPython.display import display
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde
from scipy.optimize import root_scalar
from scipy.integrate import quad
from scipy.interpolate import interp1d
import warnings
from scipy.integrate import IntegrationWarning
warnings.simplefilter("ignore", IntegrationWarning)
import pandas as pd
import pickle
from pathlib import Path
```

(a) Appendix of Libraries/Packages Used in the Equilibrium Bids and Offers Algorithm

```
# ------------------------------------------------------------------------------------
# Appendix of Libraries/Packages in Order of Use: Market-Maker Mechanism Algorithm
# ------------------------------------------------------------------------------------

import pickle
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

(b) Appendix of Libraries/Packages Used in the Market-Maker Algorithm

Appendix A Figure 1: Appendix of Packages/Libraries Used in Both Algorithms

```python
# ------------------------------------------------------------------------------------------
# Data Generating Process for the k-DA (Section 4.1: The Model, and Data Generating Process)
# Generate vb, vs, k, prices, v_lower, c_upper
# ------------------------------------------------------------------------------------------

from ipywidgets import widgets, VBox, Button, Output
from IPython.display import display
import numpy as np

vb, vs = [], []
prices = (0.0, 1.0)
k = 0.5
v_lower = None
c_upper = None

# ----------------------------------------------------------------
# Sorting Buyer and Seller Valuations According to N (num_vals)
# ----------------------------------------------------------------

def generate_values(num_vals):
    return np.linspace(0, 1, num_vals).tolist()
```

(a) Data Generation Process for the Mechanism Parameters

```python
# ------------------------------------------------
# Print Parameter Selection Mechanism (Figure 1)
# ------------------------------------------------

num_vals_widget = widgets.IntSlider(
    value=5, min=5, max=1000, step=1, description='Num Values:'
)
k_slider = widgets.FloatSlider(value=k, min=0, max=1, step=0.001, description='k:')
print("MUST CLICK CALCULATE BUTTON AFTER SETTING PARAMETERS TO RECEIVE VALUES")
print("FOR HIGH OR LOW K (ie k >= .75 or k <= .25) MUST ALSO HAVE SUFFICIENT NUMBER OF POSSIBLE VALUATIONS")
calculate_button = Button(description="Calculate")
calculate_button.on_click(calculate)

default_ui = VBox([num_vals_widget, k_slider, calculate_button, default_output])
display(default_ui)
```

(b) Data Generation Process for the Serious Trading Interval

Appendix A Figure 2: Data Generations Process for Mechanism Parameters and Trading Interval

```python
def calculate(b):
    default_output.clear_output()
    num_vals = num_vals_widget.value
    k_value = k_slider.value
    vs, vb, prices = get_game(num_vals)

    global v_lower, c_upper

    # ----------------------------------------
    # Algorithmic Formalization of Definiton 2
    # ----------------------------------------

    E = [v for v in vb if 0.1 <= v <= 0.25]
    v_lower = np.random.choice(E) if E else 0.173

    # ----------------------------
    # Explicit Symmetry of c_upper
    # ----------------------------

    c_upper = 1 - v_lower

    with default_output:
        print(f"Number of Possible Values: {num_vals}, k: {k_value:.3f}")
        print()
        print(f"Buyers: {[f'{v:.5f}' for v in vb]}")
        print(f"Sellers: {[f'{c:.5f}' for c in vs]}")
        print()
        print(f"Trading Range: [{v_lower:.5f}, {c_upper:.5f}]")
```

Appendix A Figure 3: Generation Process for $\underline{v}$ and $\bar{c}$

```
# ------------------------------------------------------------------
# Calculation of "Serious" Intervals for Buyer and Seller bids/offers
# ------------------------------------------------------------------

global vb_serious_interval, vs_serious_interval, serious_values_interval

# ------------------------------------------------------------------------------------------
# Algorithmic Formalization of the Contraints in Defintion 3 (Possible Bid/Offer Intervals)
# ------------------------------------------------------------------------------------------

vb_serious_interval = (v_lower, 1.0)
vs_serious_interval = (0.0, c_upper)
serious_values_interval = (v_lower, c_upper)

# ------------------
# v_lower in [0, 1]
# ------------------

def buyer_possible_bids(v):
    """For a buyer with valuation v, return bidding interval or message"""
    if v < vb_serious_interval[0]:
        return "not in serious bids (below threshold)"

    interval = (vb_serious_interval[0], v)
    if interval[0] == interval[1]:
        return f"[{interval[0]:.5f}] (Only Possible Bid)"
    return interval

# ------------------
# c_upper in [0, 1]
# ------------------

def seller_possible_offers(v):
    """For a seller with valuation v, return offering interval or message"""
    if v > serious_values_interval[1]:
        return "not in serious offers (above threshold)"

    interval = (v, vs_serious_interval[1])
    if interval[0] == interval[1]:
        return f"[{interval[0]:.5f}] (Only Possible Offer)"
    return interval

# ------------------------------------------------------------------
# Restrict "Serious" Bids and Offers for Valuations in vb and vs
# ------------------------------------------------------------------

possible_bids = {v: buyer_possible_bids(v) for v in vb}
possible_offers = {v: seller_possible_offers(v) for v in vs}
```

