A General Theory of Liquidity Provisioning for Prediction Markets

ADITHYA BHASKARA, University of Colorado Boulder, USA
RAFAEL FRONGILLO, University of Colorado Boulder, USA
ELIAS LINDGREN, University of Colorado Boulder, USA
MANEESHA PAPIREDDYGARI, Boston College, USA

Liquidity provisioning in automated market makers is the practice of recruiting third-party liquidity providers (LPs) to contribute assets to the market in exchange for fees skimmed off of trades. This paper introduces a general framework for liquidity provisioning in cost function prediction markets. Our most general protocol allows LPs to submit or update an arbitrary cost function that specifies their liquidity over the entire price space. We show that our protocol encapsulates several notions of running market makers in parallel, which we prove to be equivalent. We also recover existing protocols from decentralized finance as special cases. In our protocol, liquidity can be expressed as a matrix-valued function, which we argue is necessary with three or more securities. Due to this inherent multidimensionality, the design of trading fees with three or more securities is nontrivial: we show that natural axioms on the design of these fees are incompatible.

Acknowledgments

This material is based upon work supported by the Ethereum Foundation and the National Science Foundation under Grant Nos. IIS-2045347 and DMS-1928930, the latter while the second author was in residence at the Mathematical Sciences Research Institute in Berkeley, California, during the Fall 2023 semester. Part of this research was conducted when the fourth author was an intern at the Ethereum Foundation. We thank Gabriel Andrade, Davide Crapis, James Grugett, Ciamac Moallemi, Alex Solleiro, and Bo Waggoner for several interesting discussions and ideas. We also thank Guillermo Angeris for suggestions regarding the piecewise linear market maker and anonymous reviewers for their feedback.

abhaskara@mit.edu, raf@colorado.edu, elias.lindgren@colorado.edu, maneesha.papireddygari@bc.edu

1 INTRODUCTION

Hanson [2003] introduced the concept of an automated market maker to solve thin market problems in combinatorial prediction markets. In contrast to order books and continuous double auctions, where buyers are matched to sellers, automated market makers are central authorities willing to price any bundle of assets to buy or sell. More recently, automated market makers have gained in popularity in the context of decentralized finance as a low-gas way to implement a market on a blockchain [Bartoletti et al., 2022, Base, 2025, Mohan, 2022, Xu et al., 2023]. Along with this trend came the innovation of decoupling the roles of the market mechanism, which facilitates trade, and liquidity providers (LPs), which take on risk to stabilize prices. In this new paradigm, the market mechanism exposes another interface to potential LPs, which may deposit assets in exchange for a cut of the fees charged on the trades using those assets.

Thus far, however, the design of liquidity provisioning interfaces has been somewhat *ad hoc*, limited to its decentralized finance origins, and focused only on the case of two assets. For example, in the Uniswap V2 interface, LPs must deposit funds proportional to the current reserves, a natural yet restrictive interface. Zetlin-Jones et al. [2024] show how these restrictions lead LPs to actively trade against the market—that is, themselves and other LPs—leading to potential inefficiencies. Uniswap V3 adds significant flexibility, but the interface is still somewhat cumbersome: LPs must contend with discrete "buckets" in the price space to allocate their funds. Throughout, the full design space of liquidity provisioning protocols has been far from clear.

This gap is especially large in the case of prediction markets, which often exchange more than two securities. For example, an election market might offer a security for each potential candidate, or even a combinatorial market for the outcomes per state. Offering multiple independent markets for each pair of securities not only leads to information loss, but also creates large arbitrage opportunities that increase the risk of providing liquidity [Dudík et al., 2012]. Thus, effective liquidity provisioning protocols for more than two assets could have a significant impact on the performance and prevalence of prediction markets. Despite all these considerations, no liquidity provisioning protocol for prediction markets has been proposed.

1.1 Contributions

To fill this gap, we introduce a general framework for liquidity provisioning protocols for cost function prediction markets trading any number of securities (§ 3). Our protocol allows LPs to submit an arbitrary cost function, specifying their liquidity over the entire price space. Then, the mechanism determines the deposit required to ensure that LPs never owe the market maker in any future state. Our framework is built on the idea of LPs running market makers "in parallel," and we show that several notions of parallel market making are equivalent, including the scoring rule markets of Hanson [2003] (§ 4).

We also address various difficulties that arise in the case of three or more securities. First, we justify the need for our general protocol by demonstrating the inherent multidimensionality of liquidity (§ 5). The liquidity at a given price must be described by a full matrix, allowing one to assess the liquidity in each "direction," i.e., for each possible trade. As a corollary, any fully general liquidity provisioning protocol must allow liquidity between all assets simultaneously, rather than only allowing liquidity in pairwise markets. Moreover, we show that natural axioms on the design of trading fees are incompatible when there are three or more securities traded (§ 6).

Finally, we additionally show that in the two-asset case, our framework recovers several existing protocols (§ 7) used in decentralized finance. We also provide several restricted protocols that are computationally feasible. Furthermore, we contribute to the decentralized finance literature by giving a fully expressive protocol for the trade of any number of assets.

1.2 Related work

We direct readers to Abernethy et al. [2013], Chen and Pennock [2007] for an overview of the literature on automated market makers for prediction markets, and to Angeris and Chitra [2020] for those in decentralized finance. Our work heavily relies on Frongillo et al. [2023], which establishes the equivalence of (non-parallelized) automated market makers as used for prediction markets to those used in decentralized finance. One can view our analysis of the trade split $\mathbf{r} = \sum_i \mathbf{r}^i$ as a special case of optimal routing problems stated in [Angeris et al., 2022, Diamandis et al., 2023]. We provide a closed form solution to this special case not yet seen in the literature. This work is also related to recent explorations of running parallel LMSRs [Dudík et al., 2021] and of geometric aspects of automated market makers [Angeris et al., 2023]. In particular, the Minkowski sum operations in the latter paper can be seen as implicitly computing an infimal convolution, which they also view as a combined market maker. However, their connections to implementing liquidity provisioning are not explored beyond a very restricted setting. Additionally, Ramseyer et al. [2024] study similar notions of price coherence and parallel market makers to ours, but while working to design batch exchanges that support constant function market makers for liquidity provision.

One could view our general framework as theoretically formalizing Minswap [Nguyen, 2021], which is designed on Cardano to accommodate multiple LP pools. Perhaps closest to our work is Milionis et al. [2023], which asks LPs for their "demand curves" to be aggregated into a two-asset market maker. Our framework can be thought of as a more general way of thinking about LPs running several markets in parallel. Their demand curves $h(p)$ (denoted $g(p)$ in their paper) are related to our market scoring rule with $h(p) = S(p, 1)$ and $-\int pdh(p) = S(p, 0)$. While demand curves are well-rooted in microeconomic foundations [Milionis et al., 2024], we demonstrate in § 5 that cost functions (or some other suitable higher-dimensional notion of demand curves) are necessary to give a fully general liquidity provisioning protocol for more than two assets, a crucially important case for prediction markets as we describe above. They are also instrumental in helping us propose a closed-form construction of aggregate CFMMs and in helping us realize and prove different equivalent interpretations (Theorem 1). We additionally handle the technical issues that arise with more than two securities (Theorem 2).

There has additionally been some work on the *incentives* and *effects* that trading fees have on traders and LPs' behavior [Angeris et al., 2024, Campbell et al., 2025, Milionis et al., 2025], but the design of trading fees themselves has been unexplored. This gap is especially large for three or more securities, where we show several natural properties to be incompatible.

2 BACKGROUND

2.1 Notation and convex analysis

Vectors are denoted in bold, e.g., $\mathbf{q} \in \mathbb{R}^n$, and $q_i \in \mathbb{R}$ denotes the ith coordinate of $\mathbf{q}$. The all-zeros vector is $\mathbf{0} = (0, \ldots, 0)$ and the all-ones vector is $\mathbf{1} = (1, \ldots, 1)$. We define the indicator vector $\boldsymbol{\delta}^i$ by $\delta^i_i = 1$ and $\delta^i_j = 0$ for $j \neq i$. Comparison between two vectors is pointwise, e.g., $\mathbf{q} > \hat{\mathbf{q}}$ if $q_i > q'_i$ for all $i = 1, \ldots, n$, and similarly for $\geq$. We say $\mathbf{q} \gneqq \hat{\mathbf{q}}$ when $q_i \geq q'_i$ for all i and $\mathbf{q} \neq \hat{\mathbf{q}}$. Define $\mathbb{R}^n_{\geq 0} = \{\mathbf{q} \in \mathbb{R}^n \mid \mathbf{q} \geq \mathbf{0}\}$, $\mathbb{R}^n_{>0} = \{\mathbf{q} \in \mathbb{R}^n \mid \mathbf{q} > \mathbf{0}\}$, *et cetera*. Finally, we denote the probability simplex by $\Delta_n = \{\mathbf{p} \in \mathbb{R}^n_{\geq 0} \mid \langle \mathbf{p}, \mathbf{1} \rangle = 1\}$. Let $f : \mathbb{R}^n \to \mathbb{R}$. We use the following conditions:

- *convex*: $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0, 1], f(\lambda \mathbf{x} + (1 - \lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y})$.
- $\mathbf{1}$*-invariant*: $f(\mathbf{q} + \alpha \mathbf{1}) = f(\mathbf{q}) + \alpha$ for all $\mathbf{q} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$.
- *1-homogeneous (on* $\mathbb{R}^n_{\geq 0}$*)*: $f(\alpha \mathbf{q}) = \alpha f(\mathbf{q})$ for all $\mathbf{q} \geq \mathbf{0}$ and $\alpha > 0$.

We write f' to indicate differentiation when f is 1-dimensional.

Definition 1 (Convex conjugate). *For a function* $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ *we define its* convex conjugate $f^* : \mathbb{R}^n \to \mathbb{R} \cup \{\infty, -\infty\}$ *by* $f^*(\mathbf{x}^*) = \sup_{\mathbf{x} \in \mathbb{R}^n} \langle \mathbf{x}^*, \mathbf{x} \rangle - f(\mathbf{x})$.

Definition 2 (Subgradient). *For a function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $\mathbf{x} \in \mathbb{R}^n$ we define the set of* subgradients *of f at $\mathbf{x}$ by $\partial f(\mathbf{x}) = \{\mathbf{x}^* \in \mathbb{R}^n \mid \forall \mathbf{x}' \in \mathbb{R}^n\, f(\mathbf{x}') - f(\mathbf{x}) \geq \langle \mathbf{x}^*, \mathbf{x}' - \mathbf{x} \rangle\}$.*

Definition 3 (Infimal convolution). *For functions $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ we define their* infimal convolution *$f = \bigwedge_i f_i$ by $f(\mathbf{x}) = \inf\left\{\sum_i f_i(\mathbf{x}^i) \mid \sum_i \mathbf{x}^i = \mathbf{x}\right\}$, where the $\mathbf{x}^i$ range over $\mathbb{R}^n$.*

Definition 4 (1-homogeneous extension $\overline{f}$). *Given $f : \Delta_n \to \mathbb{R}$, we define its* 1-homogeneous extension *$\overline{f} : \mathbb{R}^n_{\geq 0} \to \mathbb{R}$ by $\overline{f}(\mathbf{x}) := \|\mathbf{x}\|_1 f(\mathbf{x}/\|\mathbf{x}\|_1)$ for $\mathbf{x} \neq \mathbf{0}$ and $\overline{f}(\mathbf{0}) = 0$.*

2.2 Cost functions and prediction markets

Automated market makers (AMMs) are mechanisms that are always willing to trade a bundle of n securities for some price. In contrast to traditional order book settings, where buyers and sellers must be matched, traders can trade with AMMs directly. Prediction markets are AMM mechanisms that seek to elicit probability distributions over future events by allowing traders to buy and sell securities. Some common instantiations of AMM prediction markets can be seen in horse betting, Iowa electronic markets, and more recently, Manifold [2022] and Andrade [2025].

Suppose a random variable Y about a future event takes values from set $\mathcal{Y}$, which contains n mutually exclusive and exhaustive outcomes. A trader holding an Arrow-Debreu (AD) security associated with $y \in \mathcal{Y}$ gets paid \$1 when outcome y happens and \$0 otherwise. A security market is *complete* if it trades n independent AD securities, one for each outcome. While our Protocol 1 works for incomplete markets, we restrict our attention to complete markets in this paper for ease of exposition. Our proofs in § 4.2 rely heavily on the market being complete. But, in Appendix E, we discuss how our framework extends to the incomplete case.

Abernethy et al. [2013], Chen and Pennock [2007], Chen et al. [2013] characterize prediction markets, and Abernethy et al. [2013] show that they should be implemented by a cost function-based market maker satisfying certain conditions in order to satisfy certain information elicitation axioms. We define these below.

Definition 5 (Cost function-based market maker). *A cost function-based market maker with n securities is one that prices each security via a differentiable potential function $C : \mathbb{R}^n \to \mathbb{R}$. Suppose a trader wants to purchase a bundle of securities $\mathbf{r} \in \mathbb{R}^n_{\geq 0}$; that is, r_i shares of security i, when the market has current liability of $\mathbf{q}$. Then, the trader must pay $C(\mathbf{q} + \mathbf{r}) - C(\mathbf{q})$ in cash to the market maker.*

Cost function-based markets always maintain a *liability* $\mathbf{q} \in \mathbb{R}^n$ of securities, q_i of security i that the market has sold so far. The term liability comes from the fact that q_i is the amount due to traders upon outcome i. As shown by Abernethy et al. [2013], prediction markets that elicit information well are precisely cost function-based prediction markets with C convex and $\mathbf{1}$-invariant.

Remark. *As remarked in Frongillo et al. [2023], in a complete AD securities market, holding one of each security, i.e., $\mathbf{1}$, is equivalent to holding \$1 cash. Without loss of generality, therefore, one can restrict attention to "net trades" $\mathbf{r} = \mathbf{r}' - (C(\mathbf{q} + \mathbf{r}') - C(\mathbf{q}))\mathbf{1}$, which subtract the cost of the trade $\mathbf{r}'$, converting cash to $\mathbf{1}$. Using the $\mathbf{1}$-invariant property of C, we can see that $C(\mathbf{q} + \mathbf{r}) = C(\mathbf{q})$. While it is not customary for predictions to deal with the net trade, in our setting net trades simplify various results, and more readily connect to the decentralized finance literature.*

2.3 Scoring rules

Scoring rules were introduced by Brier [1950] to score forecasts of a random variable such as Y above. In this setting, we seek to design a score $S(\mathbf{p}, y)$ that determines the quality of prediction $\mathbf{p} \in \Delta_{\mathcal{Y}}$ upon observing the outcome $y \in \mathcal{Y}$, with the property that $\mathbb{E}_{\mathbf{p}} S(\hat{\mathbf{p}}, Y)$ is maximized at $\hat{\mathbf{p}} = \mathbf{p}$. The full characterization of such "proper" scoring rules takes the form

$$S_G(\mathbf{p}, y) = G(\mathbf{p}) + \left\langle \mathbf{d}G_{\mathbf{p}}, \delta_y - \mathbf{p} \right\rangle$$

where $G : \Delta_{\mathcal{Y}} \to \mathbb{R}$ is a convex function [Gneiting and Raftery, 2007]. When $\mathcal{Y} = \{0, 1\}$, we can write $p \in [0, 1]$ to be the predicted probability that $Y = 1$, and write $S_g(p, y) = g(p) + g'(p)(y - p)$ for $g : [0, 1] \to \mathbb{R}$ convex.

Hanson [2003] showed that scoring rules could be used to design AMMs in a form we call a *scoring rule market*; see Protocol 2. The basic idea is to pay traders according to a difference of scoring rules, with the latest trader's prediction acting as the current market price. It was later shown that this formulation is equivalent in a strong sense to the cost function market makers described above [Abernethy et al., 2013]. Specifically, the scoring rule market for S_G is equivalent to the cost function market maker with cost function $C = G^*$, the convex conjugate of G.

A corollary of these connections, leveraged in Frongillo et al. [2023], is that one can use scoring rules as vectors to convert between the market price vector and the current liability/reserve vector. Specifically, let $S_G(\mathbf{p}, \cdot) = (S_G(\mathbf{p}, y))_{y \in \mathcal{Y}} \in (\mathbb{R} \cup \{\infty\})^n$ be the scoring rule vector for price $\mathbf{p}$. Then, up to a uniform shift, the liability vector of a cost function market maker with cost function $C = G^*$ at price $\mathbf{p}$ is simply $S_G(\mathbf{p}, \cdot)$. We will use this correspondence throughout the paper, as well as the 2-outcome version $S_g(p, \cdot) = (g'(p), 0) + (g(p) - p \cdot g'(p))\mathbf{1} \in (\mathbb{R} \cup \{\infty\})^2$.

2.4 Technical definitions

We give some technical definitions relating to § 2.2 and § 2.3 that we use in later sections.

Definition 6 (Smoothness of G). *We say a convex function $G : \Delta_n \to \mathbb{R}$ is* smooth *if its 1-homogeneous extension $\overline{G} : \mathbb{R}^n_{\geq 0}$ is differentiable.*

The key class of the functions G we restrict to is as follows.

Definition 7 (Generating function). *We say $G : \Delta_n \to \mathbb{R}$ is a* generating function *if it is convex, smooth, and bounded on Δ_n.*

Definition 8 (Pseudobarrier, Abernethy et al. [2013]). *A generating function G is a* pseudobarrier *if for any sequence $\{\mathbf{p}^j \in \operatorname{relint} \Delta_n\}_j$ converging to the relative boundary of Δ_n, and $\{\mathbf{q}^j \in \partial G(\mathbf{p}^j)\}_j$, then $\|\mathbf{q}^j\| \to \infty$.*

A common example of a pseudobarrier[1] is (negative) Shannon entropy $G(\mathbf{p}) = \sum_y p_y \log p_y$. Another is the dual of the constant product market maker $G(\mathbf{p}) = -n(\prod_y p_y)^{1/n}$.[2]

We now give the sets of cost and generating functions used in the general protocols. Let $\mathcal{G}_n$ be the set of nonpositive generating functions $G : \Delta_n \to \mathbb{R}_{\leq 0}$, and $\mathcal{G}_n^* \subseteq \mathcal{G}_n$ those which are pseudobarriers.[3] Let C_n and C_n^* be the sets of conjugates of $\mathcal{G}_n$ and $\mathcal{G}_n^*$, respectively.

2.5 Automated market makers and liquidity provisioning in decentralized finance

A major AMM innovation in decentralized finance is the introduction of *liquidity provisioning*. In traditional AMMs, the market maker takes on an additional risk of price fluctuations of the reserves for the ability to run a market always willing to price a bundle of assets. The decentralized finance implementations of AMMs, though, have outsourced provisioning these reserves, and hence liquidity, to external parties called liquidity providers (LPs). The AMMs typically define trade dynamics when liquidity is fixed. In this setting, traders can exchange assets with the market maker in a way that keeps the reserves/liability on the same invariant curve of φ or C. Decentralized finance protocols

[1] This term was coined in Abernethy et al. [2013] and used similarly to our setting: ensuring that the market price remains in the relative interior of the simplex.

[2] To see that this dual is correct, one can observe that it is 1-homogeneous, and thus $S(\mathbf{p}, y) = -(\prod_{y'} p_{y'})^{1-1/n}(\prod_{y' \neq y} p_y)$. Now letting $\mathbf{x} = -S(\mathbf{p}, \cdot)$ be the corresponding reserve vector, and computing the product, we have $\prod_y q_y = \prod_y (\prod_{y'} p_{y'})^{1-1/n}(\prod_{y' \neq y} p_y) = 1$.

[3] Given any bounded generating function G, to obtain the optimal liability, we can simply replace it by the function $\mathbf{p} \mapsto G(\mathbf{p}) - \langle \mathbf{p}, \mathbf{q} \rangle$ where $q_i = G(\boldsymbol{\delta}^i)$.

like Uniswap V2 and Uniswap V3 also allow liquidity providers (LPs) to change the market's liquidity while keeping the price $\mathbf{p}$ invariant [Adams et al., 2020, 2021]. LPs may either add, or *mint*, liquidity to the market or remove, or *burn*, liquidity from the market. LPs make it easier for the AMM to conduct trades by absolving the market maker of the risk of providing liquidity. As compensation for taking on the risk, LPs are rewarded using trading fees, which are skimmed off along with the trade requested. These fees form a pool to be distributed proportionally to LPs as the liquidity they allocated is used.

Notably, in decentralized finance, the goal is not information elicitation, as in prediction markets, but rather the exchange of assets—namely cryptocurrencies. Analogous to how cost function markets maintain a liability vector $\mathbf{q}$, AMMs maintain a reserve vector $\mathbf{x} = -\mathbf{q}$ of assets. Constant function market makers (CFMMs) are a special type of AMM. For various restrictions on their design, Angeris and Chitra [2020], Frongillo et al. [2023], Schlegel et al. [2022] argue that CFMMs satisfy desirable market making axioms. We define these market makers below.