Appendix A Figure 4: Generation Process for the Serious Trading Intervals

9 Appendix B: Formal Algorithms from Section 4.2

```python
# ------------------------------------------------------------------------------
# Algorithmic Formalization of Method 1 (Adaptive Bandwidth for Bounded Support)
# ------------------------------------------------------------------------------

def adaptive_bandwidth(data, data_range=(0,1)):
    """Silverman's rule adapted for bounded support"""
    n = len(data)
    range_width = data_range[1] - data_range[0]

    if n < 2:                       # ----------------------------
        return 0.1 * range_width    # Base minimum on range width
                                    # ----------------------------

    # ---------------------------------------------------------
    # Silverman's Normal Reference Rule with Reflection Adjustment
    # ---------------------------------------------------------

    sigma = np.std(data, ddof=1)
    reflection_factor = 1 + (sigma**2)/(range_width**2)
    bw = 1.06 * sigma * (n ** (-0.2)) * (reflection_factor ** 0.2)

    # ---------------------------------------------
    # Bandwidth Constraints to Avoid Oversmoothing
    # ---------------------------------------------

    return np.clip(bw, 0.05 * range_width, 0.5 * range_width)
```

(a) Algorithm For Adaptive Bandwidth for Bounded Support

```python
# ----------------------
# Bandwidth Calculations
# ----------------------

bw_vb = adaptive_bandwidth(vb_processed, (0,1))
bw_vs = adaptive_bandwidth(vs_processed, (0,1))
bw_vb_serious = adaptive_bandwidth(vb_serious_processed, (v_lower, c_upper))
bw_vs_serious = adaptive_bandwidth(vs_serious_processed, (v_lower, c_upper))
bw_serious = adaptive_bandwidth(serious_processed, (v_lower, c_upper))
```

(b) Calculation of the Adaptive Bandwidths

Appendix B Figure 1: Adaptive Bandwidth for Bounded Support and Calculation Function

```python
# ------------------------------------------------------------
# Algorithmic Formalization of Method 2 (Boundary Jittering)
# ------------------------------------------------------------

def preprocess_valuations(values, epsilon=0.005):
    """Uniform-optimized jittering"""
    processed = []
    for v in values:
        if np.isclose(v, 0) or np.isclose(v, 1):
            processed.append(v)
        else:                                                  # --------------------------
            jitter = np.random.uniform(-epsilon, epsilon)  # epsilon ~ (-delta, delta)
            processed.append(np.clip(v + jitter, 0, 1))    # --------------------------
    return np.array(processed)
```

Appendix B Figure 2: Algorithm for Boundary Jittering

```python
# ------------------------------------------------------------------
# Algorithmic Formalization of Defintion 5 (Empirical PDF and CDF)
# ------------------------------------------------------------------

def compute_distribution(values):
    """Calculate discrete PDF/CDF for values"""
    unique_vals, counts = np.unique(values, return_counts=True)

    # -----------------------------------
    # Empirical Probability Mass Function
    # -----------------------------------

    pdf = counts / counts.sum()

    # -------------------------------------------
    # Empirical Cumulative Distribution Function
    # -------------------------------------------

    cdf = np.cumsum(pdf)
    return unique_vals, pdf, cdf
```

(a) Algorithm For Empirical PDF and CDF

```python
# ------------------------------------------------------------------------------
# Algorithmic Formalization of Method 3 (Smoothed PDF and CDF via Bounded KDE)
# ------------------------------------------------------------------------------

def compute_smoothed_distribution(values, bandwidth, data_range=(0,1)):
    """Calculate KDE-smoothed PDF/CDF with proper range handling"""
    if len(values) < 2:
        return None, None, None
    kde = gaussian_kde(values, bw_method=bandwidth)
    x_smooth = np.linspace(data_range[0], data_range[1], 200)
    pdf_smooth = kde.evaluate(x_smooth)

    # ---------------------------------------------------------------
    # Normalize PDF to Integrate to 1 Over the Specified Range [l, u]
    # ---------------------------------------------------------------

    pdf_smooth /= np.trapz(pdf_smooth, x_smooth)
    cdf_smooth = np.cumsum(pdf_smooth)
    cdf_smooth /= cdf_smooth[-1]  # Ensure CDF ends at 1
    return x_smooth, pdf_smooth, cdf_smooth
```

(b) Algorithm for Smoothed PDF and CDF via Bounded KDE

Appendix B Figure 3: Algorithm for the Empirical and Smoothed PDFs and CDFs

10 Appendix C: Formal Algorithms from Section 4.3

```python
# ------------------------------------------------------------------------
# Algorithmic Formalization of k-DA Baseline Strategies in Definition 7
# ------------------------------------------------------------------------

def theoretical_buyer_bid(v):
    """From RWS Equation (3.1)"""
    return v_lower + (v - v_lower) * (1 - k)

def theoretical_seller_offer(c):
    """From RWS Equation (3.2)"""
    return c_upper - (c_upper - c) * (1 - k)
```

Appendix C Figure 1: Algorithm for Theoretical Baseline Strategies

```python
# ------------------------------------------------------------------------------
# Algorithmic Formalization of the Numerical Probability Integration in Definition 8
# ------------------------------------------------------------------------------

def log_K_nn(lam, m, n):
    """Log probability ≥n sellers bid ≥λ using numerical integration"""
    try:
        # ------------------------------------------------
        # Get valid integration bounds from KDE results
        # ------------------------------------------------

        min_c = global_kde_results['seller_domain'][0]
        max_c = global_kde_results['seller_domain'][1]
        lower = max(0.0, min_c)
        upper = min(lam, max_c)
        integ = quad(lambda x: seller_pdf_func(x), lower, upper)[0]
        return n * np.log(max(integ, 1e-100))
    except:
        return -np.inf

def log_L_nn(lam, m, n):
    """Log probability ≥m-1 buyers bid ≥λ using numerical integration"""
    try:
        # ------------------------------------------------
        # Get valid integration bounds from KDE results
        # ------------------------------------------------

        min_v = global_kde_results['buyer_domain'][0]
        max_v = global_kde_results['buyer_domain'][1]
        lower = max(lam, min_v)
        upper = min(1.0, max_v)
        integ = quad(lambda x: buyer_pdf_func(x), lower, upper)[0]
        return (m-1) * np.log(max(integ, 1e-100))
    except:
        return -np.inf

def log_M_nn(lam, m, n):
    """Combined numerical integration with domain awareness"""
    try:
        # -------------------------
        # Buyer survival component
        # -------------------------

        b_min = global_kde_results['buyer_domain'][0]
        b_max = global_kde_results['buyer_domain'][1]
        buyer_survival = quad(lambda x: buyer_pdf_func(x),
                              max(lam, b_min),
                              min(1.0, b_max))[0]

        # ---------------------
        # Seller CDF component
        # ---------------------

        s_min = global_kde_results['seller_domain'][0]
        s_max = global_kde_results['seller_domain'][1]
        seller_cdf = quad(lambda x: seller_pdf_func(x),
                          max(0.0, s_min),
                          min(lam, s_max))[0]

        return (m-1)*np.log(max(buyer_survival, 1e-100)) + n*np.log(max(seller_cdf, 1e-100))
    except:
        return -np.inf
```

Appendix C Figure 2: Algorithm for Numerical Probability Integration

```
# ---------------------------------------------------
# Parameters for Complex-Step Derivative Approximation
# ---------------------------------------------------

                        # ----------------
SAFE_DERIVATIVE = 1e-8  # delta = 10^(-8)
                        # ----------------

                        # --------------------------------------
DERIV_EPS = 1e-20       # epsilon = 10^(-20): Complex step size
                        # --------------------------------------

# ------------------------------------------------------------------------------------------
# Algorithmic Formalization of the Complex-Step Derivative Approximation in Definition 9
# ------------------------------------------------------------------------------------------

def complex_step_derivative(func, x):
    deriv = np.imag(func(x + DERIV_EPS*1j)) / DERIV_EPS
    return np.clip(deriv, SAFE_DERIVATIVE, None) if deriv > 0 else np.clip(deriv, None, -SAFE_DERIVATIVE)

# ---------------------------------------------
# Stabilized PDF Extension (KDE Interpolation)
# ---------------------------------------------

buyer_pdf_func = interp1d(
    global_kde_results['serious_buyer_x'],     # ------------------------------------
    global_kde_results['serious_buyer_pdf'],   # Linear Interpolation of Buyer Values
    bounds_error=False,                        # ------------------------------------

                                # ----------------------------------
    fill_value=(1e-100, 1e-100)   # Definiton of varepsilon = 10^(-100)
)                               # ----------------------------------

seller_pdf_func = interp1d(
    global_kde_results['serious_seller_x'],    # -------------------------------------
    global_kde_results['serious_seller_pdf'],  # Linear Interpolation of Seller Values
    bounds_error=False,                        # -------------------------------------
    fill_value=(1e-100, 1e-100)
)
```