Definition 9 (Constant function market maker, CFMM). *A constant function market maker (CFMM) is a market maker based on a potential function $\varphi : \mathbb{R}^n \to \mathbb{R}$ that maintains a liability $\mathbf{q} \in \mathbb{R}^n$. At the current liability, the set of trades $\mathbf{r}$ available are those that satisfy $\varphi(\mathbf{q} + \mathbf{r}) = \varphi(\mathbf{q})$. After a trade, the liability vector updates to $\mathbf{q} \leftarrow \mathbf{q} + \mathbf{r}$.*

Consistent with this definition, in this paper, we adopt the sign convention that trades are always oriented toward the trader. For example, a trade $\mathbf{r} = (1, -3)$ corresponds to giving the trader 1 unit of asset 1 in exchange for 3 units of asset 2

The relationship between cost function prediction market makers and CFMMs is thoroughly explored in Frongillo et al. [2023]. The two objects are different, but give rise to equivalent characterizations of markets. The classic cost function market makers commonly employed in prediction markets are special cases of CFMMs that retain the full flexibility of general potential functions φ. We use cost functions throughout, even when discussing CFMMs, as they have more mathematical structure without loss of expressiveness. For example, while for any potential φ, one can take ratios of partial derivatives to compute relative prices, this task is even easier for cost functions, as prices are normalized. In the context of prediction markets, normalization means that prices $\mathbf{p} \in \partial C(\mathbf{q})$ can be thought of as a probability distribution over outcomes. With or without this interpretation, we frequently use the fact that $\partial C(\mathbf{q}) \subseteq \Delta_n$ for all $\mathbf{q} \in \mathbb{R}^n$.

3 LIQUIDITY PROVISIONING PROTOCOL FOR PREDICTION MARKETS

In this section, we first give intuition in § 3.1 for why liquidity provisioning can be thought of as running "parallel" markets. We detail our general, cost function-based, prediction market liquidity provisioning protocol in § 3.2. Then, in § 3.3 we provide a scoring rule-based protocol. We show both protocols to be equivalent in § 4, and we also show how these protocols lead to several equivalent ways of thinking about LPs running markets in parallel. These intuitive interpretations not only enhance our understanding of how liquidity provisioning can be implemented in prediction markets, but also justify the framework we propose. We end this section by discussing some constraints on the design of our protocol's market-making functions in § 3.4 and § 3.5.

3.1 Liquidity provisioning as competing market makers

In traditional financial markets, such as continuous double-auctions, a market maker is an entity that offers both to buy and sell an asset. Typically the buy price is lower than the sell price; the difference comprises the *bid-ask spread*. Market makers earn a profit equal to the bid-ask spread whenever both buy and sell orders are executed, while remaining even with respect to the asset. In essence, market making is all about providing liquidity for a small premium, or "fee," as given by the spread. In these traditional markets, liquidity provisioning happens naturally, as often multiple

market makers coexist. Rational traders will only buy or sell from the most favorable price offered, switching at will between different market makers.

The key idea behind our protocol, therefore, is to implement liquidity provisioning in the same manner, with multiple coexisting automated market makers. That is, we seek a protocol that implements an LP as simply another "competing" market maker, and we let traders interact with them all at once, i.e., in parallel. How could one implement such a protocol?

We will show that there are several equivalent ways to imagine this transaction proceeding. First, a trader could select a valid trade $\mathbf{r}^i$ for the automated market maker of each LP i, resulting in a net trade $\mathbf{r} = \sum_i \mathbf{r}^i$. As detailed in § 2.5, these valid trades can be expressed as those satisfying $C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{q}^i)$ given a convex cost function C_i and current liability vector $\mathbf{q}^i$ for LP i. Second, a trader could execute a trade in continuous time, at each moment trading with the LP offering the most favorable price, eventually stopping and yielding a net trade $\mathbf{r}$. Perhaps surprisingly, by fundamental results in convex analysis, these two approaches are identical. Taken together, we can see that any Pareto-optimal trade leaves the combined market in a coherent state, with the price of each LP matching the global market price.

At first glance, it might appear that a major downside of our approach is the need for traders to interact directly with each LP, increasing the complexity of interaction required. Fortunately, one can simplify the interface: there always exists a single aggregate cost function C that captures the available net trades. Specifically, given the cost functions C_i defining each market maker, the valid trades in the combined market are exactly those of their *infimal convolution* $C = \bigwedge_i C_i$. Thus, the trader can simply choose any trade satisfying $C(\mathbf{q} + \mathbf{r}) = C(\mathbf{q})$, and behind the scenes, the split $\mathbf{r} = \sum_i \mathbf{r}^i$ can be computed along with the corresponding fees. This aggregation yields a third equivalent notion of competing market makers.

Equivalently, we may consider the scoring rule market (SRM) formulation of cost functions. As mentioned in § 2.3, a cost function market given by a convex cost function C is equivalent to the scoring rule market given by its convex conjugate $G = C^*$. We show that our notion of LPs as parallel market makers extends quite elegantly to SRMs. The combined market simply uses the sum of the scoring rules S_{G_i} of all the component LPs, or equivalently uses a single scoring rule S_G generated by the sum of the generating functions $G = \sum_i G_i$. See § 3.3 for the full details of the scoring rule version of our protocol.

In § 5, we discuss how liquidity can be expressed as a matrix-valued function, either by $(\nabla^2 C)^+$, or dually, $\nabla^2 \overline{G}$ where $G = C^*$. As a consequence, the total liquidity of the combined market is $\nabla^2 \overline{G} = \sum_i \nabla^2 \overline{G_i}$. In other words, just as one would hope, adding another parallel market maker simply adds the corresponding liquidity $\nabla^2 \overline{G_i}$ to the pool.

Remark. *Many existing protocols require LPs to simply deposit tokens to provide liquidity. At first glance, it might appear that requiring LPs to provide cost functions C_i is more complex. Our Protocol 1, and its equivalent protocols, clarifies that even in existing protocols that just ask LPs to deposit tokens, the LPs are implicitly specifying cost functions. Hence, we may recover existing protocols as special cases by specifying a restricted set $\mathcal{C}_{\mathrm{LP}}$ of cost functions that LPs may (explicitly or more commonly, implicitly) provide. For instance, in Protocol 4, we recover Uniswap V2, where LPs simply deposit some numerical quantity of "liquidity" that is spread evenly across the entire price space. Under the hood, LPs are specifying scaled copies of the "base shape" of Uniswap's cost function. In the more expressive Uniswap V3 which we recover in Protocol 5, LPs deposit tokens determined by a numerical quantity of "liquidity" in discrete price ranges called buckets, and again they are implicitly specifying cost functions.*

3.2 Cost function-based general protocol

Our proposed protocol for liquidity provisioning in prediction markets is described in Protocol 1. At a high level, the protocol works as follows. Let n be the number of securities. The market creator

Protocol 1 General protocol as parallel market makers

global constant C_{init}, C_{LP}, $\text{fee}(), \text{fee}_i() \in \mathbb{R}_{\geq 0}$.
global variables $k \in \mathbb{N}$, $\{\mathbf{q}^i \in \mathbb{R}^n\}_{i=0}^k$, $\{C_i \in C_{\text{LP}}\}_{i=0}^k$
$\text{liability}(C) := \mathbf{q} \in \mathbb{R}^n$ s.t. $\partial C_0(\mathbf{q}^0) \cap \partial C(\mathbf{q}) \neq \emptyset$ and $C(\mathbf{q}) = 0$ ▷ Price matching, no-liability
function Initialize($\mathbf{q} \in \mathbb{R}^n, C \in C_{\text{init}}$)
    $(k, \mathbf{q}^0, C_0) \leftarrow (0, \mathbf{q}, C)$
    check $\mathbf{q}^0 = \text{liability}(C_0)$ s.t. no-liability and optimal deposits are satisfied.
function RegisterLP($i = k + 1$)
    $(k, \mathbf{q}^i, C_i) \leftarrow (k + 1, 0, \max)$
function ModifyLiquidity($i \in \mathbb{N}, C \in C_{\text{LP}}$)
    request $\mathbf{r}^i = \mathbf{q}^i - \text{liability}(C)$ from LP i s.t. no-liability and optimal deposits are satisfied.
    $(\mathbf{q}^i, C_i) \leftarrow (\mathbf{q}^i - \mathbf{r}^i, C)$
function ExecuteTrade($\mathbf{r} \in \mathbb{R}^n$)
    $\mathbf{q} \leftarrow \sum_{i=0}^k \mathbf{q}^i$
    check $C(\mathbf{q} + \mathbf{r}) = C(\mathbf{q})$ where $C = \bigwedge_{i=0}^k C_i$
    trader pays $\text{fee}(\mathbf{r}, \mathbf{q})$ cash in fees
    give $\mathbf{r}$ to trader
    write $\mathbf{r} = \sum_{i=0}^k \mathbf{r}^i$ s.t. $\forall i, C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{q}^i)$
    for each LP i **do**
        LP i gets $\text{fee}_i(\mathbf{r}, \mathbf{q})$ fees
        $\mathbf{q}^i \leftarrow \mathbf{q}^i + \mathbf{r}^i$

acts as the initial LP, giving reserves $\mathbf{q}^0$ to the liquidity pool and specifying the initial cost function C_0. When an additional LP i enters, their liability vector and cost function are initialized to the trivial values $\mathbf{q}^i = \mathbf{0}$ and $C_i(\mathbf{q}) = \max(\mathbf{q}) := \max_j q_j$, so that they initially provide no liquidity.[4] The ModifyLiquidity function handles an LP adding, removing, or otherwise altering their deposited liquidity: they simply replace their cost function with a different one, and are charged up-front the minimal deposit to ensure *no liability*, i.e., that they will never owe the market maker in any future state. We provide more discussion of the *no-liability* condition in § 3.4. When removing all liquidity, the LP simply sets $C_i = \max$ once again, and is given back their entire deposit. ExecuteTrade checks if a trade $\mathbf{r}$ is an allowed trade with the overall cost function $C = \bigwedge_{i=0}^k C_i$, and if so, requests an additional fee of $\text{fee}(\mathbf{r}, \mathbf{q})$ cash from the trader. Under the hood, it then finds the optimal split $\mathbf{r} = \sum_i \mathbf{r}^i$ into smaller trades, executing each with the corresponding LP and doling out $\text{fee}_i(\mathbf{r}, \mathbf{q})$ in fees.

Two reasonable fee choices are $\text{fee}(\mathbf{r}, \mathbf{q}) = \beta\|\mathbf{r}\|$ and $\text{fee}_i(\mathbf{r}, \mathbf{q}) = \beta\|\mathbf{r}\| \frac{\|\mathbf{r}^i\|}{\sum_j \|\mathbf{r}^j\|}$ for some norm $\|\cdot\|$ and $\beta > 0$. The form of fee_i here is to ensure budget balance of the fees, so that the market maker does not owe LPs more than the trader pays. We provide a deeper discussion of the design of the fee functions in § 6, and prove that natural axioms are incompatible.

A key step in the protocol is the computation of liabilities, from which several questions arise. If an LP wishes to provide liquidity using C, and the current price is $\mathbf{p}$, what do they need to deposit? Also, does the required split $\mathbf{r} = \sum_i \mathbf{r}^i$ always exist? We answer these questions in the later sections.

[4] To see why this choice is correct, note that the "bid-ask spread" of C_i at $\mathbf{q}^i = 0$ is maximal; every price vector in Δ_n is consistent, and any trade occurs at the worst feasible price. More technically, adding the max function to the infimal convolution $C = \bigwedge_{i=0}^k C_i$ does not change the result. Dually, the conjugate of max is the convex indicator of Δ_n, so this choice adds liquidity $G_i = 0$; see § 3.

3.3 Scoring rule-based general protocol

As some readers might be more familiar with the scoring rule markets of Hanson [2003], we give Protocol 2 as an alternative to Protocol 1. Here, we keep track of the market price $\mathbf{p}$ instead of the liability vector $\mathbf{q}$ and use scoring rules instead of cost functions. We prove in Appendix A.3 that Protocols 1 and 2 are equivalent under mild conditions.

Protocol 2 General protocol as parallel scoring rule markets

global constant $\mathcal{G}_{\text{init}}$, $\mathcal{G}_{\text{LP}}$, $\text{fee}(), \text{fee}_i() \in \mathbb{R}_{\geq 0}$.
global variables $k \in \mathbb{N}$, $\mathbf{p} \in \operatorname{relint} \Delta_n$, $\{G_i \in \mathcal{G}_{\text{LP}}\}_{i=0}^{k}$
liability$(G, \mathbf{p}) := S_G(\mathbf{p}, \cdot)$ where $S_G(\mathbf{p}, \cdot) = \mathbf{d}G_{\mathbf{p}} + (G(\mathbf{p}) - \langle \mathbf{d}G_{\mathbf{p}}, \mathbf{p} \rangle)\mathbf{1}$.
function Initialize($\mathbf{p} \in \operatorname{relint} \Delta_n, G \in \mathcal{G}_{\text{init}}$)
    $(k, \mathbf{p}, G_0) \leftarrow (0, \mathbf{p}, G)$
    $\mathbf{q}^0 := -S_{G_0}(\mathbf{p}, \cdot)$ s.t. no-liability and optimal deposits are satisfied.
function RegisterLP($i = k + 1$)
    $(k, G_k) \leftarrow (k + 1, 0)$
function ModifyLiquidity($i \in \mathbb{N}, G' \in \mathcal{G}_{\text{LP}}$)
    request $S_{G_i}(\mathbf{p}, \cdot) - S_{G'}(\mathbf{p}, \cdot)$ from LP i s.t. no-liability and optimal deposits are satisfied.
    $G_i \leftarrow G'$
function ExecuteTrade($\hat{\mathbf{p}} \in \operatorname{relint} \Delta_n$)
    $\mathbf{r} \leftarrow S_G(\hat{\mathbf{p}}, \cdot) - S_G(\mathbf{p}, \cdot)$ where $G = \sum_i G_i$
    pay $\text{fee}(\mathbf{r}, \mathbf{p})$ cash in fees
    give $\mathbf{r}$ to trader
    for each LP i **do**
        $\mathbf{r}^i \leftarrow S_{G_i}(\hat{\mathbf{p}}, \cdot) - S_{G_i}(\mathbf{p}, \cdot)$.
        pay $\text{fee}_i(\mathbf{r}, \mathbf{p})$ in fees to LP i
        $\mathbf{q}^i \leftarrow \mathbf{q}^i + \mathbf{r}^i$

3.4 Ensuring no liability

To capture the *no-liability condition*, we require that each LP i should never owe the market maker shares of any security. When entering the market, LP i is required to deposit some assets $\mathbf{x} \in \mathbb{R}^n_{\geq 0}$ according to its cost function C_i and the current overall liability. We consider LP i's liability to be $\mathbf{q}^i = -\mathbf{x}$ where $\mathbf{q}^i$ is chosen so that $C_i(\mathbf{q})$ implies $\mathbf{q} \preceq 0$ for all possible future states $\mathbf{q}$.

One way to ensure this is that, if an LP provides a cost function C where some level set satisfies no-liability, the protocol could compute it and request a corresponding deposit to cover the liability. More formally, if $\{\mathbf{q} \in \mathbb{R}^n \mid C_i(\mathbf{q}) = \alpha\} \subseteq \mathbb{R}^n_{\preceq 0}$, i.e. every liability vector in the α-level set of C_i is nonpositive, the protocol could compute this α and request a deposit $-\mathbf{q}$ in the α-level set such that $\nabla C_i(\mathbf{q}) = \mathbf{p}$, the current price. Equivalently, we may require that the LP encode the valid trades in the zero level set by submitting an offset cost function $\hat{C}_i := C_i - \alpha$, so that the α-level set is shifted to have value 0. In Protocol 1, we require LPs submit such a cost function. Equivalently, we may ensure no-liability in Protocol 2 by requiring the generating function G_i to be nonpositive everywhere on the simplex.

Recall that for two outcomes, the constant product potential $\varphi(\mathbf{q}) = q_1 q_2$ has an equivalent cost function [Chen and Pennock, 2007, Frongillo et al., 2023] given by

$$C(\mathbf{q}) = \frac{1}{2}\left(q_1 + q_2 + \sqrt{4\alpha^2 + (q_1 - q_2)^2}\right).$$

Taking the 0-level set, we have $q_1 + q_2 + \sqrt{4\alpha^2 + (q_1 - q_2)^2} = 0$ which implies $q_1 + q_2 < 0$. Then $(q_1 + q_2)^2 = 4\alpha^2 + (q_1 - q_2)^2$, which reduces to $q_1 q_2 = \alpha^2$. By the first observation, we must have $\mathbf{q} < \mathbf{0}$, giving no liability. While one could take $\hat{C}(\mathbf{q}) = C(\mathbf{q}) + 1$ to arrive at the same liquidity level, one can check that C gives rise to the minimal deposit required for that level.

3.5 Optimal deposits

While the no-liability condition requires that an LP's liability always be nonnegative, the optimal deposits condition requires that all the assets deposited by some LP may be used in some trade, and therefore none of it is "wasted." To satisfy optimal deposits, the LP must deposit a liability $\mathbf{q}$ such that for every asset i and any $\epsilon > 0$, there exists a valid trade $\mathbf{r}$ such that the resulting liability $\hat{\mathbf{q}} = \mathbf{q} + \mathbf{r}$ contains less than ϵ securities[5] of asset i We can characterize optimal deposits in the SRM protocol (Protocol 2) by requiring that the generating function G approaches 0 at the vertices of the simplex.

Proposition 1. *Let $G : \Delta_n \to \mathbb{R}$ be a convex generating function where $\lim_{j\to\infty} G(\mathbf{p}^j) = 0$ for any Cauchy sequence $(\mathbf{p}^j)_{j=0}^{\infty}$ of prices where $\lim_{j\to\infty} \mathbf{p}^j = \hat{\mathbf{p}}$ where $\hat{\mathbf{p}}$ is the unique price with $\hat{\mathbf{p}}_i = 1$. Given any price $\mathbf{p}$ and corresponding liability vector $\mathbf{q} = S_G(\mathbf{p}, \cdot)$, for any $\epsilon < 0$, there exists a legal trade $\mathbf{r}$ at liability $\mathbf{q}$ such that $q_i + r_i > \epsilon$.*

Proof. Given an initial liability $\mathbf{q}$, legal trade $\mathbf{r}$, and resulting liability $\hat{\mathbf{q}}$, recall from Protocol 2 that $\hat{\mathbf{q}} = S_G(\hat{\mathbf{p}}, \cdot) = \mathbf{d}G_{\hat{\mathbf{p}}} + (G(\hat{\mathbf{p}}) - \langle \mathbf{d}G_{\hat{\mathbf{p}}}, \hat{\mathbf{p}} \rangle)\mathbf{1}$. The component $\hat{q}_i$ of $\hat{\mathbf{q}}$ corresponding to security i is therefore $S_G(\hat{\mathbf{p}}, y_i) = \mathbf{d}G_{\hat{p}_i} + G(\mathbf{p}) - \mathbf{d}G_{\hat{p}_i} = G(\hat{\mathbf{p}})$, where y_i is the outcome corresponding to asset i and $\mathbf{d}G_{\hat{p}_i}$ is the i^{th} component of some subgradient $\mathbf{d}G_{\hat{\mathbf{p}}}$ of G at $\hat{\mathbf{p}}$. Now, let $\hat{\mathbf{p}}$ be the unique price such that $\hat{p}_i = 1$. Let $(\mathbf{p}^j)_{j=0}^{\infty}$ be some Cauchy[6] sequence of prices approaching $\hat{\mathbf{p}}$, and $(\mathbf{q}^j)$ be the corresponding sequence of liability vectors. We have that

$$\lim_{j\to\infty} \mathbf{q}^j = \lim_{j\to\infty} S_G(\mathbf{p}^j, y_i) = \lim_{j\to\infty} \mathbf{d}G_{p_i^j} + G(\mathbf{p}^j) - \mathbf{d}G_{p_i^j} = \lim_{j\to\infty} G(\mathbf{p}^j) = 0.$$

As the limit approaches 0 from below, for any $\epsilon < 0$, there exists some $\mathbf{p}^j$ such that $q_i^j > \epsilon$. □

3.6 Example of equivalent protocols in action

We construct an example of both equivalent protocols in action that makes use of the Logarithmic Market Scoring Rule (LMSR), popularized by Hanson [2003]. We consider a setting with three assets and two LPs. While we focus on the SRM version of our protocol for the sake of easy and clean computation, it is simple to derive the exchange of securities and change in liabilities from price movement simply by computing the value scoring rule $S_G(\mathbf{p}, \cdot)$. We thus interleave both interpretations at once. Consider the following generating functions:

$$G_1(\mathbf{p}) = p_1 \log(p_1) + p_2 \log(p_2) - (p_1 + p_2) \log(p_1 + p_2),$$
$$G_2(\mathbf{p}) = p_2 \log(p_2) + p_3 \log(p_3) - (p_2 + p_3) \log(p_2 + p_3).$$

Both functions are 1-homogeneous versions of the LMSR defined on two assets. Their corresponding scoring rules are

$$S_{G_1}(\mathbf{p}, \cdot) = \nabla G_1 = \left(\log\left(\frac{p_1}{p_1 + p_2}\right), \log\left(\frac{p_2}{p_1 + p_2}\right), 0 \right),$$
$$S_{G_2}(\mathbf{p}, \cdot) = \nabla G_2 = \left(0, \log\left(\frac{p_2}{p_2 + p_3}\right), \log\left(\frac{p_3}{p_2 + p_3}\right) \right).$$

[5] written as $\hat{\mathbf{q}}_i > \epsilon$ for any $\epsilon < 0$, as liabilities are negative.

[6] with respect to a norm on the price space.