Appendix C Figure 3: Algorithm for Complex-Step Derivative Approximation

```
# ---------------------------------------------
# Stabilized PDF Extension (KDE Interpolation)
# ---------------------------------------------

buyer_pdf_func = interp1d(
    global_kde_results['serious_buyer_x'],     # ------------------------------------
    global_kde_results['serious_buyer_pdf'],   # Linear Interpolation of Buyer Values
    bounds_error=False,                        # ------------------------------------

                                # ----------------------------------
    fill_value=(1e-100, 1e-100)   # Definiton of varepsilon = 10^(-100)
)                               # ----------------------------------

seller_pdf_func = interp1d(
    global_kde_results['serious_seller_x'],    # -------------------------------------
    global_kde_results['serious_seller_pdf'],  # Linear Interpolation of Seller Values
    bounds_error=False,                        # -------------------------------------
    fill_value=(1e-100, 1e-100)
)
```

Appendix C Figure 4: Algorithm for Stabilized PDF Extension

```
# ----------------------------------------------------
# Stabilized First-Order-Condition Solver from Method 4
# ----------------------------------------------------

                # ----------------------------------------------------
LOG_CLAMP = 700  # Log-clamping Parameter from Remark 2, and Elsewhere
                # ----------------------------------------------------

# ------------------------------------------
# Buyer-Equilibrium Computation from Method 4
# ------------------------------------------

def buyer_foc(lam, v, m, n):
    if lam < v_lower or lam > c_upper:
        return np.inf if lam < v_lower else -np.inf

    try:
        f = max(seller_pdf_func(lam), 1e-100)
        g = max(buyer_pdf_func(v), 1e-100)

        dB = complex_step_derivative(theoretical_buyer_bid, v)
        dS = complex_step_derivative(theoretical_seller_offer, lam)

        # ------------------------------------------------------------------------
        # Log-space calculations with overflow protection from Equations (15)-(17)
        # ------------------------------------------------------------------------

        log_K = np.clip(log_K_nm(lam, m, n), -LOG_CLAMP, LOG_CLAMP)
        log_L = np.clip(log_L_nm(lam, m, n), -LOG_CLAMP, LOG_CLAMP)
        log_M = np.clip(log_M_nm(lam, m, n), -LOG_CLAMP, LOG_CLAMP)

        K = np.exp(log_K) if abs(log_K) < LOG_CLAMP else 0.0
        L = np.exp(log_L) if abs(log_L) < LOG_CLAMP else 0.0
        M = np.exp(log_M) if abs(log_M) < LOG_CLAMP else 0.0

        # ----------------------------------------
        # Core FOC Calculation [Exact Equation (15)]
        # ----------------------------------------

        term1 = (v - k) * (n * K * f / max(abs(dS), SAFE_DERIVATIVE) +
                           (m-1) * L * g / max(abs(dB), SAFE_DERIVATIVE))
        term2 = k * M
        return term1 - term2
    except Exception as e:
        print(f"FOC error at v={v:.4f}, λ={lam:.4f}: {str(e)}")
        return np.inf
```

Appendix C Figure 5: Algorithm for Stabilized First-Order Conditions Solver and Buyer Equilibrium Computation Part (a)

```python
# ---------------------------------------------------------------------
# 4. Buyer Equilibrium Calculation with Safe Brackets from Method 5
# ---------------------------------------------------------------------

def calculate_buyer_equilibrium():
    equilibrium = {}
    m = len(vb_serious_processed)
    n = len(vs_serious_processed)

    bid_min = v_lower
    bid_max = c_upper

    buyer_warning_printed = False

    for v in vb_serious_processed:
        try:
            theory_bid = theoretical_buyer_bid(v)
            safe_bracket = (
                max(bid_min, theory_bid - 0.1, v_lower),
                min(bid_max, theory_bid + 0.1, v)
            )
            if safe_bracket[0] >= safe_bracket[1]:
                safe_bracket = (bid_min, min(bid_max, v))

            # ---------------------------------------------
            # Use of the Robust 1-D Solver: Brent's Method
            # ---------------------------------------------

            sol = root_scalar(
                buyer_foc,
                args=(v, m, n),
                bracket=safe_bracket,
                method='brentq',
                xtol=1e-6,
                rtol=1e-5
            )

            equilibrium[v] = np.clip(sol.root, bid_min, bid_max)
        except Exception as e:
            if not buyer_warning_printed:
                print(f"Using theoretical bid for v={v:.4f}: {str(e)}")
                buyer_warning_printed = True
            equilibrium[v] = np.clip(theory_bid, bid_min, bid_max)

    return equilibrium

buyer_eq = calculate_buyer_equilibrium()
```

Appendix C Figure 6: Algorithm for Buyer Equilibrium Calculation with “Safe” Brackets

```python
# ------------------------------------------------------------------------------------
# Algorithmic Formalization of the Extension to "all valuations" from (b) in Method 5
# ------------------------------------------------------------------------------------

def extend_buyer_equilibrium(full_valuations, eq_dict, v_lower, c_upper):

    extended_eq = {}

    # ------------------------------------------------------------------------------------
    # We assume eq_dict[v_lower] and eq_dict[c_upper] exist if they're in the serious set
    # If not, we can clamp or handle them gracefully.
    # ------------------------------------------------------------------------------------

    B_v_lower = eq_dict.get(v_lower, theoretical_buyer_bid(v_lower))
    B_c_upper = eq_dict.get(c_upper, theoretical_buyer_bid(c_upper))

    for v in full_valuations:
        if v < v_lower:

            # ------------------------------------------
            # Project downward from v_lower with slope 1
            # ------------------------------------------

            extended_eq[v] = B_v_lower - (v_lower - v)
        elif v > c_upper:

            # ------------------------------------------
            # Project upward from c_upper with slope 1
            # ------------------------------------------

            extended_eq[v] = B_c_upper + (v - c_upper)
        else:
            if v in eq_dict:
                extended_eq[v] = eq_dict[v]
            else:
                extended_eq[v] = theoretical_buyer_bid(v)
    return extended_eq
```

Appendix C Figure 7: Algorithm for Extension of Buyer-Equilibrium Computation to All Valuations

```python
# ------------------------------------------------------------------------
# Algorithmic Formalization of the Seller-Equilibrium Computation from Method 6
# ------------------------------------------------------------------------

def calculate_seller_equilibrium():
    equilibrium = {}
    warned_bracket_issue = False
    bids = np.array(list(buyer_eq.values()))

    # ------------------------------------------------
    # Algorithmic Formalization of (a) from Method 6
    # ------------------------------------------------

    serious_sellers = serious_processed
    offers = [
        c_upper - (c_upper - v_lower) * (1 - k) * (c / c_upper)
        for c in serious_sellers
    ]

    for c in serious_sellers:
        try:
                                # ------------------------------------------------
            def seller_foc(a):  # Seller First-Order Condition, (b) in Method 6
                                # ------------------------------------------------

                numerator = quad(lambda x: buyer_pdf_func(x), a, c_upper)[0]
                denominator = max(buyer_pdf_func(a), 1e-100)
                return (a - c) - (numerator / denominator)

            bracket_low = max(c, v_lower)
            bracket_high = min(c_upper, c + 0.1*(c_upper - v_lower))
            sol = root_scalar(
                seller_foc,
                bracket=[bracket_low, bracket_high],

                                  # ------------------------------------------------
                method='brentq',  # Root Finding Algorithm using Brent's Method
                                  # ------------------------------------------------

                xtol=1e-6
            )

            equilibrium[c] = np.clip(sol.root, c, c_upper)
        except Exception as e:
            if not warned_bracket_issue:
                print(f"Warning: Seller root-finding failed for c={c:.4f}. Using theoretical offer. Error: {str(e)}")
                warned_bracket_issue = True
            equilibrium[c] = np.clip(theoretical_seller_offer(c), c, c_upper)

    return equilibrium
seller_eq = calculate_seller_equilibrium()
```

Appendix C Figure 8: Algorithm for Seller-Equilibrium Computation

```python
# ------------------------------------------------
# Agorithmic Formalization of (b) from Method 6
# ------------------------------------------------

def extend_seller_equilibrium(full_valuations, eq_dict, v_lower, c_upper):

    extended_eq = {}

    S_v_lower = eq_dict.get(v_lower, theoretical_seller_offer(v_lower))
    S_c_upper = eq_dict.get(c_upper, theoretical_seller_offer(c_upper))

    for c in full_valuations:
        if c < v_lower:

            # ----------------------
            # slope=1 below v_lower
            # ----------------------

            extended_eq[c] = S_v_lower - (v_lower - c)
        elif c > c_upper:

            # ----------------------
            # slope=1 above c_upper
            # ----------------------

            extended_eq[c] = S_c_upper + (c - c_upper)
        else:
            if c in eq_dict:
                extended_eq[c] = eq_dict[c]
            else:
                extended_eq[c] = theoretical_seller_offer(c)
    return extended_eq
```