Suppose we start at an initial price vector of $\mathbf{p} = (1/3, 1/3, 1/3)$, and trade to a price of $\hat{\mathbf{p}} = (1/7, 2/7, 4/7)$. Both LPs must make an initial deposit equal to $S_{G_i}(\mathbf{p}, \cdot)$:

$$\mathbf{q}^1 = \nabla G_1(\mathbf{p}) = (\log(1/2), \log(1/2), 0), \quad \mathbf{q}^2 = \nabla G_2(\mathbf{p}) = (0, \log(1/2), \log(1/2)).$$

For each LP i, we calculate the trade $\mathbf{r}^i$ as $S_{G_i}(\hat{\mathbf{p}}, \cdot) - S_{G_i}(\mathbf{p}, \cdot)$, and then set a new liability for LP_i of $\hat{\mathbf{q}}^i = \mathbf{q}^i + \mathbf{r}^i$. Equivalently, we can calculate $\hat{\mathbf{q}}^i = S_{G_i}(\hat{\mathbf{p}}, \cdot)$, and then calculate $\mathbf{r}^i$ as the difference in liabilities $\hat{\mathbf{q}}^i - \mathbf{q}^i$. At the new price of $\hat{\mathbf{p}} = (1/7, 2/7, 4/7)$, the respective liabilities are

$$\hat{\mathbf{q}}^1 = \nabla G_1(\hat{\mathbf{p}}) = (\log(1/3), \log(2/3), 0), \quad \hat{\mathbf{q}}^2 = \nabla G_2(\hat{\mathbf{p}}) = (0, \log(1/3), \log(2/3)).$$

And the net trades $\mathbf{r}^1, \mathbf{r}^2$ are therefore:

$$\mathbf{r}^1 = \hat{\mathbf{q}}^1 - \mathbf{q}^1 = (\log(1/3), \log(2/3), 0) - (\log(1/2), \log(1/2), 0) = (\log(2/3), \log(4/3), 0),$$
$$\mathbf{r}^2 = \hat{\mathbf{q}}^2 - \mathbf{q}^2 = (0, \log(1/3), \log(2/3)) - (0, \log(1/2), \log(1/2)) = (0, \log(2/3), \log(4/3)).$$

With some fee $\text{fee}_i(\mathbf{r}^i, \mathbf{p})$ assessed for each LP i.

Note that these generating functions exhibit an intuitive property: they make a deposit of 0 and facilitate a trade of 0 in the asset they are not parameterized by. The same holds for 1-homogeneous generating functions in general. When we discuss matrix-valued liquidity in § 5.2, we will note that these functions also do not provide any liquidity for trading in those assets.

Also observe that this LP configuration is exactly equivalent to having a single LP with the generating function $G = G_1 + G_2$. For the same initial and final prices, $S_G(\mathbf{p}, \cdot) = S_{G_1}(\mathbf{p}, \cdot) + S_{G_2}(\mathbf{p}, \cdot)$, and accordingly the LP's initial liability will be $\mathbf{q} = \mathbf{q}^1 + \mathbf{q}^2$, its final liability will be $\hat{\mathbf{q}} = \hat{\mathbf{q}}^1 + \hat{\mathbf{q}}^2$, and the net trade will be $\mathbf{r} = \mathbf{r}^1 + \mathbf{r}^2$.

We now consider the convex conjugates $C_1 = G_1^*$ and $C_2 = G_2^*$, and show that the cost function protocol yields an equivalent result to the scoring rule protocol. We have

$$C_1(\mathbf{q}) = \max\{\log(e^{q_1} + e^{q_2}, q_3)\}, \quad C_2(\mathbf{q}) = \max\{\log(e^{q_2} + e^{q_3}, q_1)\}.$$

While one might expect C_1 and C_2 to be parameterized by the same 2 assets as their conjugates, the maximization term has a natural interpretation—purchasing more than a small amount of an asset for which an LP provides no liquidity will cause its price to reach 1. We can verify that the trades described above comport with the cost function protocol. Observe that

$$\begin{aligned} C_1(\mathbf{q}^1 + \mathbf{r}^1) - C_1(\mathbf{q}^1) = C_1(\hat{\mathbf{q}}^1) - C_1(\mathbf{q}^1) &= \log(e^{\log(1/3)} + e^{\log(2/3)}) - \log(e^{\log(1/2)} + e^{\log(1/2)}) \\ &= \log(1/3 + 2/3) - \log(1/2 + 1/2) = 0. \end{aligned}$$

As $C(\mathbf{q}^1 + \mathbf{r}^1) - C(\mathbf{q}^1) = 0$, $\mathbf{r}^1$ is a legal trade at liability $\mathbf{q}^1$ Both C_2 and the infimal convolution $C = C_1 \wedge C_2 = (G_1 + G_2)^*$ can be checked similarly.

4 EQUIVALENCE OF INTERPRETATIONS

As we have argued informally, one can regard liquidity provisioning as (a) recruiting multiple market makers, which then (b) process trades in parallel. We now study (b) formally, showing that five natural ways to interpret this parallelism are all equivalent. The equivalence of (a) in these interpretations, the process of recruiting market makers and securing deposits, is then straightforward (§ A.3).

4.1 Five interpretations of parallel market making

To begin, we revisit the interpretation of liquidity provisioning as running several market makers in parallel, and show that five natural interpretations of this idea lead to equivalent protocols.

In interpretations 1, 2, and 5, each market maker i is specified by a cost function C_i and a state $\mathbf{q}^i$. In interpretations 3 and 4, market makers instead each have a scoring rule S_i generated by a convex function $G_i = C_i^*$ and maintain a price $\mathbf{p}^i$. In all cases, we assume the trader behaves rationally, in the sense that the overall trade is Pareto optimal: if $\mathbf{r}, \mathbf{r}'$ are both valid trades, and $\mathbf{r} \geq \mathbf{r}'$, the trader

chooses $\mathbf{r}$. Recall that trades are oriented toward the trader, so here $\mathbf{r}$ gives the trader weakly more of each security.

In each interpretation, we capture the market state by the collection of liability vectors $\{\mathbf{q}^i \in \mathbb{R}^n\}_i$. After a trade $\mathbf{r} = \sum_i \mathbf{r}^i$, the state updates to $\{\mathbf{q}^i + \mathbf{r}^i\}_i$. The set of consistent market prices is defined to be $\partial C_i(\mathbf{q}^i)$ for interpretations 1, 2, and 5, and an analogous definition for 3 and 4. We say the overall market state is *coherent* if there is a consistent price $\mathbf{p} \in \operatorname{relint} \Delta_n$ for all market makers simultaneously.

(1) **The trader selects a valid trade from each market maker's cost function and executes them all.** Formally, for each market maker i the trader selects $\mathbf{r}^i$ such that $C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{q}^i)$, for a total trade of $\mathbf{r} = \sum_i \mathbf{r}^i$.

(2) **A centralized market maker uses the infimal convolution of cost functions.** This interpretation corresponds to the rules for trade in Protocol 1. Formally, the trader selects any $\mathbf{r} \in \mathbb{R}^n$ such that $C(\mathbf{q} + \mathbf{r}) = C(\mathbf{q})$, where $C = \bigwedge_i C_i$ and $\mathbf{q} = \sum_i \mathbf{q}^i$. The central market maker first gives $\mathbf{r}$ to the trader. Behind the scenes, it then computes a split $\mathbf{r} = \sum_i \mathbf{r}^i$ such that $C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{q}^i)$, whose existence we establish below, and executes these trades in each constituent market maker.

(3) **The trader is paid according to the sum of each market maker's scoring rule.** Formally, each market maker has a scoring rule $S_i(\mathbf{p}, y) = G_i(\mathbf{p}) + \left\langle \mathbf{d}G_{\mathbf{p}}, \boldsymbol{\delta}^y - \mathbf{p} \right\rangle$ where $G_i = C_i^*$ and $\{\mathbf{d}G_{\mathbf{p}} \in \partial G(\mathbf{p}) \mid \mathbf{p} \in \operatorname{relint} \Delta_n\}$ is an arbitrary selection of subgradients. Market maker i maintains a price vector $\mathbf{p}^i \in \operatorname{relint} \Delta_n$, and the trader may choose any $\hat{\mathbf{p}}^i \in \operatorname{relint} \Delta_n$, resulting in the trade $\mathbf{r}^i = S_i(\hat{\mathbf{p}}^i, \cdot) - S_i(\mathbf{p}^i, \cdot) \in \mathbb{R}^n$. See Protocol 2. For the purposes of comparing interpretations, define the set of consistent prices as $\{\mathbf{p} \in \operatorname{relint} \Delta_n \mid \forall i\, S_i(\mathbf{p}, \cdot) = S_i(\mathbf{p}^i, \cdot)\}$.

(4) **A centralized market maker chooses the generating function $G = \sum_{i=1}^k G_i$ and the corresponding scoring rule $S_G = \sum_{i=1}^k S_{G_i}$.** The market maker maintains a price vector $\mathbf{p} \in \operatorname{relint} \Delta_n$, and the trader may choose any $\hat{\mathbf{p}} \in \operatorname{relint} \Delta_n$, resulting in the trade $\mathbf{r} = S_G(\hat{\mathbf{p}}, \cdot) - S_G(\mathbf{p}, \cdot) \in \mathbb{R}^n$.

(5) **The trader continuously trades at the most favorable price and at some point stops.** Recall that we can interpret a cost function C_i as quoting a cost $C_i(\mathbf{q}^i + \mathbf{v}^i) - C_i(\mathbf{q}^i)$ for each bundle of assets/securities $\mathbf{v}^i \in \mathbb{R}^n$. Formally, in this interpretation, the trader specifies a direction $\mathbf{v} \in \mathbb{R}^n$, and a stopping point α, and continuously purchases $\mathbf{v}dt$ for the smallest price $C_i'(\mathbf{q}^i; \mathbf{v})$ over all i, for α units of time. Here $C_i'(\mathbf{q}^i; \mathbf{v}) := \lim_{h \to 0^+} \frac{C_i(\mathbf{q}^i + h\mathbf{v}) - C_i(\mathbf{q}^i)}{h}$ is the directional derivative of C_i. In other words, the trades are of the form $(\mathbf{v} - C_i'(\mathbf{q}^i; \mathbf{v})\mathbf{1})dt$, where we recall that the numeraire is simply $\mathbf{1}$. Crucially, we also allow the trader to take advantage of any arbitrage opportunity that arises from this continuous trade: after the α units of time, the trader may place trades $\{\hat{\mathbf{r}}^i\}_i$ if they have negative net cost and $\sum_i \hat{\mathbf{r}}^i = \mathbf{0}$.

4.2 Equivalence of the interpretations

Before stating the equivalence of these interpretations, we must address an important technical point. Recall the definitions in § 2.4. By our assumptions on G, the resulting scoring rule vectors are unique for each price $\mathbf{p} \in \operatorname{relint} \Delta_n$; see Lemma 6 in § A. But, on the relative boundary of the simplex Δ_n, uniqueness can fail. If we take $G(\mathbf{p}) = \|\mathbf{p}\|_2^2$, or any LP that does not provide infinite liquidity at the boundary of Δ_n, the resulting G will not have a unique subgradient (even modulo $\mathbf{1}$) at those boundary points, meaning we will have $\mathbf{q}, \hat{\mathbf{q}} \in \mathbb{R}^n$ with $\mathbf{p} \in \partial C(\mathbf{q}) \cap \partial C(\hat{\mathbf{q}})$, but with $\hat{\mathbf{q}} - \mathbf{q} \neq \alpha\mathbf{1}$ for any α. The scoring rule S must pick just one of these vectors, meaning the scoring rule market (Interpretation 4) will be strictly less expressive than the others. Somewhat conversely, consider G to be negative Shannon entropy, which gives rise to the log scoring rule $S(\mathbf{p}, y) = \log p_y$. Here the liquidity does become infinite on the boundary, and consequently the scoring rule vectors have infinite entries. These vectors cannot be captured by any $\mathbf{q} \in \mathbb{R}^n$, only in the limit.

For these two reasons, we restrict to relint Δ_n in Lemma 6. The first issue is somewhat surmountable, however: if one defines $\mathbf{d}G_{\mathbf{p}}$ to be the most favorable $\mathbf{q}$ (modulo $\mathbf{1}$) tangent to G at a boundary point $\mathbf{p}$, then all of the Pareto optimal trades will still be available to the scoring rule market trader. Indeed, the only trades missing are those at the maximum possible price, which is Pareto-suboptimal for the trader anyway—consider a two outcome example when the price of the first security is 1, the maximum possible, so that purchasing the first security at this price is weakly worse than simply refraining from trade. Thus, while in Theorem 1 we assume there is a "log-like" LP, typically the market creator, which keeps the price away from the boundary, in principle one could generalize this statement using the ideas above to the case where liquidity runs out.

Theorem 1. *Let $C_i = G_i^*$ for generating functions G_i, where at least one G_i is a pseudobarrier. Then the interpretations 1-5 above are equivalent in the following sense: given a coherent market state, the set of valid trades is identical, and the resulting market state is coherent.*

Implicit in the proof of Theorem 1 is that, in interpretations 1, 3, and 5, if the market state is not initially coherent, it becomes coherent after a sufficiently large trade.

5 DEFINING LIQUIDITY AS MATRIX-VALUED PRICE INSENSITIVITY

Informally, liquidity is the extent to which assets/securities can be exchanged. One way to capture liquidity, locally around a given price, is to quantify the extent to which the price stays stable during a transaction. In other words, the lower the rate of change of the price, the higher the liquidity.

Thus far in the prediction market literature, even for large numbers of securities, liquidity is often captured by a single parameter. The most popular approach is to consider some base cost function C and define C_η via the *perspective transform* $C_\eta(\mathbf{q}) = \eta C(\mathbf{q}/\eta)$, where η corresponds to the liquidity of the market [Abernethy et al., 2014, Li and Vaughan, 2013, Othman and Sandholm, 2011, Othman et al., 2013]. Another approach is to define the liquidity of C to be some function of its Hessian $\nabla^2 C$ such as the inverse of its norm [Abernethy et al., 2013]. A noted exception is Dudík et al. [2014], where liquidity is acknowledged to depend on which specific subset of securities is under consideration. Our approach is closest to the latter: as we argue soon in § 5.2, liquidity is indeed inherently multidimensional. Any subspace of securities could have any degree of liquidity. It is true that for very restricted protocols such as Uniswap V2, which corresponds to the perspective transform, liquidity has a fixed shape that can be scaled by a single real parameter. In general, however, liquidity must be captured by a higher-dimensional object. We begin by considering the case of 2 securities and constructing a scalar measure of liquidity that naturally corresponds to price insensitivity. Then, we construct a matrix-valued measure of liquidity that recovers our scalar measure for the 2-security case. We show that we can use our matrix-valued liquidity measure to recover a scalar liquidity measure *in the direction* of any trade.

5.1 Scalar-valued liquidity for 2 securities

Consider a prediction market with 2 securities. We first note that by $\mathbf{1}$-invariance, a 2-dimensional cost function may be specified by a 1-dimensional convex function. Given any $\mathbf{1}$-invariant cost function $C : \mathbb{R}^n \to \mathbb{R}$, we may write $C(\mathbf{q}) = c(q_1 - q_2) + q_2$ for some convex $c : \mathbb{R} \to \mathbb{R}$ given by $c(q) = C((q, 0))$. Letting $\mathbf{q} = (q, 0)$, then, the price of security 1 is $\nabla C(\mathbf{q}) = c'(q)$. Note that we additionally use the lowercase variants of C and G in § 7.

Appealing to the above notion of liquidity as a function of the local price, let us define the liquidity at price p to be the reciprocal of the rate of change of the price when the price is p. While not required in general, for the purpose of this derivation, suppose that $c'' > 0$ everywhere. Then we may define the liquidity at price $p \in [0, 1]$ as $\ell(p) = 1/c''(q) > 0$, where $p = c'(q)$. Since ℓ is strictly positive, we find that $\ell(p) = g''(p)$ for some convex function $g : [0, 1] \to \mathbb{R}$. This relationship is in essence a special case of convex conjugate duality: we may simply take $g = c^*$. From this duality, we have

$g' = (c')^{-1}$, which is well-defined as c' is strictly monotone; by the inverse function theorem, we could equivalently derive ℓ as $\ell(p) = ((c')^{-1})'(p) = g''(p)$.

5.2 Liquidity as a Hessian matrix

In the 2-security case, we showed that liquidity can be measured as $\ell(p) = 1/c''(q) = g''(p)$, adopting the 1-dimensional expression of the market. In higher dimensions, we can analogously define the liquidity at price $\mathbf{p}$ to be $\ell(\mathbf{p}) = (\nabla^2 C(\mathbf{q}))^+$ at any vector $\mathbf{q}$ with price $\nabla C(\mathbf{q}) = \mathbf{p}$, where A^+ is the pseudoinverse of A.[7] Here, liquidity is a matrix, which specifies the (inverse) rate of change of the price in any direction (or more generally, subspace) of interest. Again appealing to convex duality, we can write $\mathbf{q} = \nabla\overline{G}(\mathbf{p})$, where $\overline{G}$ is the 1-homogeneous extension of the dual function $G = C^*$. Thus, we have $\ell(\mathbf{p}) = \left(\nabla^2 C(\nabla\overline{G}(\mathbf{p}))\right)^+$, or equivalently, $\ell(\mathbf{p}) = \nabla^2\overline{G}(\mathbf{p})$. We may recover the usual measure of scalar liquidity as price insensitivity in the direction of any price $\hat{\mathbf{p}} \neq \mathbf{p}$. Let $H(\mathbf{p}) = \nabla^2(\overline{G}(\mathbf{p}))$, and consider the product $(\hat{\mathbf{p}})^\top H(\mathbf{p})\hat{\mathbf{p}}$. This gives us the second derivative with respect to t of the function $g(t) = f(\mathbf{p} + t\hat{\mathbf{p}})$, representing the price insensitivity in the direction of $\hat{\mathbf{p}}$ locally at $\mathbf{p}$. Consider again the generating functions used in the example from § 3.6:

$$G_1(\mathbf{p}) = p_1 \log(p_1) + p_2 \log(p_2) - (p_1 + p_2)\log(p_1 + p_2),$$
$$G_2(\mathbf{p}) = p_2 \log(p_2) + p_3 \log(p_3) - (p_2 + p_3)\log(p_2 + p_3).$$

As G_1 and G_2 are already 1-homogeneous, they are equal to their 1-homogeneous extensions and we may take their Hessians directly. Respectively, these are

$$H_1 = \nabla^2 G_1 = \begin{bmatrix} \frac{1}{p_1} - \frac{1}{p_1+p_2} & -\frac{1}{p_1+p_2} & 0 \\ -\frac{1}{p_1+p_2} & \frac{1}{p_2} - \frac{1}{p_1+p_2} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{p_1} & 0 & 0 \\ 0 & \frac{1}{p_2} & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} \frac{1}{1-p_3} & \frac{1}{1-p_3} & 0 \\ \frac{1}{1-p_3} & \frac{1}{1-p_3} & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$H_2 = \nabla^2 G_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{p_2} - \frac{1}{p_2+p_3} & -\frac{1}{p_2+p_3} \\ 0 & -\frac{1}{p_2+p_3} & \frac{1}{p_3} - \frac{1}{p_2+p_3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{p_2} & 0 \\ 0 & 0 & \frac{1}{p_3} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{1-p_1} & \frac{1}{1-p_1} \\ 0 & \frac{1}{1-p_1} & \frac{1}{1-p_1} \end{bmatrix}.$$

Note that G_1 is flat in the direction of $\hat{\mathbf{p}} = (0, 0, 1)$, and G_2 is flat in the direction of $\hat{\mathbf{p}} = (1, 0, 0)$. As we observed in our earlier example, neither function provides any liquidity to trades involving those assets. But, by adding the generating functions, and therefore adding their Hessians, we get

$$\nabla^2 G = \begin{bmatrix} \frac{1}{p_1} - \frac{1}{p_1+p_2} & -\frac{1}{p_1+p_2} & 0 \\ -\frac{1}{p_1+p_2} & \frac{2}{p_2} - \frac{1}{p_1+p_2} - \frac{1}{p_2+p_3} & -\frac{1}{p_2+p_3} \\ 0 & -\frac{1}{p_2+p_3} & \frac{1}{p_3} - \frac{1}{p_2+p_3} \end{bmatrix},$$

and liquidity is present for all assets.

6 AN IMPOSSIBILITY RESULT ON THE DESIGN OF TRADING FEES

Our protocol calls for assessing a cash fee to be paid by the trader and to each LP after each trade, parameterized by the trade vector, liability, and cost/generating function of the LP. We do not inherently place any other restrictions on the fee function.

[7] The pseudoinverse is needed as the Hessian $\nabla^2 C$ is rank-deficient since C is always flat in the $\mathbf{1}$ direction. This observation also explains why we can express liquidity between 2 securities in one real number, since there is only 1 free parameter in $\nabla^2 C$ in that case.

6.1 Natural fees "break" when $n \geq 3$

In practice, the design of fees has not been studied for markets with $n \geq 3$ securities. Commonly used fees, like those used in Uniswap, exhibit deeply undesirable behavior when extended to these markets; see Appendix B.1. Here, we enumerate a set of axioms describing desirable behavior of the fee function. Unfortunately, we show that it is impossible for a fee function to satisfy all properties. We denote the fee that LP i receives by $\mathsf{fee}_i(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}) \in \mathbb{R}$. Let the fee that a trader pays be given by $\mathsf{fee}_\mathrm{T}(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}) \in \mathbb{R}$.