Appendix C Figure 9: Algorithm for Extension of Seller-Equilibrium Computation to All Valuations

11 Appendix D: Formal Algorithms from Section 4.4

```
# ------------------------------------------------------------------------------
# Algorithmic Formalization of Method 7 (Stabilized Buyer Inequality Verification)
# ------------------------------------------------------------------------------

for v, lam in buyer_eq.items():
    try:

        # ---------------------------------------------
        # Stabilized Derivative Calculation, Part (c)
        # ---------------------------------------------

        dB = complex_step_derivative(theoretical_buyer_bid, v)
        dB = max(abs(dB), SAFE_EPS) * np.sign(dB)

        log_L = log_L_nm(lam, m, n)
        log_M = log_M_nm(lam, m, n)

        L = np.exp(log_L) if not np.isinf(log_L) else 0.0
        M = np.exp(log_M) if not np.isinf(log_M) else 0.0

        g_val = max(buyer_pdf_func(v), SAFE_EPS)

        # ---------------------------------------------------------
        # Algorithmic Formalization of Equation (21) from Method 7
        # ---------------------------------------------------------

        term1 = (v - lam) * (m-1) * L * (g_val / dB) if abs(dB) > SAFE_EPS else 0
        term2 = k * M
        lhs = term1 - term2

        if np.isinf(lhs):
            lhs = np.sign(lhs) * 1e20

        status = "VIOLATED" if lhs > numerical_tolerance else "OK"
        color_code = "\033[91m" if lhs > numerical_tolerance else "\033[92m"

        print("{:<10.5f} | {:<10.5f} | {:<10.2e} | {:<10.2e} | {}{:<12}\033[0m".format(
            v, lam, lhs, numerical_tolerance, color_code, status))

        if lhs > numerical_tolerance:
            inequality_violated = True

    except Exception as e:
        print(f"Error verifying v={v:.5f}: {str(e)}")
        inequality_violated = True

print("\n" + "="*65)
if inequality_violated:
    print("\033[91mWARNING: Potential violations detected\033[0m")
    print("- Possible causes:")
    print("  1. Numerical approximations in density estimation")
    print("  2. Boundary effects near trading interval edges")
    print("  3. Discontinuities in strategic bidding functions")
else:
    print("\033[92mSUCCESS: Inequality holds within tolerance\033[0m")
print("="*65 + "\n")
```

Appendix D Figure 1: Algorithm for Stabilized Buyer-Inequality Verification

```
# ------------------------------------------------------------------------------------
# Algorithmic Formalization of Method 8 (Stabilized Seller-Inequality Verification)
# ------------------------------------------------------------------------------------

seller_inequality_violated = False
numerical_tolerance = 1e-6
SAFE_EPS = 1e-12

# ------------------------------------------
# Same Real Parameters as Buyer Verification
# ------------------------------------------

m = len(buyer_eq)
n = len(seller_eq)

print(f"Market Parameters:")
print(f"- Number of Buyer Values: {m}, Number of Seller Values: {n}")
print(f"- k value: {k:.2f}")
print(f"- Trading interval: [{v_lower:.4f}, {c_upper:.4f}]\n")

print("{:<12} {:<12} {:<12} {:<12} {:<14}".format(
    "Valuation", "Offer", "LHS", "Threshold", "Status"))
print("-"*65)

for c, sigma in seller_eq.items():
    try:

        # ---------------------------------------
        # Seller-specific derivative calculation
        # ---------------------------------------

        dS = complex_step_derivative(theoretical_seller_offer, c)
        dS = max(abs(dS), SAFE_EPS) * np.sign(dS)

        # -----------------------------------------
        # Seller probability terms (swap m/n roles)
        # -----------------------------------------

        log_K = log_K_nm(sigma, n, m)
        log_N = log_M_nm(sigma, n, m)

        K = np.exp(log_K) if not np.isinf(log_K) else 0.0
        N = np.exp(log_N) if not np.isinf(log_N) else 0.0

        # ---------------------------------------
        # Seller Density, Floored at varepsilon
        # ---------------------------------------

        f_val = max(seller_pdf_func(c), SAFE_EPS)

        # --------------------------------
        # Equation (23) from Method 8
        # --------------------------------

        term1 = (sigma - c) * (n-1) * K * (f_val / dS) if abs(dS) > SAFE_EPS else 0
        term2 = k * N
        lhs = term1 - term2

        if np.isinf(lhs):
            lhs = np.sign(lhs) * 1e20

        status = "VIOLATED" if lhs > numerical_tolerance else "OK"
        color_code = "\033[91m" if lhs > numerical_tolerance else "\033[92m"

        print("{:<10.5f} | {:<10.5f} | {:<10.2e} | {:<10.2e} | {}{:<12}\033[0m".format(
            c, sigma, lhs, numerical_tolerance, color_code, status))

        if lhs > numerical_tolerance:
            seller_inequality_violated = True
```

Appendix D Figure 2: Algorithm for Stabilized Seller-Inequality Verification

12 Appendix E: Formal Algorithms from Section 4.5

```
# ------------------------------------------------------------
# Algorithmic Formalization of Method 9 (Strategy Serialization)
# ------------------------------------------------------------

try:
    validate_equilibrium_dict(extended_buyer_eq, extended_seller_eq)

    # ------------------------------------------------------------
    # Encode Buyer Strategies as Ordered Pairs from Equation (26)
    # ------------------------------------------------------------

    df_buyers = pd.DataFrame(
        sorted(extended_buyer_eq.items(), key=lambda x: x[0]),
        columns=["Buyer Valuation", "Bid B(vb)"]
    )

    # ------------------------------------------------------------
    # Utility Calculation for Sellers from the End of Method 8
    # ------------------------------------------------------------

    df_buyers["Buyer Utility"] = df_buyers["Buyer Valuation"] - df_buyers["Bid B(vb)"]

    # ------------------------------------------------------------
    # Encode Seller Strategies as Ordered Pairs from Equation (26)
    # ------------------------------------------------------------

    df_sellers = pd.DataFrame(
        sorted(extended_seller_eq.items(), key=lambda x: x[0]),
        columns=["Seller Valuation", "Offer S(vs)"]
    )

    # ------------------------------------------------------------
    # Utility Calculation for Sellers from the End of Method 8
    # ------------------------------------------------------------

    df_sellers["Seller Utility"] = df_sellers["Offer S(vs)"] - df_sellers["Seller Valuation"]

except Exception as e:
    raise RuntimeError(f"Data validation failed: {str(e)}") from e
```

Appendix E Figure 1: Algorithm for Strategy Serialization

13 Appendix F: Formal Algorithms from Section 5.2

```python
# ------------------------------------------------------------------------
# Algorithmic Formalization of the "Good" Market-Maker Mechanism (Section 5.2)
# ------------------------------------------------------------------------

def good_mm(b_vals, b_bids, s_vals, s_offers):

    # ------------------------------------------------
    # Generate Candidate Pairs as in (a) of Method II
    # ------------------------------------------------

    candidates = []
    for i, (bv, bb) in enumerate(zip(b_vals, b_bids)):
        for j, (sv, sa) in enumerate(zip(s_vals, s_offers)):
            if bv >= sv:
                subsidy = max(0, sa - bb)
                candidates.append((i, j, subsidy, bb >= sa))

    # ----------------------------------------
    # Compute Required Subsidies as in Part (b)
    # ----------------------------------------

    natural = [c for c in candidates if c[3]]
    subsidized = sorted([c for c in candidates if not c[3]],
                        key=lambda x: x[2])

    # ------------------------------------------------------------------
    # Partition Candidates and Process Matches as in Parts (c) and (d)
    # ------------------------------------------------------------------

    used_b, used_s = set(), set()
    matched_pairs = []

    # ------------------------
    # Process Natural Matches
    # ------------------------

    for i, j, _, _ in natural:
        if i not in used_b and j not in used_s:
            used_b.add(i)
            used_s.add(j)
            matched_pairs.append(create_trade_entry(b_vals, b_bids, s_vals, s_offers, i, j, 0))

    # -------------------
    # Subsidized Matches
    # -------------------

    for i, j, subsidy, _ in subsidized:
        if i not in used_b and j not in used_s:
            used_b.add(i)
            used_s.add(j)
            matched_pairs.append(create_trade_entry(b_vals, b_bids, s_vals, s_offers, i, j, subsidy))

    total_subsidy = round(sum(p['subsidy'] for p in matched_pairs), 3)
    return matched_pairs, total_subsidy

def create_trade_entry(b_vals, b_bids, s_vals, s_offers, i, j, subsidy):
    return {
        "buyer_val": round(b_vals[i], 3),
        "buyer_bid": round(b_bids[i], 3),
        "seller_val": round(s_vals[j], 3),
        "seller_offer": round(s_offers[j], 3),
        "subsidy": round(subsidy, 3),
        "price": round((b_bids[i] + s_offers[j])/2, 3)
    }
```