Axiom 1 (Budget Balanced, (BB)). *The sum of the fees collected by the LPs must be equal to the fee that a trader pays. That is, for all* $\{\mathbf{r}^i\}, \{\mathbf{q}^i\}$,

$$\mathsf{fee}_\mathrm{T}(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}) = \sum_{i=1}^{k} \mathsf{fee}_i(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}).$$

Axiom 2 (Trader Simple, (TS)). *The fee paid by a trader should be independent of how the trade is split across LPs and their individual liquidities and should depend only on the aggregate trade* $\mathbf{r}$ *and aggregate liquidity* $\mathbf{q}$*. The axiom captures the goal of keeping the trader interface simple and abstracted from the inner dynamics of LPs. Hence, we require that there exists* $\overline{\mathsf{fee}}_\mathrm{T}$ *such that for all* $\{\mathbf{r}^i\}, \{\mathbf{q}^i\}$,

$$\mathsf{fee}_\mathrm{T}(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}) = \overline{\mathsf{fee}}_\mathrm{T}(\mathbf{r}, \mathbf{q})$$

whenever $\mathbf{r} = \sum_i \mathbf{r}^i$ *and* $\mathbf{q} = \sum_i \mathbf{q}^i$.

Axiom 3 (LP Decomposability, (LD)). *The fee LP* i *charges should only depend on inputs* $\mathbf{r}^i, \mathbf{q}^i$ *to enable the parallel market interpretation proposed in § 4.1. Hence there exists* fee_LP *such that for all* $i \in \{1, \ldots, k\}$,

$$\mathsf{fee}_i(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}) = \mathsf{fee}_\mathrm{LP}(\mathbf{r}^i, \mathbf{q}^i).$$

Axiom 4 (Nonnegativity, (NN)). *All fees must be nonnegative, so for all* $\{\mathbf{r}^i\}, \{\mathbf{q}^i\}$,

$$\mathsf{fee}_i(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}), \mathsf{fee}_\mathrm{T}(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}) \in \mathbb{R}^n_{\geq 0}$$

and $\mathsf{fee}_i(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}), \mathsf{fee}_\mathrm{T}(\{\mathbf{r}^i\}, \{\mathbf{q}^i\}) = 0$ *if and only if* $\mathbf{r}^i = \mathbf{0}$ *or* $\mathbf{r} = \mathbf{0}$ *respectively.*

The above axioms are sufficient for our impossibility result; see § 6.2 for an additional Axiom 5. We start by stating a useful lemma. See Appendix B for proofs.

Lemma 1. *Axioms 1, 2, and 3 allow us to write* $\overline{\mathsf{fee}}_\mathrm{T} = \mathsf{fee}_\mathrm{LP}$.

Due to Lemma 1, we use $\mathsf{fee}()$ from now on in the place of functions $\overline{\mathsf{fee}}_\mathrm{T}()$ and $\mathsf{fee}_\mathrm{LP}()$. We now state our impossibility result, providing a proof sketch but deferring the full constructive proof to the appendix.

Theorem 2. *Axioms 1, 2, 3, and 4 are incompatible.*

Proof Sketch. We approach the problem from the market scoring rule point of view, considering a 3 asset instance. We consider the three generating functions $G_1 = -2\sqrt{2p_2p_3}, G_2 = -2\sqrt{2p_1p_3}$, and $G_3 = -2\sqrt{2p_1p_2}$, which are symmetric (i.e., equal with each other up to permutation of assets), 1-homogeneous, and flat with respect to one asset. We consider a setting in which LPs with generating functions G_1 and G_2 are present. We set initial and final prices such that the trades facilitated are $\mathbf{r}^1 = (0, -1, 1)$ and $\mathbf{r}^2 = (1, 0, -1)$, and therefore the overall trade is $\mathbf{r}^1 + \mathbf{r}^2 = (1, -1, 0)$. We then consider a setting in which only LP_3 with G_3 is present, and set prices to create a direct trade of $\mathbf{r} = \mathbf{r}^3 = (1, -1, 0)$. We add "filler" LPs to both settings that add liquidity so that the starting liability vector is the same in both settings, but are flat with respect to both asset 1 and asset 2, and therefore do not participate in the trade. By Axiom 4, the fees that these "filler" LPs collect is $\mathbf{0}$, since the trade facilitated by these LPs is $\mathbf{0}$.

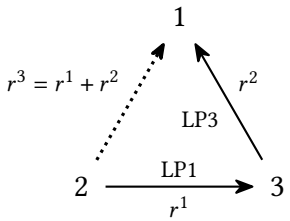
In both settings, the trader makes a trade of $(1, -1, 0)$, with the same starting liability $\mathbf{q}$, and so pays the same fee. By Lemma 1, we have

$$\overline{\mathsf{fee}}_T(\mathbf{r}, \mathbf{q}) = \mathsf{fee}_1(\mathbf{r}^1, \mathbf{q}) + \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}), \quad \overline{\mathsf{fee}}_T(\mathbf{r}, \mathbf{q}) = \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}).$$

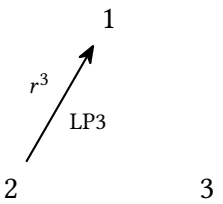
Now, we consider an alternative pair of scenarios, in which we permute the assets and the roles of the LPs. With LP_1 and LP_3 present, we set prices so that $\mathbf{r}^1 = (0, 1, -1)$, the negative of its original value, $\mathbf{r}^3 = (1, -1, 0)$, and the total trade is $(1, 0, -1)$. In the second scenario, LP_2 facilitates a trade of $\mathbf{r}^2 = (1, 0, -1)$. Note that both $\mathbf{r}^2$ and $\mathbf{r}^3$ have the same value throughout. We again introduce "filler" LPs so that the overall initial liability is the same as in the previous pair of settings. We then have

$$\overline{\mathsf{fee}}_T(\mathbf{r}, \mathbf{q}) = \mathsf{fee}_1(-\mathbf{r}^1, \mathbf{q}) + \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}), \quad \overline{\mathsf{fee}}_T(\mathbf{r}, \mathbf{q}) = \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}).$$

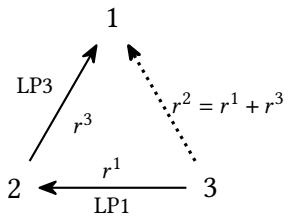
So we have that $\mathsf{fee}_1(\mathbf{r}^1, \mathbf{q}) + \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}) = \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q})$ and $\mathsf{fee}_1(-\mathbf{r}^1, \mathbf{q}) + \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}) = \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q})$. But by Axiom 4, all these fees are strictly positive, and so this is impossible. □



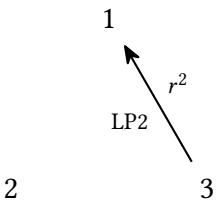
(a) trades r^1 and r^2 facilitated by LP1 and LP2 add to r^3.



(b) Trade r^3 facilitated directly by LP3.



(c) Trades r^1 and r^3 facilitated by LP1 and LP3 add to r^2.



(d) Trade r^2 facilitated directly by LP2

Fig. 1. A set of trades that, with the starting liquidity set to be equal in all cases, results in $\mathsf{fee}_3 = \mathsf{fee}_1 + \mathsf{fee}_2$ and $\mathsf{fee}_2 = \mathsf{fee}_1 + \mathsf{fee}_3$, violating Axiom 4.

6.2 Fee design in practice and future work

In light of our impossibility result, what fee structures are best in practice? Many markets use $\mathsf{fee}_i(\mathbf{r}^i, \mathbf{q}^i) = \beta\|\mathbf{r}^i\|_1$ or $\mathsf{fee}_i(\mathbf{r}, \mathbf{q}) = \beta\|\mathbf{r}\|\frac{\|\mathbf{r}^i\|}{\sum_j \|\mathbf{r}^j\|}$ for some $0 < \beta < 1$, or some minor variant thereof. Uniswap uses $\mathsf{fee}_i(\mathbf{r}^i, \mathbf{q}^i) = \beta(-\mathbf{r}^i)_+$. We provide a worked example of Uniswap's fee in Appendix B.1. While these fees violate trader simplicity (Axiom 2), any violation of the sort we construct in the proof of Theorem 2 is "reasonable" in the sense that the trader must pay a higher fee for an "indirect" trade that requires the participation of multiple LPs. Axiom 5, given below, is also violated, but only when the difference between the trader's belief and the price for any security they are trading is less than β.

A possible approach is to allow fees that are a function not only of $\mathbf{r}$ and $\mathbf{q}$ but also the cost function/scoring rule set by the liquidity provider. Surprisingly, adding this additional parameter allows us to overcome the impossibility result of Theorem 2 and satisfy Axioms 1 through 4. However, while such fees do not violate Axiom 2 as written, they violate the spirit of the axiom, as the trader fee depends on the configuration of LPs. Despite this concern, such fees exist whose dependence on G can be reduced to trader-interpretable quantities such as $\mathbf{p}$ and $\hat{\mathbf{p}}$. One such fee is $\mathbf{r} \cdot (\hat{\mathbf{p}} - \mathbf{p})$, which we use to prove that a fee function parameterized by G may satisfy Axioms 1 through 4.

Proposition 2. *A fee function of the form* $\mathsf{fee}(\mathbf{r}, \mathbf{q}, G)$ *may satisfy Axioms 1 through 4.*

Proof. To show that Axioms 1, 2 and 3 hold, first note that for any fixed $\mathbf{q}$, $\mathbf{p}$, legal trade $\mathbf{r}$, and set of LPs with generating functions $\{G_i\}$, the final price $\hat{\mathbf{p}}$ after performing trade $\mathbf{r}$ is exactly determined by $\mathbf{q}, \mathbf{p}$, and $\{G_i\}$. We then observe that

$$\mathsf{fee}(\mathbf{r}, \mathbf{q}) = \mathbf{r} \cdot (\hat{\mathbf{p}} - \mathbf{p}) = \sum_{i=1}^{k} \mathbf{r}^i \cdot (\hat{\mathbf{p}} - \mathbf{p}) = \sum_{i=1}^{k} \mathsf{fee}_i(\mathbf{r}^i, \mathbf{q}^i).$$

To show that Axiom 4 is satisfied, note that whenever an element r_j of $\mathbf{r}$ is positive, so is $\hat{p}_j - p_j$, and whenever r_j is negative, so is $\hat{p}_j - p_j$, and so $\mathbf{r} \cdot (\hat{\mathbf{p}} - \mathbf{p}) = \sum_{j=1}^{n} r_j(\hat{p}_j - p_j) \geq 0$, with equality exactly when $\mathbf{r} = \mathbf{0}$. □

Axiom 5 (Profitability, (PF)). *If* $\mathbf{r}$ *is a legal trade given* $\mathbf{q}$ *and* C*, then* $\mathbf{r}^* := \sum_i \max\{r_i, 0\}$ *is the maximum possible profit that a trader can make off of* $\mathbf{r}$*, if all acquired securities are sold at price 1. Then* $\mathsf{fee}_T(\{\mathbf{r}\}, \{\mathbf{q}\}) \leq \mathbf{r}^*$.

However, this fee still exhibits very undesirable behavior. A trader may reduce their fee by dividing their trade into smaller sub-trades, each of which creates a smaller difference in price. Indeed, for sufficiently large trades, the fee can be arbitrarily large, even though the maximum profit of any trade is bounded by the deposited liquidity, severely violating Axiom 5. For example, given a pseudobarrier generating function G on three assets, any trade $\mathbf{r}$ to $\hat{\mathbf{p}} = (1 - \epsilon, 0, 0)$ will be bounded in the first term by the maximum deposit of G but approach $-\infty$ in the other two terms. Therefore, for any starting price $\mathbf{p}$, the fee for trading to $(1 - \epsilon, 0, 0)$ grows arbitrarily large as $\epsilon \to 0$, while the trader's maximum profit is bounded above by $-\min_{\mathbf{q} \in \Delta_3} G(\mathbf{q})$. Ultimately, these concerns nullify any practical use for the fee. The design of simple, liquidity-dependent fees that are more practical for future use is a promising direction of future work.

7 RECOVERING AND EXTENDING PROTOCOLS FROM DECENTRALIZED FINANCE

In this section, we discuss our contributions to the decentralized finance literature. We recover several common protocols, like Uniswap V2 and V3, as special instances of our general protocol. For this section, we restrict to the exchange of two securities, as is common in decentralized finance. In Appendix C.1, we provide a general protocol (Protocol 3) for this setting. In Appendix D, we use the flexibility of our general protocols to propose new ones, to be used either in decentralized finance or for prediction markets.

7.1 Conventional differences

There are many conventional differences to note between the prediction market and the decentralized finance AMM literature. For example, with regard to prices, decentralized finance typically uses an "exchange-rate" version of the contract price: the rate at which one can exchange one asset for another. Taking advantage of the structure of cost functions, we instead adopt a *normalized price* convention. One can view normalized prices $\mathbf{p} \in \Delta_n$ as an exchange rate between assets and the "grand bundle" $\mathbf{1}$ of all assets; p_i denotes the instantaneous price, in units of $\mathbf{1}$, to purchase asset i.

Converting between the two conventions is straightforward. Given normalized prices $\mathbf{p}$, one can simply define the exchange rate between i and j as $\hat{p}_{ij} = p_i/p_j$. Conversely, given pairwise exchange rates, one can define $\mathbf{x} = (1, \hat{p}_{21}, \hat{p}_{31}, \ldots, \hat{p}_{n1})$ and take $\mathbf{p} = \mathbf{x}/\|\mathbf{x}\|_1$. The conversion simplifies in the case of two assets, as $\hat{p}_{12} = \frac{p}{1-p}$ and $p = \hat{p}_{12}/(\hat{p}_{12} + 1)$.

In the decentralized finance literature, trades are also typically oriented toward the market maker, whereas we instead consider all trades to be oriented towards the trader. Also, CFMMs in decentralized finance track the reserves held by the market maker, while cost function prediction markets typically track their liabilities as a function of the eventual outcome. So, a vector $\mathbf{x}$ of reserves corresponds to a vector $-\mathbf{q}$ of liabilities.

7.2 Uniswap V2

Uniswap V2, introduced by Adams et al. [2020], is a commonly used AMM in the Ethereum ecosystem to trade assets and has the functional invariant $\varphi_\alpha(\mathbf{x}) = x_1 x_2 = \alpha^2$. We track the reserve vector $\mathbf{x}$, so that x_1 and x_2 represent the amount of assets 1 and 2, respectively, held by the market maker. We also still use normalized prices, leading to minor differences from the literature. The normalized price of the first asset can be computed as $p = \frac{x_2}{x_1+x_2}$, while the exchange rate model has $\hat{p} = \frac{x_2}{x_1}$. We refer the reader to Fan et al. [2022, 2023] for a detailed breakdown of Uniswap V2 mechanics.

Uniswap V2 restricts how an LP can add or remove their liquidity by constraining them to use the same base function $g_0 = -2\sqrt{p(1-p)}$ and to express their liquidity only via a parameter α^i where $g_i = \alpha^i g_0$. We state the Uniswap V2 protocol as Protocol 4 explicitly in Appendix C.2. Recall that from Fan et al. [2022], the bundle required to change liquidity from α^i to α' while keeping the price invariant is $\left(\frac{\alpha'-\alpha^i}{\sqrt{\hat{p}}}, (\alpha'-\alpha^i)\sqrt{\hat{p}}\right)$. Using normalized prices, this is represented as $\left((\alpha'-\alpha^i)\sqrt{\frac{1-p}{p}}, (\alpha'-\alpha^i)\sqrt{\frac{p}{1-p}}\right)$ in our protocol. We note that instead of skimming γ from $(-\mathbf{r})_+$ for trading fees, we ask for $\beta(-\mathbf{r})_+$ from the trader when they request the trade $\mathbf{r}$. These two fee schemes are equivalent when $\beta = \frac{\gamma}{1+\gamma}$.

Lemma 2 (Informal). *Protocol 4 is a special case of Protocol 3 for specific restrictions on $\mathcal{G}_{\text{init}}$.*

We defer the formal statements and proofs to Appendix C. There, we show that the liability vectors from the latter indeed satisfy the constant product invariant for our choice of generating functions using Propositions 6 and 7.

7.3 Uniswap V3 and general bucketing

Uniswap V2 requires LPs to provide liquidity on the entire price space. This restriction may look intuitive, but it is suboptimal since liquidity allocations far from the current price may not be used. For example, a market that trades securities on a sports game might not benefit from having liquidity at prices in the $(0, 0.1)$ range, say. Moreover, when LPs provide liquidity, they take on the risk of price volatility, and ideally we would like to allow them to bound that risk. On the other hand, it is computationally challenging to maintain an infinitely flexible LP protocol.

Motivated by these concerns, the Uniswap V3 protocol, [Adams et al., 2021], partitions the price space, allowing each LP i to contribute a proportion α^{ij} of liquidity on any price bucket $[a_j, b_j]$ of their choosing. We refer the reader to Fan et al. [2022, 2023] for a detailed analysis of Uniswap V3. In Appendix C.5 we show that Uniswap V3, given by Protocol 5, is a special case of our general protocol.

In this section, we *generalize* the idea of bucketing with regards to an arbitrary cost function C. For ease of exposition, let us restrict again to the two-outcome setting. We do so as this allows for LPs to be more expressive, depending on the number of price intervals and also keeps the complexity of implementation from blowing up. Let $G(\mathbf{p})$ indicate the corresponding dual where $p_2 = 1 - p_1$. Let $\ell^{(j)}$ be its liquidity function restricted to the j-th price interval $[a_j, b_j]$ of p_1 and be given by

$\ell^{(j)} = \nabla^2 \overline{G} \mathbf{1}_{p_1 \in [a_j, b_j]}$. Let the liability vector associated with the corresponding cost function dual for $\ell^{(j)}$ be given below, which we derive in Appendix C.3.

$$\text{liability}(C^{(j)}) = \begin{pmatrix} S_g(\max\{a_j, p_1\}, 1) - S_g(\max\{b_j, p_1\}, 1) \\ S_g(\min\{b_j, p_1\}, 0) - S_g(\min\{a_j, p_1\}, 0) \end{pmatrix}$$

where $\mathbf{p} = (p_1, p_2)$ is the current price, $S_g(p, y) = g(p) + g'(p)(y - p)$ and $G(\mathbf{p}) = g(p_1)$ as discussed in § 2.3, § 5 and $C^{(j)} = (\iint \ell^{(j)})^*$.

In Table 1, we use the above general liability vector to state the liability vectors for Uniswap V3, as well as new bucketing protocols where we use $G(\mathbf{p}) = p_1 \log p_1 + p_2 \log p_2$ from LMSR and $S_g(p, y) = -(p - y)^2$ of the Brier scoring rule as the base "shapes." We defer detailed workings for the LMSR bucketing protocol to Appendix C.4.

	Uniswap V3	LMSR	Brier
$p < a_j$	$\alpha_j \begin{pmatrix} \sqrt{\frac{1-b_j}{b_j}} - \sqrt{\frac{1-a_j}{a_j}} \\ 0 \end{pmatrix}$	$\begin{pmatrix} \log \frac{a_j}{b_j} \\ 0 \end{pmatrix}$	$\begin{pmatrix} (1-b_j)^2 - (1-a_j)^2 \\ 0 \end{pmatrix}$
$p \in [a_j, b_j]$	$\alpha_j \begin{pmatrix} \sqrt{\frac{1-b_j}{b_j}} - \sqrt{\frac{1-p}{p}} \\ \sqrt{\frac{a_j}{1-a_j}} - \sqrt{\frac{p}{1-p}} \end{pmatrix}$	$\begin{pmatrix} \log \frac{p}{b_j} \\ \log \frac{1-p}{1-a_j} \end{pmatrix}$	$\begin{pmatrix} (1-b_j)^2 - (1-p)^2 \\ a_j^2 - p^2 \end{pmatrix}$
$p > b_j$	$\alpha_j \begin{pmatrix} 0 \\ \sqrt{\frac{a_j}{1-a_j}} - \sqrt{\frac{b_j}{1-b_j}} \end{pmatrix}$	$\begin{pmatrix} 0 \\ \log \frac{1-b_j}{1-a_j} \end{pmatrix}$	$\begin{pmatrix} 0 \\ a_j^2 - b_j^2 \end{pmatrix}$

Table 1. Liability vectors for bucketing liquidity protocols.

8 DISCUSSION AND OPEN DIRECTIONS

We have given a general protocol for liquidity provisioning in prediction markets (§ 3), and more broadly any automated market making setting including decentralized finance. In a sense that we formalize in § 4 and § 5, this protocol is maximally expressive, though it can be restricted for computational convenience, or LP ease-of-use. We also provide an impossibility result on the design of trading fees (§ 6). One avenue for future work is the design of simple liquidity-dependent fees that can work in practice. Another direction is to further explore specific restrictions, and study the tradeoffs in expressiveness and computation, and the implications for market efficiency.

When implementing Protocol 1 in practice, there is a tradeoff between LPs' expressiveness and the computational cost of running the protocol. This tradeoff is a strong consideration in decentralized finance, as all computations must be done on-chain. As prediction markets are typically not as constrained, much more expressive protocols are possible. For example, one could allow LPs to specify their liquidity functions via polynomials of bounded degree, or weighted sums of basis functions, or any computationally convenient class of functions that well approximate any possible liquidity allocation. In decentralized finance, Protocol 2 seems to be the more practically appealing of the two. Instead of specifying a trade directly, the "inversion" from proposed trades to implied prices is handled off-chain by the trader. The remaining computation, of the implied liability vectors, is more straightforward. We leave the analysis of further restrictions and tradeoffs to future work.

REFERENCES

Jacob Abernethy, Yiling Chen, and Jennifer Wortman Vaughan. Efficient market making via convex optimization, and a connection to online learning. *ACM Transactions on Economics and Computation*, 1(2):12, 2013. URL http://dl.acm.org/citation.cfm?id=2465777.

Jacob Abernethy, Rafael M. Frongillo, Xiaolong Li, and Jennifer Wortman Vaughan. A General Volume-parameterized Market Making Framework. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC '14, pages 413–430, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2565-3. doi: 10.1145/2600057.2602900. URL http://doi.acm.org/10.1145/2600057.2602900.

Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core, 2020. URL https://uniswap.org/whitepaper.pdf.

Hayden Adams, Noah Zinsmeister, Moody Salem, River Keefer, and Dan Robinson. Uniswap v3 core, 2021. URL https://uniswap.org/whitepaper-v3.pdf.

Gabriel P. Andrade. Lmsr (logarithmic market scoring rule). https://blog.gensyn.ai/lmsr-logarithmic-market-scoring-rule/, December 2025. Gensyn Blog, published 10 Dec. 2025.

Guillermo Angeris and Tarun Chitra. Improved price oracles: Constant function market makers. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 80–91, 2020.

Guillermo Angeris, Alex Evans, Tarun Chitra, and Stephen Boyd. Optimal routing for constant function market makers. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, EC '22, page 115–128, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391504. doi: 10.1145/3490486.3538336. URL https://doi.org/10.1145/3490486.3538336.

Guillermo Angeris, Tarun Chitra, Theo Diamandis, Alex Evans, and Kshitij Kulkarni. The geometry of constant function market makers. *arXiv preprint arXiv:2308.08066*, 2023.

Guillermo Angeris, Theo Diamandis, and Ciamac C. Moallemi. Multidimensional blockchain fees are (essentially) optimal, 2024. URL https://arxiv.org/abs/2402.08661.

Franz Aurenhammer. Power diagrams: Properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. doi: 10.1137/0216006. URL https://doi.org/10.1137/0216006.

Massimo Bartoletti, James Hsin-yu Chiang, and Alberto Lluch-Lafuente. A theory of automated market makers in defi. *Logical Methods in Computer Science*, Volume 18, Issue 4, 2022. ISSN 1860-5974. doi: 10.46298/lmcs-18(4:12)2022. URL http://dx.doi.org/10.46298/lmcs-18(4:12)2022.

Base. Gas use in ethereum transactions, 2025. URL https://docs.base.org/base-learn/docs/introduction-to-ethereum/gas-use-in-eth-transactions/. Retrieved 2/8/2025.

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950. ISSN 1520-0493.

Steven Campbell, Philippe Bergault, Jason Milionis, and Marcel Nutz. Optimal fees for liquidity provision in automated market makers, 2025. URL https://arxiv.org/abs/2508.08152.

Yiling Chen and David M. Pennock. A utility framework for bounded-loss market makers. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 49–56, 2007.

Yiling Chen, Mike Ruberry, and Jennifer Wortman Vaughan. Cost function market makers for measurable spaces. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 785–802, 2013. URL http://dl.acm.org/citation.cfm?id=2482608.

Theo Diamandis, Max Resnick, Tarun Chitra, and Guillermo Angeris. An efficient algorithm for optimal routing through constant function market makers, 2023. URL https://arxiv.org/abs/2302.04938.

Miroslav Dudík, Sebastien Lahaie, and David M. Pennock. A tractable combinatorial market maker using constraint generation. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, page 459–476, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450314152. doi: 10.1145/2229012.2229047. URL https://doi.org/10.1145/2229012.2229047.

Miroslav Dudík, Rafael Frongillo, and Jennifer Wortman Vaughan. Market Making with Decreasing Utility for Information. *Conference on Uncertainty in Artificial Intelligence*, 2014.

Miroslav Dudík, Xintong Wang, David M. Pennock, and David M. Rothschild. Log-time prediction markets for interval securities, 2021. URL https://arxiv.org/abs/2102.07308.

Zhou Fan, Francisco Marmolejo-Cossío, Ben Altschuler, He Sun, Xintong Wang, and David C. Parkes. Differential liquidity provision in uniswap v3 and implications for contract design, 2022. URL https://arxiv.org/abs/2204.00464.

Zhou Fan, Francisco Marmolejo-Cossío, Daniel J. Moroz, Michael Neuder, Rithvik Rao, and David C. Parkes. Strategic liquidity provision in uniswap v3, 2023. URL https://arxiv.org/abs/2106.12033.

Rafael Frongillo, Maneesha Papireddygari, and Bo Waggoner. An axiomatic characterization of cfmms and equivalence to prediction markets, 2023. URL https://arxiv.org/abs/2302.00196.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Robin Hanson. Combinatorial Information Market Design. *Information Systems Frontiers*, 5(1):107–119, 2003.

Xiaolong Li and Jennifer Wortman Vaughan. An axiomatic characterization of adaptive-liquidity market makers. In *ACM EC*, 2013.

Manifold. Maniswap. https://www.notion.so/Maniswap-ce406e1e897d417cbd491071ea8a0c39, 2022.

Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. Complexity-approximation trade-offs in exchange mechanisms: Amms vs. lobs. In *International Conference on Financial Cryptography and Data Security*, pages 326–343. Springer, 2023.

Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. A Myersonian Framework for Optimal Liquidity Provision in Automated Market Makers. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference (ITCS 2024)*, volume 287 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 81:1–81:19, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-309-6. doi: 10.4230/LIPIcs.ITCS.2024.81. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2024.81.

Jason Milionis, Ciamac C. Moallemi, and Tim Roughgarden. *Automated Market Making and Arbitrage Profits in the Presence of Fees*, page 159–171. Springer Nature Switzerland, 2025. ISBN 9783031786761. doi: 10.1007/978-3-031-78676-1_9. URL http://dx.doi.org/10.1007/978-3-031-78676-1_9.

Vijay Mohan. Automated market makers and decentralized exchanges: a defi primer. *Financial Innovation*, 8(20), 2022. doi: 10.1186/s40854-021-00314-5. URL http://dx.doi.org/10.1145/3570639.

Long Nguyen. Minswap - multi-pool decentralized exchange on cardano. https://docs.minswap.org/get-started/whitepaperhttps://docs.minswap.org/get-started/whitepaper, 2021.

Abraham Othman and Tuomas Sandholm. Liquidity-Sensitive Automated Market Makers via Homogeneous Risk Measures, 2011.

Abraham Othman, David M. Pennock, Daniel M. Reeves, and Tuomas Sandholm. A practical liquidity-sensitive automated market maker. *ACM Transactions on Economics and Computation (TEAC)*, 1(3):1–25, 2013.

Evgeni Y. Ovcharov. Existence and uniqueness of proper scoring rules. *J. Mach. Learn. Res.*, 16:2207–2230, 2015. ISSN 1532-4435,1533-7928.

Evgeni Y. Ovcharov. Proper scoring rules and Bregman divergence. *Bernoulli*, 24(1):53 – 79, 2018. doi: 10.3150/16-BEJ857. URL https://doi.org/10.3150/16-BEJ857.

Geoffrey Ramseyer, Mohak Goyal, Ashish Goel, and David Mazières. Augmenting batch exchanges with constant function market makers. In *Proceedings of the 25th ACM Conference on Economics and Computation*, EC '24, page 986–1016, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400707049. doi: 10.1145/3670865.3673569. URL https://doi.org/10.1145/3670865.3673569.

R.T. Rockafellar. *Convex analysis*, volume 28 of *Princeton Mathematics Series*. Princeton University Press, 1997.

Jan Christoph Schlegel, Mateusz Kwaśnicki, and Akaki Mamageishvili. Axioms for constant function market makers, 2022. URL https://arxiv.org/abs/2210.00048.

Thomas Strömberg. *A study of the operation of infimal convolution.* PhD thesis, Lule tekniska universitet, 1994.

Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Computing Surveys*, 55(11):1–50, February 2023. ISSN 1557-7341. doi: 10.1145/3570639. URL http://dx.doi.org/10.1145/3570639.

Ariel Zetlin-Jones, Bryan Routledge, and Yikang Shen. Automated exchange economies. 2024. URL https://www.andrew.cmu.edu/user/azj/files/rsz_slides.pdf.

A PROOFS FROM SECTION 4

We prove in Appendix A.3 that Protocol 2 and Protocol 1 are essentially equivalent under some mild conditions.

A.1 Technical lemmas

We begin with some standard facts from convex analysis.

Proposition 3 (Rockafellar [1997, Theorem 23.5]). *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be a closed convex function and f^* its conjugate. Then for all $\mathbf{x}, \mathbf{x}^* \in \mathbb{R}^n$ the following are equivalent:*

(1) $\mathbf{x}^* \in \partial f(\mathbf{x})$
(2) $\mathbf{x} \in \partial f^*(\mathbf{x}^*)$
(3) $f(\mathbf{x}) + f^*(\mathbf{x}^*) = \langle \mathbf{x}^*, \mathbf{x} \rangle$

Proposition 4. *Let $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be convex for $i \in \{1, \dots, k\}$ such that $\bigcap_i \operatorname{relint} \operatorname{dom} f_i \neq \emptyset$. Let $f = \sum_i f_i$. Then $f^* = \bigwedge_i f_i^*$, and the infimum in $\bigwedge$ in the definition of f^* is always attained. Moreover, for $\mathbf{v} \in \operatorname{relint} \operatorname{dom} f^*$, and any split $\mathbf{v} = \sum_i \mathbf{v}^i$, we have $f^*(\mathbf{v}) = \sum_i f_i^*(\mathbf{v}^i)$ if and only if $\bigcap_i \partial f_i(\mathbf{v}^i) \neq \emptyset$.*

Proof. The first statement follows from Rockafellar [1997, Theorem 16.4]; we will prove the second. First suppose $f^*(\mathbf{v}) = \sum_i f_i^*(\mathbf{v}^i)$ for $\mathbf{v} \in \operatorname{relint} \operatorname{dom} f^*$ and $\mathbf{v} = \sum_i \mathbf{v}^i$. From Rockafellar [1997, Theorem 23.4], $\partial f^*(\mathbf{v}) \neq \emptyset$. Now Strömberg [1994, Theorem 3.6] gives $\bigcap_i \partial f_i^*(\mathbf{v}^i) = \partial f^*(\mathbf{v}) \neq \emptyset$. [8]

For the converse, let $\mathbf{x} \in \bigcap_i \partial f_i^*(\mathbf{v}^i)$ and define $\mathbf{v} = \sum_i \mathbf{v}^i$. From Proposition 3, $\mathbf{v}^i \in \partial f_i(\mathbf{x})$ for all i. Now Rockafellar [1997, Theorem 23.8] gives $\mathbf{v} = \sum_i \mathbf{v}^i \in \partial f(\mathbf{x})$. Proposition 3 again implies gives $f^*(\mathbf{v}) = \langle \mathbf{x}, \mathbf{v} \rangle - f(\mathbf{x}) = \sum_i \left\langle \mathbf{x}, \mathbf{v}^i \right\rangle - \sum_i f_i(\mathbf{x}) = \sum_i f_i^*(\mathbf{v}^i)$. □

We now prove several technical lemmas we use in the proof of Theorem 1.

Lemma 3. *Let G be a pseudobarrier and $C = G^*$. Then for all $\mathbf{q}, \hat{\mathbf{q}} \in \mathbb{R}^n$ we have $\partial C(\mathbf{q}) \subseteq \operatorname{relint} \Delta_n$ and $\hat{\mathbf{q}} \not\geq \mathbf{q} \implies C(\hat{\mathbf{q}}) > C(\mathbf{q})$.*

Proof. Let $\mathbf{p} \in \partial C(\mathbf{q})$. By Proposition 3, $\mathbf{q} \in \partial G(\mathbf{p})$. If $\mathbf{p} \notin \operatorname{relint} \Delta_n$, then we have an interior sequence $\{\mathbf{p}^j\}_j$ such that $\mathbf{p}^j \to_j \mathbf{p}$ and subgradients $\mathbf{q}^j \to_j \mathbf{q}$, violating the definition of a pseudobarrier. For this $\mathbf{p}$, we have $p_y > 0$ for all $y \in \{1, \dots, n\}$. As $\hat{\mathbf{q}} \not\geq \mathbf{q}$, we have $\hat{q}_y \geq q_y$ for all y, with at least one inequality strict. Thus by the subgradient inequality, $C(\hat{\mathbf{q}}) - C(\mathbf{q}) \geq \langle \mathbf{p}, \hat{\mathbf{q}} - \mathbf{q} \rangle > 0$, as desired. □

The following lemma is essentially a restatement of results due to Ovcharov [2015, 2018]. It says that subgradients of generating functions G are unique modulo $\mathbf{1}$.

Lemma 4. *Let G be a generating function. Then for all $p \in \Delta_n$, and all $\mathbf{q}, \hat{\mathbf{q}} \in \partial G(\mathbf{p})$, there exists $\alpha \in \mathbb{R}$ such that $\hat{\mathbf{q}} = \mathbf{q} + \alpha \mathbf{1}$.*

[8] As needed for that result to apply, the assumption that there exists some $\mathbf{x} \in \bigcap_i \operatorname{relint} \operatorname{dom} f_i$ implies that f_i^* is bounded from below by the same affine function, namely one with gradient $\mathbf{x}$.

Proof. Let $\mathbf{q} \in \partial G(\mathbf{p})$. We will show $\hat{\mathbf{q}} \in \partial \overline{G}(\mathbf{p})$, where $\hat{\mathbf{q}} = \mathbf{q} + (G(\mathbf{p}) - \langle \mathbf{q}, \mathbf{p} \rangle)\mathbf{1}$. By construction, $\langle \hat{\mathbf{q}}, \mathbf{p} \rangle = \langle \mathbf{q}, \mathbf{p} \rangle + G(\mathbf{p}) - \langle \mathbf{q}, \mathbf{p} \rangle = G(\mathbf{p})$. For all $\mathbf{x} \in \mathbb{R}^n_{\geq 0} \setminus \{\mathbf{0}\}$, we have

$$\begin{aligned}
\overline{G}(\mathbf{x}) &= \|\mathbf{x}\|_1 G(\mathbf{x}/\|\mathbf{x}\|_1) \\
&\geq \|\mathbf{x}\|_1 \left(G(\mathbf{p}) + \langle \mathbf{q}, \mathbf{x}/\|\mathbf{x}\|_1 - \mathbf{p} \rangle\right) \\
&= \|\mathbf{x}\|_1 \left(G(\mathbf{p}) + \langle \hat{\mathbf{q}}, \mathbf{x}/\|\mathbf{x}\|_1 - \mathbf{p} \rangle\right) \\
&= \|\mathbf{x}\|_1 \left(\langle \hat{\mathbf{q}}, \mathbf{x}/\|\mathbf{x}\|_1 \rangle + G(\mathbf{p}) - \langle \hat{\mathbf{q}}, \mathbf{p} \rangle\right) \\
&= \langle \hat{\mathbf{q}}, \mathbf{x} \rangle \\
&= \overline{G}(\mathbf{p}) - \langle \hat{\mathbf{q}}, \mathbf{p} \rangle + \langle \hat{\mathbf{q}}, \mathbf{x} \rangle \\
&= \overline{G}(\mathbf{p}) + \langle \hat{\mathbf{q}}, \mathbf{x} - \mathbf{p} \rangle \ .
\end{aligned}$$

Thus, every element of $\partial G(\mathbf{p})$, up to a shift by $\mathbf{1}$, is an element of $\partial \overline{G}(\mathbf{p})$. As the latter is a singleton set, by assumption on G, the result follows. □

Lemma 5. *Let $C = \bigwedge_i C_i$ where C_i^* are generating functions.*
Then for all $\mathbf{q} \in \mathbb{R}^n$, the infimum in the definition of $C(\mathbf{q})$ is attained, and $C(\mathbf{q}) = \sum_i C_i(\mathbf{q}^i)$ if and only there exists $\mathbf{p} \in \Delta_n$ such that $p \in \partial C_i(\mathbf{q}^i)$ for all i.

Proof. We have $\operatorname{dom} C_i^* = \Delta_n$ and $\operatorname{dom} C_i = \mathbb{R}^n$ for all i, so Proposition 4 applies.

□

Lemma 6. *Let $C = G^*$ where G is a generating function. For $\mathbf{p} \in \operatorname{relint} \Delta_n$, let $S(\mathbf{p}, y) = G(\mathbf{p}) - \left\langle \mathbf{d}G_{\mathbf{p}}, \boldsymbol{\delta}^y - \mathbf{p} \right\rangle$, where $\{\mathbf{d}G_{\mathbf{p}} \in \partial G(\mathbf{p}) \mid \mathbf{p} \in \operatorname{relint} \Delta_n\}$ is a selection of subgradients.*
Then for all $\mathbf{p} \in \operatorname{relint} \Delta_n$ and $\mathbf{q} \in \mathbb{R}^n$, we have $(\mathbf{p} \in \partial C(\mathbf{q}) \wedge C(\mathbf{q}) = 0) \iff \mathbf{q} = S(\mathbf{p}, \cdot)$. In particular, $\{S(\mathbf{p}, \cdot) \mid \mathbf{p} \in \operatorname{relint} \Delta_n\} = \{\mathbf{q} \in C^{-1}(0) \mid \partial C(\mathbf{q}) \cap \operatorname{relint} \Delta_n \neq \emptyset\}$.

Proof. Let $\mathbf{q} = S(\mathbf{p}, \cdot)$. Then $C(\mathbf{q}) = C(\mathbf{d}G_{\mathbf{p}} + (G(\mathbf{p}) - \left\langle \mathbf{d}G_{\mathbf{p}}, \mathbf{p} \right\rangle)\mathbf{1}) = G(\mathbf{d}G_p) + G(\mathbf{p}) - \left\langle \mathbf{d}G_{\mathbf{p}}, \mathbf{p} \right\rangle = 0$ by Proposition 3. Furthermore, by the same theorem, $\mathbf{d}G_{\mathbf{p}} \in \partial G(\mathbf{p}) \iff \mathbf{p} \in \partial C(\mathbf{d}G_{\mathbf{p}})$, and by $\mathbf{1}$-invariance, $\partial C(\mathbf{d}G_{\mathbf{p}}) = \partial C(S(\mathbf{p}, \cdot))$. Thus, $\mathbf{p} \in \partial C(\mathbf{q})$.

Now let $\mathbf{q}$ such that $C(\mathbf{q}) = 0$, and take $\mathbf{p} \in \partial C(\mathbf{q})$; we will show $\mathbf{q} = S(\mathbf{p}, \cdot)$. By Proposition 3, $\mathbf{q} \in \partial G(\mathbf{p})$. From Lemma 4, we have $\partial G(\mathbf{p}) = \{\mathbf{d}G_{\mathbf{p}} + \alpha\mathbf{1} \mid \alpha \in \mathbb{R}\}$. Thus $\mathbf{q} = \mathbf{d}G_{\mathbf{p}} + \alpha\mathbf{1}$ for some $\alpha \in \mathbb{R}$. Now $0 = C(\mathbf{q}) = C(\mathbf{d}G_{\mathbf{p}} + \alpha\mathbf{1}) = C(\mathbf{d}G_{\mathbf{p}}) + \alpha = \left\langle \mathbf{d}G_{\mathbf{p}}, \mathbf{p} \right\rangle - G(\mathbf{p}) + \alpha$ by Proposition 3. Thus $\alpha = G(\mathbf{p}) - \left\langle \mathbf{d}G_{\mathbf{p}}, \mathbf{p} \right\rangle$, and we have $\mathbf{q} = \mathbf{d}G_{\mathbf{p}} + (G(\mathbf{p}) - \left\langle \mathbf{d}G_{\mathbf{p}}, \mathbf{p} \right\rangle)\mathbf{1} = S(\mathbf{p}, \cdot)$, as desired.

□

A.2 Proof of Theorem 1

Proof. We will show that 1 and 5 are each equivalent to 2, 3 is equivalent to 1, and 4 is equivalent to 1, and 3 is equivalent to 4. For each, let $\{\mathbf{q}^i\}_i$ be the current market state, $\mathbf{q} = \sum_i \mathbf{q}^i$, $\mathbf{p}$ a consistent price, and $C = \bigwedge_i C_i$. From Lemma 5, price consistency implies $C(\mathbf{q}) = \sum_i C_i(\mathbf{q}^i)$, i.e., the $\mathbf{q}^i$ vectors achieve the infimum in the definition of the infimal convolution.