Appendix F Figure 1: Algorithm for A "Good" Market-Maker Mechanism

14 Appendix G: Formal Algorithms from Section 5.3

```python
# ------------------------------------------------------------------
# Algorithmic Formalization of the "Evil" Market Maker from Section 5.3
# ------------------------------------------------------------------

def evil_mm(b_vals, b_bids, s_vals, s_offers, manipulation=0.3):

    if not 0 <= manipulation <= 1:
        raise ValueError("Manipulation parameter must be between 0 and 1")

    # ------------------------------------------------------------------
    # Algorithmic Formulation of Definition 11 (Bid offer Manipulation)
    # ------------------------------------------------------------------


    adjusted_bids = [min(bb + manipulation*(bv - bb), bv)
                     for bv, bb in zip(b_vals, b_bids)]


    adjusted_offers = [max(so - manipulation*(so - sv), sv)
                       for sv, so in zip(s_vals, s_offers)]

    # ------------------------------------------------------------------
    # Algorithmic Formalization of Method 12 (Strategic Matching Process)
    # ------------------------------------------------------------------

    candidates = []
    for i, ab in enumerate(adjusted_bids):
        for j, ao in enumerate(adjusted_offers):
            if ab >= ao:
                profit = ab - ao
                candidates.append((i, j, profit))

    # ------------------------------------------------
    # Sort by Descending Profit for (c) in Method 12
    # ------------------------------------------------

    candidates.sort(key=lambda x: -x[2])

    # ---------------
    # Greedy Matching
    # ---------------

    used_b, used_s = set(), set()
    matched_pairs = []
    total_profit = 0.0

    for i, j, profit in candidates:
        if i not in used_b and j not in used_s:
            used_b.add(i)
            used_s.add(j)
            orig_bid = b_bids[i]
            orig_offer = s_offers[j]
            price = (adjusted_bids[i] + adjusted_offers[j])/2

            matched_pairs.append({
                "buyer_val": round(b_vals[i], 3),
                "original_bid": round(orig_bid, 3),
                "adjusted_bid": round(adjusted_bids[i], 3),
                "seller_val": round(s_vals[j], 3),
                "original_offer": round(orig_offer, 3),
                "adjusted_offer": round(adjusted_offers[j], 3),
                "profit": round(profit, 3),
                "price": round(price, 3)
            })
            total_profit += profit

    total_profit = round(total_profit, 3)
    return matched_pairs, total_profit
```

Appendix F Figure 2: Algorithm for An “Evil” Market-Maker Mechanism

15 Appendix H: Source Code for Replication

For relevant demos of the algorithm at different parameters N and k, click the link below.

https://github.com/JohnWalsh36/John-Walsh-Masters-Thesis

16 Bibliography

References

George A. Akerlof. The market for "lemons": Quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, 1970. doi: 10.2307/1879431.

Richard P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

Kalyan Chatterjee and William Samuelson. Bargaining under incomplete information. *Operations Research*, 31(5):835–851, 1983.

John C. Harsanyi. Games with incomplete information played by "bayesian" players, i: The basic model. *Management Science*, 14(3):159–182, 1967. doi: 10.1287/mnsc.14.3.159.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, 2nd edition, 2002.

Kenneth L. Judd. *Numerical Methods in Economics*. MIT Press, Cambridge, MA, 1998.

Joaquim R. R.Ã. Martins, Peter Sturdza, and Juan J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, 2003. doi: 10.1145/838250.838251.

Roger B. Myerson and Mark A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983. doi: 10.1016/0022-0531(83)90048-0. URL `https://www.sciencedirect.com/science/article/pii/0022053183900480`.

John G. Riley and William F. Samuelson. Optimal auctions. *American Economic Review*, 71(3): 381–392, 1981.

Aldo Rustichini, Mark A. Satterthwaite, and Steven R. Williams. Convergence to efficiency in a simple market with incomplete information. *Econometrica*, 62(5):1041–1063, 1994.

Mark A. Satterthwaite and Steven R. Williams. The rate of convergence to efficiency in the buyer's bid double auction as the market becomes large. *Review of Economic Studies*, 56(3):477–498, 1989. Key pp. 487–488 (Thm. 2, Cor.).

B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, London, 1986. See p. 48, Eq. 3.31.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961. doi: 10.1111/j.1540-6261.1961.tb02789.x.

Steven R. Williams. Existence and convergence of equilibria in the buyer's bid double auction. *Review of Economic Studies*, 58(2):351–374, 1991. doi: 10.2307/2297972.

Robert B. Wilson. Incentive efficiency of double auctions. *Econometrica*, 53(5):1101–1115, 1985. doi: 10.2307/1911013.