(1 ↔ 2) A trade for 2 satisfies the conditions for 1, so we only need to show that this choice is Pareto optimal for the trader; coherence will then follow by Lemma 5. More formally, let $\{\mathbf{r}^i\}_i$ satisfy $C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{q}^i)$ for all i. Letting $\mathbf{r} = \sum_i \mathbf{r}^i$, from the definition of infimal convolution, $C(\mathbf{q} + \mathbf{r}) \leq \sum_i C_i(\mathbf{q}^i + \mathbf{r}^i) = \sum_i C_i(\mathbf{q}^i) = C(\mathbf{q})$. We wish to show that $\mathbf{r}$ is Pareto optimal if and only if $C(\mathbf{q}+\mathbf{r}) = C(\mathbf{q})$, or equivalently, that $\mathbf{r}$ is Pareto suboptimal if and only if $C(\mathbf{q}+\mathbf{r}) < C(\mathbf{q})$. Suppose first that we had some $\hat{\mathbf{r}} \not\geq \mathbf{r}$ such that $C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i) = C_i(\mathbf{q}^i)$ where $\hat{\mathbf{r}} = \sum_i \hat{\mathbf{r}}^i$. From the same argument as above, we have $C(\mathbf{q} + \hat{\mathbf{r}}) \leq \sum_i C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i) = \sum_i C_i(\mathbf{q}^i) = C(\mathbf{q})$. By Lemma 3, $C(\mathbf{q} + \hat{\mathbf{r}}) > C(\mathbf{q} + \mathbf{r})$, giving $C(\mathbf{q} + \mathbf{r}) < C(\mathbf{q})$. Conversely, suppose $C(\mathbf{q} + \mathbf{r}) \neq C(\mathbf{q})$, which from the inequality above implies $C(\mathbf{q} + \mathbf{r}) < C(\mathbf{q})$. Let $\hat{\mathbf{r}} = \mathbf{r} + (C(\mathbf{q}) - C(\mathbf{q} + \mathbf{r}))\mathbf{1} \not\geq \mathbf{r}$. By

1 invariance, we have $C(\mathbf{q}+\hat{\mathbf{r}}) = C(\mathbf{q})$. From part (3) of the proof below, there exists a split $\hat{\mathbf{r}} = \sum_i \hat{\mathbf{r}}^i$ such that of $C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i) = C_i(\mathbf{q}^i)$. Thus, $\mathbf{r}$ was not a Pareto-optimal total trade.

(2 $\leftrightarrow$ 5) Let $\mathbf{r}$ be a trade from interpretation 2, so that $C(\mathbf{q}+\mathbf{r}) = C(\mathbf{q})$ where $C = \bigwedge_i C_i$. Set $\mathbf{v} = \mathbf{r}$ and $\alpha = 1$ for interpretation 5. Let $\alpha_i \geq 0$ be the amount of $\mathbf{r}$ purchased from cost function i, and $\beta_i \in \mathbb{R}$ the total cost, so that the net trade from cost function i is $\mathbf{r}^i = \alpha_i \mathbf{r} - \beta_i \mathbf{1}$. Then we have $\sum_i \alpha_i = 1$, so that the net trade is $\mathbf{r} - \beta\mathbf{1}$ where $\beta = \sum_i \beta_i$. By definition of $\mathbf{r}^i$, the cost of trades, and the $\mathbf{1}$-invariance of C_i, we have $C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{q}^i)$.
If $\beta = 0$ we are done. Otherwise, as $\mathbf{r}$ is Pareto optimal from part (1) above, we must have $\beta > 0$. From part (3) of the proof below, there exists a set of trades $\{\hat{\mathbf{r}}^i\}_i$ with $\sum_i \hat{\mathbf{r}}^i = \mathbf{r}$ such that $C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i) = C_i(\mathbf{q}^i) = C_i(\mathbf{q}^i + \mathbf{r}^i - \beta_i\mathbf{1})$ for all i. Thus, from state $\{\hat{q}^i\}_i := \{\mathbf{q}^i + \mathbf{r}^i - \beta_i\mathbf{1}\}_i$, the trades $\{\hat{r}^i\}_i := \{\hat{\mathbf{r}}^i - \mathbf{r}^i\}_i$ are an arbitrage, with $\sum_i \hat{r}^i = \mathbf{r} - \mathbf{r} = \mathbf{0}$ and net cost

$$\begin{aligned}\sum_i C_i(\hat{q}^i + \hat{r}^i) - C_i(\hat{q}^i) &= \sum_i C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i - \beta_i\mathbf{1}) - C_i(\mathbf{q}^i + \mathbf{r}^i - \beta_i\mathbf{1}) \\ &= \sum_i C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i - \beta_i\mathbf{1}) - C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i) \\ &= -\sum_i \beta_i = -\beta < 0\,.\end{aligned}$$

For the converse, let $\mathbf{r}$ be any net trade from the continuous trading process, and $\beta \leq 0$ the optimal net cost from any arbitrage. By part (1) and the argument above, we have $\beta = C(\mathbf{q}) - C(\mathbf{q}+\mathbf{r})$; taking $\hat{\mathbf{r}} = \mathbf{r} + (C(\mathbf{q}) - C(\mathbf{q}+\mathbf{r}))\mathbf{1}$,
we again split $\hat{\mathbf{r}}$ into $\{\hat{\mathbf{r}}^i\}_i$ and construct an arbitrage $\{\mathbf{r}^{i\,\prime}\}_i := \{\hat{\mathbf{r}}^i - \mathbf{r}^i\}_i$ which achieves net cost $C(\mathbf{q}+\mathbf{r}) - C(\mathbf{q}) = -\beta$.

(3 $\leftrightarrow$ 1) Let $\mathbf{r}$ such that $C(\mathbf{q}+\mathbf{r}) = C(\mathbf{q})$. We must show that there exist $\{\mathbf{r}^i\}_i$ such that $C_i(\mathbf{q}^i+\mathbf{r}^i) = C_i(\mathbf{q}^i)$ and $\mathbf{r} = \sum_i \mathbf{r}^i$. From the definition of infimal convolution, $C(\mathbf{q}+\mathbf{r}) = \inf\{\sum_i C_i(\mathbf{v}^i) \mid \sum_i \mathbf{v}^i = \mathbf{q}+\mathbf{r}\}$. By Lemma 5, this infimum is attained by some $\{\mathbf{v}^i\}_i$. Define $\mathbf{r}^i := \mathbf{v}^i - \mathbf{q}^i + (C_i(\mathbf{q}^i) - C_i(\mathbf{v}^i))\mathbf{1}$. For the first condition, $C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{v}^i + (C_i(\mathbf{q}^i) - C_i(\mathbf{v}^i))\mathbf{1}) = C_i(\mathbf{q}^i)$. For the second,

$$\begin{aligned}\sum_i \mathbf{r}^i &= \sum_i \mathbf{v}^i - \sum_i \mathbf{q}^i + \sum_i (C_i(\mathbf{q}^i) - C_i(\mathbf{v}^i))\mathbf{1} \\ &= (\mathbf{q}+\mathbf{r}) - \mathbf{q} + \left(\sum_i C_i(\mathbf{q}^i) - \sum_i C_i(\mathbf{v}^i)\right)\mathbf{1} \\ &= \mathbf{r} + (C(\mathbf{q}) - C(\mathbf{q}+\mathbf{r}))\mathbf{1} = \mathbf{r}\,.\end{aligned}$$

Coherence again follows from Lemma 5.

(4 $\leftrightarrow$ 1) We will show equivalence to interpretation 1.
Let $\alpha_i = C_i(\mathbf{q}^i)$ for all i, so that $C(\mathbf{q}^i - \alpha\mathbf{1}) = 0$. By Lemma 6, we may therefore write $\mathbf{q}^i = S(\mathbf{p}^i, \cdot) + \alpha_i\mathbf{1}$. From Lemma 5, we again have $C(\mathbf{q}) = \sum_i C(\mathbf{q}^i)$. From Lemma 6 again, and $\mathbf{1}$-invariance,

$$\begin{aligned}&\{\mathbf{r}^i \in \mathbb{R}^n \mid C_i(\mathbf{q}^i + \mathbf{r}^i) = C_i(\mathbf{q}^i)\} \\ &= \{\mathbf{r}^i \in \mathbb{R}^n \mid C_i(\mathbf{q}^i + \mathbf{r}^i - \alpha_i\mathbf{1}) = C_i(\mathbf{q}^i - \alpha_i\mathbf{1})\} \\ &= \{\mathbf{r}^i \in \mathbb{R}^n \mid C_i(S_i(\mathbf{p}^i, \cdot) + \mathbf{r}^i) = C_i(S_i(\mathbf{p}^i, \cdot))\} \\ &= \{\mathbf{r}^i \in \mathbb{R}^n \mid C_i(S_i(\mathbf{p}^i, \cdot) + \mathbf{r}^i) = 0\} \\ &= \{\hat{\mathbf{q}}^i - S(\mathbf{p}^i, \cdot) \mid C_i(\hat{\mathbf{q}}^i) = 0\} \\ &= \{S(\hat{\mathbf{p}}^i, \cdot) - S(\mathbf{p}^i, \cdot) \mid \hat{\mathbf{p}}^i \in \operatorname{relint}\Delta_n\}\,.\end{aligned}$$

We conclude that the possible trades $\{\mathbf{r}^i\}_i$ in interpretation 1, such that $C_i(\mathbf{q}^i + \hat{\mathbf{r}}^i) = C_i(\mathbf{q}^i)$ for all i, are exactly the same as the trades $\{S_i(\hat{\mathbf{p}}^i, \cdot) - S_i(\mathbf{p}^i, \cdot)\}_i$ allowed in interpretation 4; we have simply reparameterized the trades by $\{\hat{\mathbf{p}}^i\}_i$.

(4 $\leftrightarrow$ 3) We will show equivalence to interpretation 3. In interpretation 3, suppose k market makers use scoring rules $S_{G_i}(\mathbf{p}, \cdot)$ for some corresponding set of generating functions $\{G_i\}_{i=1}^k$. Assuming a coherent market state, all market makers maintain the same initial price $\mathbf{p}$. The trader chooses a price $\hat{p}$, and receives the trade $\mathbf{r}^i = S_{G_i}(\hat{\mathbf{p}}, \cdot) - S_{G_i}(\mathbf{p}, \cdot)$ from market maker i. Therefore, for a given initial price $\mathbf{p}$ and final price $\hat{\mathbf{p}}$, the trader receives the trade

$$\mathbf{r} = \sum_{i=1}^{k} \mathbf{r}^i = \sum_{i=1}^{k} S_{G_i}(\hat{\mathbf{p}}, \cdot) - S_{G_i}(\mathbf{p}, \cdot)$$

Let $G = \sum_{i=1}^k G_i$, and let $S_G(\mathbf{p}, \cdot)$ be the corresponding scoring rule. It follows that $S_G(\mathbf{p}, \cdot) = \sum_{i=1}^k S_{G_i}(\mathbf{p}, \cdot)$. Now, considering interpretation 4, let the market maker pick generating function G and thus scoring rule S_G, and the same initial price $\mathbf{p}$ as above. A trader chooses a new price $\hat{p}$, and receives trade $\mathbf{r} = S_G(\hat{\mathbf{p}}, \cdot) - S_G(\mathbf{p}, \cdot)$, the same as in interpretation 3. □

A.3 Equivalence of the full protocols and practical considerations

Theorem 1 tells us that the process of trading in Protocols 1 and 2 are the same. The equivalence of the rest of the protocol follows from Lemma 6, as $\text{liability}(C) = S_G(\mathbf{p}, \cdot)$ where $\mathbf{p} \in \partial C(\mathbf{q})$.

The two-outcome case is similar; we need only verify the translation from G and C to thei 1-dimensional counterparts. Letting $G(\mathbf{p}) = g(p_1)$, we have $\overline{G}(\mathbf{x}) = (x_1 + x_2)g(x_1/(x_1 + x_2))$ which is differentiable. Lemma 4 now gives $\partial G(\mathbf{p}) = \{(g'(p_1), 0) + \alpha\mathbf{1} \mid \alpha \in \mathbb{R}\}$ and thus $S_G(\mathbf{p}, \cdot) = \mathbf{d}G_{\mathbf{p}} + (G(\mathbf{p}) + \langle \mathbf{d}G_{\mathbf{p}}, \mathbf{p} \rangle)\mathbf{1} = (g'(p_1), 0) + (g(p_1) - p_1 g'(p_1))\mathbf{1} = \text{liability}(g, p_1)$. Computing the conjugate, we have

$$\begin{aligned} G^*(\mathbf{q}) &= \sup_{\mathbf{p} \in \Delta_2} \langle \mathbf{p}, \mathbf{q} \rangle - G(\mathbf{p}) \\ &= \sup_{p \in [0,]1} pq_1 + (1-p)q_2 - g(p) \\ &= \left(\sup_{p \in [0,1]} p(q_1 - q_2) - g(p) \right) + q_2 \\ &= g^*(q_1 - q_2) + q_2 \,, \end{aligned}$$

as desired. Finally, to verify $\text{price}(\cdot)$, note that $c' = (g')^{-1}$ whenever both derivatives are defined. By assumption on $\mathcal{G}_{\text{init}}$, any argument to price is both differentiable and strictly convex, and thus c is differentiable. We have now established Proposition 5.

B PROOFS AND ADDITIONAL WORK FROM SECTION 6

B.1 Worked example for Uniswap fees

Consider a market with two LPs $G = G_1 + G_2$ where $G_1(\mathbf{p}) = -2\sqrt{p_1 p_2}$ and $G_2(\mathbf{p}) = -2\sqrt{p_2 p_3}$. Per Protocol 2, a trade $\mathbf{p} \to \hat{\mathbf{p}}$ is given by $\mathbf{r} = S_G(\hat{\mathbf{p}}, \cdot) - S_G(\mathbf{p}, \cdot)$, where

$$S_G(\mathbf{p}, \cdot) = \nabla \overline{G}(\mathbf{p}) = \nabla \overline{G}_1(\mathbf{p}) + \nabla \overline{G}_2(\mathbf{p}) = \left(\sqrt{p_1/p_2}, \sqrt{p_2/p_1} + \sqrt{p_2/p_3}, \sqrt{p_3/p_2} \right) .$$

The trade is then split among the two LPs, as

$$\begin{aligned} \mathbf{r}^1 &= \nabla G_1(\hat{\mathbf{p}}) - \nabla G_1(\mathbf{p}) = \left(\sqrt{\hat{p}_1/\hat{p}_2} - \sqrt{p_1/p_2}, \sqrt{\hat{p}_2/\hat{p}_1} - \sqrt{p_2/p_1}, 0 \right) , \\ \mathbf{r}^2 &= \nabla G_2(\hat{\mathbf{p}}) - \nabla G_2(\mathbf{p}) = \left(0, \sqrt{\hat{p}_2/\hat{p}_3} - \sqrt{p_2/p_3}, \sqrt{\hat{p}_3/\hat{p}_2} - \sqrt{p_3/p_2} \right) . \end{aligned}$$

Start the market at the uniform price $\mathbf{p} = (1/3, 1/3, 1/3)$, and consider a trader wishing to purchase security 1 in exchange for security 3, as above. Intuitively, as there is liquidity between securities 1 and 2 (provided by LP 1) and between securities 2 and 3 (provided by LP 2), there should be "combined" liquidity between 1 and 3. And indeed that is the case: if the trader selects $\hat{\mathbf{p}} = \left(\frac{3/2+\sqrt{2}}{3+\sqrt{2}}, \frac{1}{3+\sqrt{2}}, \frac{1}{2(3+\sqrt{2})}\right)$, an expression chosen for arithmetic convenience, we have a resulting trade $\mathbf{r} = \nabla\overline{G}(\hat{\mathbf{p}}) - \nabla\overline{G}(\mathbf{p}) = (\sqrt{3/2} - 1, 0, \sqrt{1/2} - 1)$. The split $\mathbf{r} = \mathbf{r}^1 + \mathbf{r}^2$ between the LPs is also roughly as one would expect, each $\mathbf{r}^i$ being between the corresponding pair of securities:

$$\mathbf{r}^1 = \nabla G_1(\hat{\mathbf{p}}) - \nabla G_1(\mathbf{p}) = \left(0, \sqrt{2} - 1, \sqrt{1/2} - 1\right),$$
$$\mathbf{r}^2 = \nabla G_2(\hat{\mathbf{p}}) - \nabla G_2(\mathbf{p}) = \left(\sqrt{3/2} - 1, 1 - \sqrt{2}, 0\right).$$

Using Uniswap's fee, i.e., $\text{fee}(\mathbf{r}, \mathbf{q}) = \beta(-\mathbf{r})_+$, $\text{fee}_i(\mathbf{r}, \mathbf{q}) = \beta(-\mathbf{r}^i)_+$, presents an issue. The fee charged to the trader, $\beta(-\mathbf{r})_+ = \beta(0, 0, 1 - \sqrt{1/2})$, ignores the fact that LP 2 provided liquidity that facilitated the trade! Indeed, looking at the fees paid to LPs, we see this same fee $\beta(-\mathbf{r}^1)_+ = \beta(0, 0, 1 - \sqrt{1/2})$ paid to LP 1, plus an additional fee of $\beta(-\mathbf{r}^2)_+ = \beta(0, \sqrt{2} - 1, 0)$ to LP 2.

Fortunately, this issue is not present in 2-asset protocols like Protocol 3, but clearly it can emerge beyond 2 assets/outcomes. Moreover, it seems to emerge precisely when there is "synergy" among the LPs, enabling trades that fruitfully combine their liquidity. While one could easily fix this issue by directly charging the trader for the sum of the fees to the LPs, doing so may be problematic. For example, this proposal would break the abstraction barrier, in the sense that the fees would depend intimately on the LP profile, not just their combined liquidity.

B.2 Proof of Lemma 1

Lemma 1. *Axioms 1, 2, and 3 allow us to write*

$$\overline{\text{fee}}_{\text{T}} = \text{fee}_{\text{LP}}.$$

Proof. Assume without loss of generality, by Theorem 1, that the market maker has only one LP providing all the liquidity. Hence $\mathbf{r} = r^1$ and $\mathbf{q} = q^1$. Then, Axioms 1, 3 give us that

$$\begin{aligned}\text{fee}_1(\{\mathbf{r}^1\}, \{\mathbf{q}^1\}) &= \text{fee}_{\text{T}}(\{\mathbf{r}^1\}, \{\mathbf{q}^1\}) \\ &= \text{fee}_{\text{LP}}(\mathbf{r}^1, \mathbf{q}^1)\end{aligned}$$

Moreover, Axiom 2 says

$$\text{fee}_{\text{T}}(\{\mathbf{r}^1\}, \{\mathbf{q}^1\}) = \overline{\text{fee}}_{\text{T}}(\mathbf{r}^1, \mathbf{q}^1)$$

Hence $\overline{\text{fee}}_{\text{T}}() = \text{fee}_{\text{LP}}()$. □

B.3 Proof of Theorem 2

Theorem 2. *Axioms 1, 2, 3, and 4 are incompatible.*

Proof. We will consider the scoring rule interpretation of liquidity provisioning and parallel market making, according to Protocol 3. Consider $n = 3$ assets, and 5 LPs, which provide liquidity

according to the following 5 generating functions:

$$\begin{aligned}
G_1 &= -2\sqrt{2p_2p_3},\\
G_2 &= -2\sqrt{2p_1p_3},\\
G_3 &= -2\sqrt{2p_1p_2},\\
G_4 &= 49 \cdot \frac{p_3^2 - p_3}{21p_3 + 4},\\
G_5 &= 9(p_3^2 - p_3).
\end{aligned}$$

Note that G_1, G_2, and G_3 are 1-homogeneous generating functions, each symmetric with respect to two assets and flat with respect to the third, and equal to each other up to permutation of the assets. G_4 and G_5 are designed to provide specific liquidity vectors in certain scenarios while also not facilitating any part of a particular trade. Now consider the following market. Set an initial price vector of $\mathbf{p} = (1/7, 4/7, 2/7)$. Let LPs LP_1, LP_2, and LP_4 enter the market with generating functions G_1, G_2, and G_4, respectively. The individual liabilities $\mathbf{q}^i$ for each LP i according to $S_{G_i}(\mathbf{p}, \cdot)$ are

$$\begin{aligned}
\mathbf{q}^1 &= (0, -1, -2),\\
\mathbf{q}^2 &= (-2, 0, -1),\\
\mathbf{q}^4 &= (-1, -1, -1).
\end{aligned}$$

And the total liability is

$$\mathbf{q} = \mathbf{q}^1 + \mathbf{q}^2 + \mathbf{q}^4 = (-3, -2, -4)$$

according to $S_{G_1+G_2+G_4}(\mathbf{p}, \cdot)$. Now, we trade to the price $\mathbf{p}_{\text{post}} = (4/7, 1/7, 2/7)$. At price $\mathbf{p}_{\text{post}}$, the liabilities according to the protocol are

$$\begin{aligned}
\mathbf{q}^1_{\text{post}} &= (0, -2, -1),\\
\mathbf{q}^2_{\text{post}} &= (-1, 0, -2),\\
\mathbf{q}^4_{\text{post}} &= (-1, -1, -1),\\
\mathbf{q}_{\text{post}} &= (-1, -2, -3).
\end{aligned}$$

Therefore, the net trades facilitated by each LP are

$$\begin{aligned}
\mathbf{r}^1 &= (0, -1, 1),\\
\mathbf{r}^2 &= (1, 0, -1),\\
\mathbf{r}^4 &= (0, 0, 0).
\end{aligned}$$

And the total net trade is

$$\mathbf{r} = \mathbf{r}^1 + \mathbf{r}^2 + \mathbf{r}^4 = (1, -1, 0).$$

Then, the fees assessed are

$$\begin{aligned}
&\mathrm{fee}(\mathbf{r}, \mathbf{q})\\
=&\mathrm{fee}((1, -1, 0), (-3, -2, -4))\\
=&\mathrm{fee}_1(\mathbf{r}^1, \mathbf{q}^1) + \mathrm{fee}_2(\mathbf{r}^2, \mathbf{q}^2) + \mathrm{fee}_4(\mathbf{0}, \mathbf{q}^4)\\
=&\mathrm{fee}_1((0, -1, 1), (0, -1, -2)) + \mathrm{fee}_2((1, 0, -1), (-2, 0, -1))
\end{aligned} \tag{1}$$

Note the application of Axiom 4, which states that the fee for the trade of $\mathbf{r}^4 = \mathbf{0}$ must be 0.

Now, consider the following alternative scenario. We set an initial price of $\mathbf{p} = (2/9, 4/9, 1/3)$. LPs LP_3 and LP_5 enter the market with generating functions G_3 and G_5, respectively. The liabilities deposited are

$$\begin{aligned}\mathbf{q}^3 &= (-2, -1, 0),\\ \mathbf{q}^5 &= (-1, -1, -4),\\ \mathbf{q} &= (-3, -2, -4).\end{aligned}$$

Now, we trade to the price $\mathbf{p}_{\text{post}} = (4/9, 2/9, 1/3)$. At price $\mathbf{p}_{\text{post}}$, the liabilities according to the protocol are

$$\begin{aligned}\mathbf{q}^3_{\text{post}} &= (-1, -2, 0),\\ \mathbf{q}^5_{\text{post}} &= (-1, -1, -4),\\ \mathbf{q}_{\text{post}} &= (-2, -3, -4).\end{aligned}$$

And the net trades are therefore

$$\begin{aligned}\mathbf{r}^3 &= (1, -1, 0),\\ \mathbf{r}^5 &= (0, 0, 0),\\ \mathbf{r} &= (1, -1, 0).\end{aligned}$$

The fee assessed is

$$\begin{aligned}&\text{fee}(\mathbf{r}, \mathbf{q})\\ =&\text{fee}((1, -1, 0), (-3, -2, -4))\\ =&\text{fee}_3(\mathbf{r}^3, \mathbf{q}^3) + \text{fee}_5(\mathbf{r}^5, \mathbf{q}^5)\\ =&\text{fee}_3((1, -1, 0), (-1, -2, 0)) + \text{fee}_5(\mathbf{0}, (-1, -1, -4))\\ =&\text{fee}_3((1, -1, 0), (-1, -2, 0))\end{aligned} \tag{2}$$

Note that the fee for $\mathbf{r}^5 = \mathbf{0}$ is 0, per Axiom 4. Combining eqs. (1) and (2), we have that

$$\text{fee}_1(\mathbf{r}^1, \mathbf{q}^1) + \text{fee}_2(\mathbf{r}^2, \mathbf{q}^2) = \text{fee}_3(\mathbf{r}^3, \mathbf{q}^3) \tag{3}$$

Now, we consider a scenario that is identical up to permutation of assets, with assets 2 and 3 being swapped. We introduce two more generating functions G_6 and G_7, which are identical to G_4 and G_5 but parameterized by p_2 rather than p_3.

$$\begin{aligned}G_4 &= 49 \cdot \frac{p_2^2 - p_2}{21p_2 + 4},\\ G_5 &= 9(p_2^2 - p_2).\end{aligned}$$

Let LPs LP_1, LP_3, and LP_6 enter the market with generating functions G_1, G_3, and G_6 respectively. We set an initial price vector of $\overline{\mathbf{p}} = (1/7, 2/7, 4/7)$, and trade to a price vector of $\overline{\mathbf{p}}_{\text{post}} = (4/7, 2/7, 1/7)$.

The initial liabilities $\overline{\mathbf{q}}^i$, trades $\overline{\mathbf{r}}^i$, and final liabilities $\overline{\mathbf{q}}^i_{\text{post}}$ are as follows:

$$\begin{aligned}
\overline{\mathbf{q}}^1 &= (0,-2,-1),\\
\overline{\mathbf{q}}^3 &= (-2,-1,0),\\
\overline{\mathbf{q}}^6 &= (-1,-1,-1),\\
\overline{\mathbf{q}} &= (-3,-4,-2),
\end{aligned}$$

$$\begin{aligned}
\overline{\mathbf{r}}^1 &= (0,1,-1),\\
\overline{\mathbf{r}}^3 &= (1,-1,0),\\
\overline{\mathbf{r}}^6 &= (0,0,0),\\
\overline{\mathbf{r}} &= (1,0,-1),
\end{aligned}$$

$$\begin{aligned}
\overline{\mathbf{q}}^1_{\text{post}} &= (0,-1,-2),\\
\overline{\mathbf{q}}^3_{\text{post}} &= (-1,-2,0),\\
\overline{\mathbf{q}}^6_{\text{post}} &= (-1,-1,-1),\\
\overline{\mathbf{q}}_{\text{post}} &= (-2,-4,-3).
\end{aligned}$$

The fees assessed are

$$\begin{aligned}
&\text{fee}(\overline{\mathbf{r}},\overline{\mathbf{q}})\\
=&\text{fee}((1,0,-1),(-3,-4,-2))\\
=&\text{fee}_1(\overline{\mathbf{r}}^1,\overline{\mathbf{q}}^1)+\text{fee}_3(\overline{\mathbf{r}}^3,\overline{\mathbf{q}}^3)+\text{fee}_6(\mathbf{0},\overline{\mathbf{q}}^6)\\
=&\text{fee}_1((0,1,-1),(0,-2,-1))+\text{fee}_3((1,-1,0),(-2,-1,0))
\end{aligned} \tag{4}$$

Now, instead consider a market with LPs LP_2 and LP_7, with generating functions G_2 and G_7 respectively, an initial price of $\overline{\mathbf{p}} = (2/9, 1/3, 4/9)$, and a final price of $\overline{\mathbf{p}}_{\text{post}} = (4/9, 2/9, 1/3)$. The corresponding liabilities and trades are

$$\begin{aligned}
\overline{\mathbf{q}}^2 &= (-2,0,-1),\\
\overline{\mathbf{q}}^7 &= (-1,-4,-1),\\
\overline{\mathbf{q}} &= (-3,-4,-2),
\end{aligned}$$

$$\begin{aligned}
\overline{\mathbf{r}}^2 &= (1,0,-1),\\
\overline{\mathbf{r}}^7 &= (0,0,0),\\
\overline{\mathbf{r}} &= (1,0,-1),
\end{aligned}$$

$$\begin{aligned}
\overline{\mathbf{q}}^2_{\text{post}} &= (-1,0,-2),\\
\overline{\mathbf{q}}^7_{\text{post}} &= (-1,-4,-1),\\
\overline{\mathbf{q}}_{\text{post}} &= (-2,-4,-3).
\end{aligned}$$

The fees assessed are

$$\begin{aligned}
&\mathsf{fee}(\overline{\mathbf{r}}, \overline{\mathbf{q}}) \\
=&\mathsf{fee}((1,0,-1),(-3,-4,-2)) \\
=&\mathsf{fee}_2(\overline{\mathbf{r}}^2, \overline{\mathbf{q}}^2) + \mathsf{fee}_7(\overline{\mathbf{r}}^7, \overline{\mathbf{q}}^7) \\
=&\mathsf{fee}_2((1,0,-1),(-1,0,-2)) + \mathsf{fee}_7(\mathbf{0},(-1,-4,-1)) \\
=&\mathsf{fee}_2((1,0,-1),(-1,0,-2))
\end{aligned} \tag{5}$$

Combining eqs. (4) and (5), we have that

$$\mathsf{fee}_1(\overline{\mathbf{r}}^1, \overline{\mathbf{q}}^1) + \mathsf{fee}_3(\overline{\mathbf{r}}^3, \overline{\mathbf{q}}^3) = \mathsf{fee}_2(\overline{\mathbf{r}}^2, \overline{\mathbf{q}}^2) \tag{6}$$

From eqs. (3) and (6), we have that

$$\begin{aligned}
\mathsf{fee}_1(\mathbf{r}^1, \mathbf{q}^1) + \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}^2) &= \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}^3), \\
\mathsf{fee}_1(\overline{\mathbf{r}}^1, \overline{\mathbf{q}}^1) + \mathsf{fee}_3(\overline{\mathbf{r}}^3, \overline{\mathbf{q}}^3) &= \mathsf{fee}_2(\overline{\mathbf{r}}^2, \overline{\mathbf{q}}^2).
\end{aligned}$$

Observe that $\overline{\mathbf{r}}^2 = \mathbf{r}^2$, $\overline{\mathbf{r}}^3 = \mathbf{r}^3$, $\overline{\mathbf{q}}^2 = \mathbf{q}^2$, and $\overline{\mathbf{q}}^3 = \mathbf{q}^3$, giving us

$$\begin{aligned}
\mathsf{fee}_1(\mathbf{r}^1, \mathbf{q}^1) + \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}^2) &= \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}^3), \\
\mathsf{fee}_1(\overline{\mathbf{r}}^1, \overline{\mathbf{q}}^1) + \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}^3) &= \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}^2).
\end{aligned}$$

Because $\mathbf{r}^1 \neq \mathbf{0}$, by Axiom 4, $\mathsf{fee}_1(\mathbf{r}^1, \mathbf{q}^1) > 0$, and therefore $\mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}^2) < \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}^3)$. Similarly, because $\overline{\mathbf{r}}^1 \neq \mathbf{0}$, $\mathsf{fee}_1(\overline{\mathbf{r}}^1, \overline{\mathbf{q}}^1) > 0$, and therefore $\mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}^3) < \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}^2)$. But then $\mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}^2) < \mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}^3)$ and $\mathsf{fee}_3(\mathbf{r}^3, \mathbf{q}^3) < \mathsf{fee}_2(\mathbf{r}^2, \mathbf{q}^2)$, a contradiction. □

C PROOFS AND ADDITIONAL WORK FROM SECTION 7

C.1 General protocol for the exchange of two securities

Before we analyze Uniswap, we introduce a simpler version of Protocol 1 for the two outcome case. This is our Protocol 3. This helps us reason about liquidity functions more conveniently as they apply to decentralized finance, as only two securities are exchanged. Recall that we use lowercase g and c for the two outcome case instead of G and C; see § 5.1. We detail the specifications in the next paragraph.

Since $C(S_g(p,\cdot)) = 0$ and $\nabla C(S_g(p,\cdot)) = (p, 1-p)$, we can replace $\mathsf{liability}(C)$ on Line 3 of Protocol 1 with $\mathsf{liability}(g,p) = S_g(p,\cdot)$ where p is the current price of asset 1. Similarly, we can more easil compute the trade check on Line 14, and optimal split on Line 17 in Protocol 1 using $S_g(p,\cdot)$ without needing to compute infimal convolutions explicitly.

The formal proof of equivalence of Protocol 1 and Protocol 3 follows from the equivalence between the corresponding n-asset protocols (Theorem 1). Let $\mathcal{G}$ be the set of functions $g : [0,1] \to \mathbb{R}_{\leq 0}$ which are convex, continuously differentiable, and bounded. Let $\mathcal{G}^* \subseteq \mathcal{G}$ be those which additionally have $|g'(p)| \to \infty$ as $p \to 0$ or $p \to 1$.

Proposition 5. *Let $\mathcal{G}_{\text{init}} \subseteq \mathcal{G}^*$ be a set of functions which are strictly convex. Then for $n = 2$ Protocol 1 is equivalent to Protocol 3 for the choices $c_i = g_i^*$ where $C_i(\mathbf{q}) = c_i(q_1 - q_2) + q_2$.*

C.2 Proofs related to Uniswap V2 in Section 7

In this section, we show that Uniswap V2, outlined in Protocol 4, is a special case of Protocol 3.

Proposition 6. *For $\mathcal{G}_{\text{init}} = \{\alpha g_0 \mid \alpha > 0\}$, $\mathcal{G}_{\text{LP}} = \{\alpha g_0 \mid \alpha \geq 0\}$ where $g_0(p) = -2\sqrt{p(1-p)}$ in Protocol 3,* $\mathsf{liability}(g,p) = -\alpha\left(\sqrt{\frac{1-p}{p}}, \sqrt{\frac{p}{1-p}}\right)^\top$ *for $g = \alpha g_0$. The vector $\mathbf{x}$ of reserves in Protocol 4 satisfies $x_1 \cdot x_2 = \alpha^2$ for some $\alpha > 0$ iff $\mathbf{q} = \mathsf{liability}(g, \mathsf{price}(g,p))$, where $\mathbf{x} = -\mathbf{q}$.*

Protocol 3 General two-asset protocol via liquidity functions

global constant $\mathcal{G}_{\text{init}} \subseteq \mathcal{G}^*$, $\mathcal{G}_{\text{LP}} \subseteq \mathcal{G}$, $\text{fee}(), \text{fee}_i() \in \mathbb{R}_{\geq 0}$
global variables $k \in \mathbb{N}$, $\{\mathbf{q}^i \in \mathbb{R}^2\}_{i=0}^k$, $\{g_i \in \mathcal{G}_{\text{LP}}\}_{i=0}^k$
$\text{price}(g, \mathbf{q}) := (g')^{-1}(q_1 - q_2)$
$\text{liability}(g, p) := S_g(p, \cdot)$ where $S_g(p, \cdot) = (g'(p), 0) + (g(p) - p \cdot g'(p))\mathbf{1}$.
function Initialize($\mathbf{q} \in \mathbb{R}^2, g \in \mathcal{G}_{\text{init}}$) ▷ Equivalently, the market creator can specify $\ell = g''$
    $(k, \mathbf{q}^0, g_0) \leftarrow (0, \mathbf{q}, g)$
    check $\mathbf{q}^0 = \text{liability}(g_0, \text{price}(g_0, \mathbf{q}))$
function RegisterLP($i = k + 1$)
    $(k, \mathbf{q}^i, g_i) \leftarrow (k + 1, 0, 0)$
function ModifyLiquidity($i \in \mathbb{N}, g \in \mathcal{G}_{\text{LP}}$) ▷ Equivalently, the LP can specify $\ell = g''$
    request $\mathbf{r}^i = \mathbf{q}^i - \text{liability}(g, \text{price}(g_0, \mathbf{q}^0))$ from LP i
    $(\mathbf{q}^i, g_i) \leftarrow (\mathbf{q}^i - \mathbf{r}^i, g)$
function ExecuteTrade($\mathbf{r} \in \mathbb{R}^2$)
    $\hat{p} \leftarrow \text{price}(\sum_{j=0}^k g_j, \sum_{i=0}^k \mathbf{q}^i + \mathbf{r})$ ▷ The price after this trade
    check $\sum_{i=0}^k \mathbf{q}^i + \mathbf{r} = \text{liability}(\sum_{j=0}^k g_j, \hat{p})$
    trader pays $\text{fee}(\mathbf{r}, \mathbf{q}, g)$ cash in fee
    Give r to trader
    for each LP i **do**
        $\mathbf{r}^i \leftarrow \text{liability}(g_i, \hat{p}) - \mathbf{q}^i$.
        LP i gets $\text{fee}_i(\mathbf{r}, \mathbf{q}, \{g_i\}_{i=1}^k)$ fees
        $\mathbf{q}^i \leftarrow \mathbf{q}^i + \mathbf{r}^i$

Proof. Observe that any change in liability vector in the ModifyLiquidity or Initialize phases results in a liability vector that takes the form $\text{liability}(g, p)$ for some $g \in \mathcal{G}_{\text{LP}}$, $p \in [0, 1]$. Let this $g(p) = \alpha g_0(p) = -2\alpha\sqrt{p(1-p)}$ and thereby $g'(p) = -\alpha \frac{1-2p}{\sqrt{p(1-p)}}$. So,

$$\begin{aligned}\text{liability}(g, p) &= (g(p) - p \cdot g'(p)\mathbf{1} + \begin{pmatrix} g'(p) \\ 0 \end{pmatrix} \\ &= \alpha \left(\left(-2\sqrt{p(1-p)} + p\frac{1-2p}{\sqrt{p(1-p)}} \right) \mathbf{1} - \begin{pmatrix} \frac{1-2p}{\sqrt{p(1-p)}} \\ 0 \end{pmatrix} \right) = \alpha \begin{pmatrix} -\sqrt{\frac{1-p}{p}} \\ -\sqrt{\frac{p}{1-p}} \end{pmatrix}.\end{aligned}$$

Hence if $\mathbf{q} = \text{liability}(g, \text{price}(g, p))$,

$$\begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \text{liability}(g, \text{price}(g, p)) = -\alpha \begin{pmatrix} \sqrt{\frac{1-\text{price}(g,p)}{\text{price}(g,p)}} \\ \sqrt{\frac{\text{price}(g,p)}{1-\text{price}(g,p)}} \end{pmatrix}$$

which implies $q_1 \cdot q_2 = \alpha^2 = x_1 \cdot x_2$.

For the if direction, price of market at q is given by $(g')^{-1}(q_1 - q_2)$, call this p.

$$q_1 - q_2 = g'(p) = -\alpha \frac{1-2p}{\sqrt{p(1-p)}}$$

Protocol 4 Uniswap V2

global constants β.
global variables $\mathbf{x} \in \mathbb{R}^2$, $k \in \mathbb{N}$, $\{\alpha^i \in \mathbb{R}_{\geq 0}\}_{i=0}^{k}$.
$\mathsf{price}(\mathbf{x}) := \frac{x_2}{x_1+x_2}$
function Initialize($\mathbf{x}^0 \in \mathbb{R}^2, \beta$)
    $(k, \beta) \leftarrow (0, \beta)$
    $\alpha^0 \leftarrow \sqrt{x_1^0 \cdot x_2^0}$
function RegisterLP($i = k + 1$)
    $(k, q^i, \alpha^i) \leftarrow (k + 1, 0, 0)$
function ModifyLiquidity($i \in \mathbb{N}, \alpha' \geq 0$)
    $p = \mathsf{price}(\mathbf{x})$
    request $\mathbf{x}' = \left((\alpha' - \alpha^i)\sqrt{\frac{1-p}{p}}, (\alpha' - \alpha^i)\sqrt{\frac{p}{1-p}}\right)$ from LP i.
    $(\mathbf{x}, \alpha^i) \leftarrow (\mathbf{x} + \mathbf{x}', \alpha')$
function ExecuteTrade($\mathbf{r} \in \mathbb{R}^2$)
    check $\varphi(\mathbf{x}) = \varphi(\mathbf{x} - \mathbf{r})$, where $\varphi(\mathbf{x}) = x_1 x_2$.
    pay $\frac{\beta\alpha^i}{\alpha}(-\mathbf{r})_+$ to LP i, where $\alpha = \sum_i \alpha^i$.
    $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{r}$.

Solving this, we get that $p = \frac{q_2}{q_1+q_2}$.

$$\begin{aligned}\mathsf{liability}(g, \mathsf{price}(g, p)) &= \mathsf{liability}(g, \frac{q_2}{q_1 + q_2}) \\ &= \alpha \begin{pmatrix} -\sqrt{\frac{q_1}{q_2}} \\ -\sqrt{\frac{q_2}{q_1}} \end{pmatrix} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}.\end{aligned}$$

The last line uses the fact that $q_1 \cdot q_2 = \alpha^2$. □

Proposition 7. *Protocol 4 is equivalent to Protocol 3 for* $\mathcal{G}_{\text{init}} = \{\alpha g_0 \mid \alpha > 0\}$, $\mathcal{G}_{\text{LP}} = \{\alpha g_0 \mid \alpha \geq 0\}$ *where* $g_0(p) = -2\sqrt{p(1-p)}$, $\mathsf{fee}(\mathbf{r}, \mathbf{q}, g) = \beta\mathbf{r}_+$ *and* $\mathsf{fee}_i(\mathbf{r}, \mathbf{q}, \{g_i\}_i) = \frac{\beta\alpha_i}{\alpha}\mathbf{r}_+$ *for* $\beta > 0$.

Proof. Showing that Protocol 3 gives Protocol 4, for the choices of g specified, involves showing the equivalence of three specific components. That is, we wish to show that the initialization check in Line 7 of Protocol 3 is satisfied and that the request vectors in ModifyLiquidity and ExecuteTrade routines match.

First, we note that modifying $\mathbf{x}$ preserves the invariant $x_1 \cdot x_2 = \alpha^2$ for some α in Protocol 4. As shown in the proof of Proposition 6, $(g')^{-1}(q_1 - q_2) = \mathsf{price}(g_0, \mathbf{q}) = \frac{q_2}{q_1+q_2}$ where $\mathbf{q} = -\mathbf{x}$, showing that the Uniswap price function is a special case.

For the initialization phase, $x_1^0 \cdot x_2^0 = (\alpha^0)^2$ and Proposition 6 give us that $\mathsf{liability}(g_0, \mathsf{price}(g_0, q^0)) = \mathbf{q}^0$ hence satisfying the check. By induction, this statement holds for all states $\mathbf{q}$ i.e. $\mathbf{q} = \mathsf{liability}(g, \mathsf{price}(g, \mathbf{q}))$.

A liquidity change of α^i to α' at price p reflects a change of g_i from $-2 \cdot \alpha^i\sqrt{p(1-p)}$ to $\hat{g} = -2 \cdot \alpha'\sqrt{p(1-p)}$ in Uniswap V2. The quantity of assets requested by Protocol 3 is given by

$$\begin{aligned}\mathbf{r}^i &= -(\text{liability}(\hat{g}, p) - \mathbf{q}^i) = -(\text{liability}(\hat{g}, p) - \text{liability}(g_i, p)) \\ &= (\alpha' - \alpha^i)\begin{pmatrix}\sqrt{\frac{1-p}{p}} \\ \sqrt{\frac{p}{1-p}}\end{pmatrix} = x'\end{aligned}$$

Now, we show that the check in Line 19 of Protocol 3 for the given g gives us the condition $x_1 \cdot x_2 = (x_1 - r_1) \cdot (x_2 - r_2)$ that appears in Uniswap V2.

Let $\mathbf{q} = \sum_i \mathbf{q}^i$ and $\mathbf{r} = \sum_i \mathbf{r}^i$. The check in Protocol 3 can be rewritten as

$$\begin{aligned}\mathbf{q} + \mathbf{r} &= \text{liability}\left(\sum_{j=0}^{k} g_j, \hat{p}\right) \\ &= \text{liability}\left(\sum_{j=0}^{k} g_j, \text{price}\left(\sum_{j=0}^{k} g_j, \mathbf{q} + \mathbf{r}\right)\right)\end{aligned}$$

From this, by applying Proposition 6, $(q_1 + r_1)(q_2 + r_2) = (-x_1 + r_1)(-x_2 + r_2) = \alpha^2 = x_1 \cdot x_2$. Again the note the difference in sign conventions for liability and reserves.

The last fact we want to show to complete the proof is that $\mathbf{r}^i = \frac{\alpha^i}{\alpha}\mathbf{r}$ for Uniswap V2. This can be obtained by observing that $\mathbf{r} = \text{liability}(g, \hat{p}) - \mathbf{q} = \text{liability}(g, \hat{p}) - \text{liability}(g, p)$ from Proposition 6 and $g = \frac{\alpha^i}{\alpha} g_i$ being true for Uniswap V2.

□

C.3 Liquidity vectors for the general bucketing mechanism

In this section, we derive the expressions for the liquidity vectors for general bucketing mechanisms. We are given, for a specified g, an $\ell^{(j)}$ function of the form below,

$$\ell^{(j)}(p) = g''(p)\mathbf{1}_{[a_j, b_j]}(p) = \begin{cases}0 & p < a_j \\ g''(p) & p \in [a_j, b_j] \\ 0 & p > b_j\end{cases}.$$

Then, $(\hat{g}^{(j)})'(p)$ is given by

$$(\hat{g}^{(j)})'(p) = \int_0^p \ell^{(j)}(s)\, ds = \begin{cases}0 & p < a_j \\ g'(p) - g'(a_j) & p \in [a_j, b_j] \\ g'(b_j) - g'(a_j) & p > b_j\end{cases}.$$

We integrate from 0 to p to see that

$$\hat{g}^{(j)}(p) = \int_0^p (\hat{g}^{(j)})'(s)ds = \begin{cases}0 & p < a_j \\ g(p) - g(a_j) - g'(a_j)(p - a_j) & p \in [a_j, b_j] \\ (g'(b_j) - g'(a_j))(p - b_j) + g(b_j) - g(a_j) - g'(a_j)(b_j - a_j) & p > b_j\end{cases}.$$

Then,

$$\begin{aligned}g^{(j)}(p) &= \hat{g}^{(j)}(p) - p\hat{g}^{(j)}(1) \\ &= \begin{cases}p(g(a_j) - g(b_j) - g'(a_j)(a_j - 1) + g'(b_j)(b_j - 1)) & p < a_j \\ g(p) + g(a_j)(p - 1) - pg(b_j) - a_j g'(a_j)(p - 1) + pg'(b_j)(b_j - 1) & p \in [a_j, b_j] \\ (p - 1)(g(a_j) - g(b_j) - a_j g'(a_j) + b_j g'(b_j)) & p > b_j\end{cases}.\end{aligned}$$

From this we can calculate the liability vector as follows

$$\begin{aligned}
\text{liability}(g^{(j)}, p) &= (g'(p), 0) + (g(p) - pg'(p))\mathbf{1} \\
&= \begin{cases} \begin{pmatrix} g(a_j) - g(b_j) - g'(a_j)(a_j - 1) + g'(b_j)(b_j - 1) \\ 0 \end{pmatrix} & p < a_j \\ \begin{pmatrix} g(p) - g(b_j) - g'(p)(p-1) + g'(b_j)(b_j - 1) \\ g(p) - g(a_j) - pg'(p) + a_j g'(a_j) \end{pmatrix} & p \in [a_j, b_j] \\ \begin{pmatrix} 0 \\ g(b_j) - g(a_j) + a_j g'(a_j) - b_j g'(b_j) \end{pmatrix} & p > b_j \end{cases} \\
&= \begin{cases} \begin{pmatrix} S(a_j, 1) - S(b_j, 1) \\ 0 \end{pmatrix} & p < a_j \\ \begin{pmatrix} S(p, 1) - S(b_j, 1) \\ S(p, 0) - S(a_j, 0) \end{pmatrix} & p \in [a_j, b_j] \\ \begin{pmatrix} 0 \\ S(b_j, 0) - S(a_j, 0) \end{pmatrix} & p > b_j \end{cases} \\
&= \begin{pmatrix} S_g(\max\{a_j, p\}, 1) - S_g(\max\{b_j, p\}, 1) \\ S_g(\min\{b_j, p\}, 0) - S_g(\min\{a_j, p\}, 0) \end{pmatrix}
\end{aligned}$$

where $S(p, y) = g(p) + g'(p)(y - p)$.

C.4 A bucketing scheme for logarithmic market scoring rule (LMSR)

Here, we apply the techniques of the previous section to consider an interesting new protocol. What if we used g from LMSR but with a bucketing scheme similar to Uniswap V3? That is, LPs can deposit according to $g(p) = p \log p + (1 - p) \log(1 - p)$ on discrete buckets analogous to what we see in Uniswap V3. We see that $g'(p) = \log p - \log(1 - p) = \log\left(\frac{p}{1-p}\right)$, so

$$g^{(j)}(p) = \begin{cases} p \log \frac{a_j}{b_j} & p < a_j \\ p \log \frac{p}{b_j} + (1 - p) \log \frac{(1-p)}{(1-a_j)} & p \in [a_j, b_j] \\ (1 - p) \log \frac{(1-b_j)}{(1-a_j)} & p > b_j \end{cases}.$$

$$\begin{aligned}
\text{liability}(g^{(j)}, p) &= (g'(p), 0) + (g(p) - pg'(p))\mathbf{1} \\
&= \begin{cases} \begin{pmatrix} \log \frac{a_j}{b_j} \\ 0 \end{pmatrix} & p < a_j \\ \begin{pmatrix} \log \frac{p}{b_j} \\ \log \frac{(1-p)}{(1-a_j)} \end{pmatrix} & p \in [a_j, b_j] \\ \begin{pmatrix} 0 \\ \log \frac{(1-a_j)}{(1-b_j)} \end{pmatrix} & p > b_j \end{cases}.
\end{aligned}$$

C.5 Discussion on Uniswap V3

Readers familiar with the original protocol may recognize Protocol 5 as Uniswap V3 mechanics but with minor changes coming from using normalized prices. Line 14 comes from [Fan et al., 2022]'s analysis of Uniswap V3, and Line 19 comes from the shifted reserve curve characteristic of Uniswap V3 as seen in both [Fan et al., 2022] and [Adams et al., 2021]. For clarity, we want to reiterate that Fan

Protocol 5 Uniswap V3

global constants β, m, $\{B^j = [a_j, b_j]\}_{j=0}^{m}$.
global variables $\mathbf{x} \in \mathbb{R}^2$, $k \in \mathbb{N}$, $\{\alpha^{ij} \in \mathbb{R}_{\geq 0}\}_{i \in \{0,\ldots,k\}, j \in \{0,\cdots,m\}}$
function price($\mathbf{x} \in \mathbb{R}^2$)
    return $\dfrac{x_2+\alpha_j\sqrt{\frac{a_j}{1-a_j}}}{x_1+\alpha_j\sqrt{\frac{1-b_j}{b_j}}+x_2+\alpha_j\sqrt{\frac{a_j}{1-a_j}}}$
function Initialize($\mathbf{x} \in \mathbb{R}^2, \alpha > 0, \beta$)
    $(k, \beta) \leftarrow (0, \beta)$
    ModifyLiquidity$(0, \mathbf{x}, \alpha, [0, 1])$ ▷ We technically have to split this into function call for each price interval.
function RegisterLP()
    $k \leftarrow k + 1$
    $\alpha^{kj} \leftarrow 0, \forall j \in \{0, \ldots, m\}$
    return k ▷ ID of the new LP
function ModifyLiquidity($i \in \mathbb{N}, \alpha' \geq 0, j \in \{0, \ldots, m\}$)
    $p = \text{price}(\mathbf{x})$
    request $\mathbf{x}' = \begin{cases} \left((\alpha' - \alpha^{ij})\left(\sqrt{\frac{1-a_j}{a_j}} - \sqrt{\frac{1-b_j}{b_j}}\right), 0\right) & \text{if } p < a_j \\ \left(0, (\alpha' - \alpha^{ij})\left(\sqrt{\frac{b_j}{1-b_j}} - \sqrt{\frac{a_j}{1-a_j}}\right)\right) & \text{if } p > b_j \\ \left((\alpha' - \alpha^{ij})\left(\sqrt{\frac{1-p}{p}} - \sqrt{\frac{1-b_j}{b_j}}\right), (\alpha' - \alpha^{ij})\left(\sqrt{\frac{p}{1-p}} - \sqrt{\frac{a_j}{1-a_j}}\right)\right) & \text{if } p \in [a_j, b_j] \end{cases}$
    $(\mathbf{x}, \alpha^{ij}) \leftarrow (\mathbf{x} + \mathbf{x}', \alpha')$
function ExecuteTrade($\mathbf{r} \in \mathbb{R}^2$)
    Let $p = \text{price}(\mathbf{x})$, $\hat{p} = \text{price}(\mathbf{x} - \mathbf{r})$ [9]
    Let l, u be such that $a_l \leq p \leq b_l$ and $a_u \leq \hat{p} \leq b_u$.
    check
$$\frac{1}{(\sum_{i=0}^{k}\alpha^{il})^2}\left(x_1 + \sum_{i=0}^{k}\alpha^{il}\sqrt{\frac{1-b_l}{b_l}}\right)\left(x_2 + \sqrt{\frac{a_l}{1-a_l}}\sum_{i=0}^{k}\alpha^{il}\right) = \frac{1}{(\sum_{i=0}^{k}\alpha^{iu})^2}\left(x_1 - r_1 + \sum_{i=0}^{k}\alpha^{iu}\sqrt{\frac{1-b_u}{b_u}}\right)\left(x_2 - r_2 + \sqrt{\frac{a_u}{1-a_u}}\sum_{i=0}^{k}\alpha^{iu}\right)$$
    pay $\beta\frac{\sum_j \alpha^{ij}}{\sum_j \sum_o \alpha^{oj}}(-\mathbf{r})_+$ to each LP i where j sums over buckets in $[B^l, B^u]$. ▷ WLOG assume that the B^u bucket comes later than B^l
    $\mathbf{x} \leftarrow \mathbf{x} - \mathbf{r}$

et al. [2022] uses an exchange rate price $\hat{p}$, and we use its normalized version p. The two quantities are related by $\hat{p} = \frac{p}{1-p}$.

Proposition 8. *For* $\mathcal{G}_{\text{init}} = \{\sum_j \alpha_j g^{(j)} \mid \alpha_j > 0\}$, $\mathcal{G}_{\text{LP}} = \{\sum_j \alpha_j g^{(j)} \mid \alpha_j \geq 0\}$ *where*

$$g^{(j)}(p) = \begin{cases} p(\sqrt{\frac{1-b_j}{b_j}} - \sqrt{\frac{1-a_j}{a_j}}) & \text{if } p \leq a_j \\ -2\sqrt{p(1-p)} + p\sqrt{\frac{1-b_j}{b_j}} + (1-p)\sqrt{\frac{a_j}{1-a_j}} & \text{if } a_j \leq p \leq b_j \\ (1-p)(\sqrt{\frac{a_j}{1-a_j}} - \sqrt{\frac{b_j}{1-b_j}}) & \text{if } p \geq b_j \end{cases},$$

the vector $\mathbf{q}$ *of liability in Protocol 3 always satisfies* $\left(x_1 + \alpha_j\sqrt{\frac{1-b_j}{b_j}}\right)\cdot\left(x_2 + \alpha_j\sqrt{\frac{a_j}{1-a_j}}\right) = \alpha_j^2$ *for some* $\alpha_j > 0$, *where* $\mathbf{x} = -\mathbf{q}$.

Proof. Observe that any change in liability vector in the ModifyLiquidity or Initialize phases results in a liability vector that results in the vector taking the form liability(g, p) for some $g \in \mathcal{G}_{\text{LP}}$, $p \in [0, 1]$. Let this g be $\alpha_j g^{(j)}(p)$ where α_j is the total liquidity in jth price interval which is $\sum_i \alpha^{ij}$ in Protocol 5. We have

$$g'(p) = \begin{cases} \alpha_j(\sqrt{\frac{1-b_j}{b_j}} - \sqrt{\frac{1-a_j}{a_j}}) & \text{if } p \le a_j \\ \alpha_j(-\frac{1-2p}{\sqrt{p(1-p)}} + \sqrt{\frac{1-b_j}{b_j}} - \sqrt{\frac{a_j}{1-a_j}}) & \text{if } a_j \le p \le b_j \\ (\sqrt{\frac{b_j}{1-b_j}} - \sqrt{\frac{a_j}{1-a_j}}) & \text{if } p \ge b_j \end{cases}.$$

Solving for liability$(g, p) = (g(p) - p \cdot g'(p)\mathbf{1} + \begin{pmatrix} g'(p) \\ 0 \end{pmatrix}$ for these three cases gives us

$$\text{liability}(g, p) = \begin{cases} \alpha_j \begin{pmatrix} \sqrt{\frac{1-b_j}{b_j}} - \sqrt{\frac{1-a_j}{a_j}} \\ 0 \end{pmatrix} & \text{if } p \le a_j \\ \alpha_j \begin{pmatrix} -\sqrt{\frac{1-p}{p}} + \sqrt{\frac{1-b_j}{b_j}} \\ -\sqrt{\frac{p}{1-p}} + \sqrt{\frac{a_j}{1-a_j}} \end{pmatrix} & \text{if } a_j \le p \le b_j \\ \alpha_j \begin{pmatrix} 0 \\ \sqrt{\frac{a_j}{1-a_j}} - \sqrt{\frac{b_j}{1-b_j}}) \end{pmatrix} & \text{if } p \ge b_j \end{cases}.$$

In each of these cases, with a bit of algebra we can see that

$$\left(q_1 - \alpha_j\sqrt{\frac{1-b_j}{b_j}}\right)\cdot\left(q_2 - \alpha_j\sqrt{\frac{a_j}{1-a_j}}\right) = \left(x_1 + \alpha_j\sqrt{\frac{1-b_j}{b_j}}\right)\cdot\left(x_2 + \alpha_j\sqrt{\frac{a_j}{1-a_j}}\right) = \alpha_j^2,$$

as $\mathbf{q}$ in Protocol 3 is liability which is negative of reserves in Protocol 5. □

Proposition 9. *Protocol 5 is equivalent to Protocol 3 for* $\mathcal{G}_{\text{init}} = \{\sum_j \alpha_j g^{(j)} \mid \forall j\, \alpha_j > 0\}$, $\mathcal{G}_{\text{LP}} = \{\sum_j \alpha_j g^{(j)} \mid \forall j\, \alpha_j \ge 0\}$ *where*

$$g^{(j)}(p) = \begin{cases} p(\sqrt{\frac{1-b_j}{b_j}} - \sqrt{\frac{1-a_j}{a_j}}) & \text{if } p \le a_j \\ -2\sqrt{p(1-p)} + p\sqrt{\frac{1-b_j}{b_j}} + (1-p)\sqrt{\frac{a_j}{1-a_j}} & \text{if } a_j \le p \le b_j \\ (1-p)(\sqrt{\frac{a_j}{1-a_j}} - \sqrt{\frac{b_j}{1-b_j}}) & \text{if } p \ge b_j \end{cases}.$$

Proof. To show that Protocol 3 gives us Protocol 5, for the choices of g specified, involves showing three specific components are equivalent. They include showing that the initialization check satisfies and the request vectors in ModifyLiquidity and ExecuteTrade routines match.

[9]We also note that price in Uniswap V3 is not calculated on demand like we do here but is a state variable thats maintained throughout the implementation. Uniswap V3 also implements the Line 19 by passing through all price ranges consecutive to p and checking which price interval satisfies this check. We abstract away from this to avoid being caught up in technicalities as this is not the main problem we tackle.

We saw α_j to mean total liquidity in jth interval i.e. $\sum_{i=0}^{k} \alpha^{ij}$ and let $g(p) = \alpha_j g^{(j)}$. Firstly, to show the initialization check, we derive that

$$(g')^{-1}(q_1 - q_2) = (g')^{-1}(x_2 - x_1) = \frac{x_2 + \alpha_j \sqrt{\frac{a_j}{1-a_j}}}{x_1 + \alpha_j \sqrt{\frac{1-b_j}{b_j}} + x_2 + \alpha_j \sqrt{\frac{a_j}{1-a_j}}}.$$

We consider only the case when this price falls in a bucket j as for all other buckets, $(g')^{-1}$ would not give a definitive result.

$$\begin{aligned}
\text{liability}(g_0, \text{price}(g_0, q^0)) &= \alpha_j \begin{pmatrix} -\sqrt{\frac{x_1^0 + \alpha_j \sqrt{\frac{1-b_j}{b_j}}}{x_2^0 + \alpha_j \sqrt{\frac{a_j}{1-a_j}}}} + \sqrt{\frac{1-b_j}{b_j}} \\ -\sqrt{\frac{x_2^0 + \alpha_j \sqrt{\frac{a_j}{1-a_j}}}{x_1^0 + \alpha_j \sqrt{\frac{1-b_j}{b_j}}}} + \sqrt{\frac{a_j}{1-a_j}} \end{pmatrix} \\
&= \alpha_j \begin{pmatrix} -\frac{x_1^0 + \alpha_j \sqrt{\frac{1-b_j}{b_j}}}{\alpha_j} + \sqrt{\frac{1-b_j}{b_j}} \\ -\frac{x_2^0 + \alpha_j \sqrt{\frac{a_j}{1-a_j}}}{\alpha_j} + \sqrt{\frac{a_j}{1-a_j}} \end{pmatrix} \\
&= \begin{pmatrix} -x_1^0 \\ -x_2^0 \end{pmatrix} = \begin{pmatrix} q_1^0 \\ q_2^0 \end{pmatrix}
\end{aligned}$$

The first two steps are a result of Proposition 8.

Now, we show that the quantity of assets requested from a LP to change the liquidity level from α^{ij} to α' given by Line 14 in Protocol 5 is equivalent to Line 11 of Protocol 3.

We first note that the price p, does not change in this operation, as liquidity must be added by keeping the price constant. Another way to see it is that $\text{price}(g_0, \mathbf{q}^0)$ remains unchanged. A liquidity change of α^{ij} to α' at price $p \in B_j$ reflects a change of g_i from $\alpha^{ij} g^{(j)}$ to $\hat{g} = \alpha' g^{(j)}$ in Uniswap V3. The quantity of asset requested by Protocol 3 is given by

$$\begin{aligned}
-(\text{liability}(\hat{g}, p) - \mathbf{q}^i) &= \text{liability}(g_i, p) - \text{liability}(\hat{g}, p) \\
&= \begin{cases} \alpha^{ij} \left(-\sqrt{\frac{1-a_j}{a_j}} + \sqrt{\frac{1-b_j}{b_j}}, 0 \right) - \alpha' \left(-\sqrt{\frac{1-a_j}{a_j}} + \sqrt{\frac{1-b_j}{b_j}}, 0 \right) & \text{if } p < a_j \\ \alpha^{ij} \left(0, \sqrt{\frac{a_j}{1-a_j}} + \sqrt{\frac{b_j}{1-b_j}} \right) - \alpha' \left(0, \sqrt{\frac{a_j}{1-a_j}} + \sqrt{\frac{b_j}{1-b_j}} \right) & \text{if } p > b_j \\ (\alpha^{ij} - \alpha') \left(-\sqrt{\frac{1-p}{p}} + \sqrt{\frac{1-b_j}{b_j}}, \sqrt{\frac{p}{1-p}} + \sqrt{\frac{a_j}{1-a_j}} \right) & \text{if } p \in [a_j, b_j] \end{cases} \\
&= \begin{cases} \left((\alpha' - \alpha^{ij}) \left(\sqrt{\frac{1-a_j}{a_j}} - \sqrt{\frac{1-b_j}{b_j}} \right), 0 \right) & \text{if } p < a_j \\ \left(0, (\alpha' - \alpha^{ij}) \left(\sqrt{\frac{b_j}{1-b_j}} - \sqrt{\frac{a_j}{1-a_j}} \right) \right) & \text{if } p > b_j \\ \left((\alpha' - \alpha^{ij}) \left(\sqrt{\frac{1-p}{p}} - \sqrt{\frac{1-b_j}{b_j}} \right), (\alpha' - \alpha^{ij}) \left(\sqrt{\frac{p}{1-p}} - \sqrt{\frac{a_j}{1-a_j}} \right) \right) & \text{if } p \in [a_j, b_j] \end{cases}
\end{aligned}.$$

Now, we show that the check in Line 15 of Protocol 3 for the given g gives us the condition $\left(x_1 + \alpha_l \sqrt{\frac{1-b_l}{b_l}} \right) \left(x_2 + \sqrt{\frac{a_l}{1-a_l}} \alpha_l \right) = \left(x_1 - r_1 + \alpha_u \sqrt{\frac{1-b_u}{b_u}} \right) \left(x_2 - r_2 + \sqrt{\frac{a_u}{1-a_u}} \alpha_u \right)$ where $\alpha_x = \sum_{i=0}^{k} \alpha^{ix}$ that appears in Uniswap V3.

Let $\mathbf{q} = \sum_i \mathbf{q}^i$ and $\mathbf{r} = \sum_i \mathbf{r}^i$. The check in Protocol 3 can be rewritten as

$$\begin{aligned}\mathbf{q} + \mathbf{r} &= \text{liability}\left(\sum_{i=0}^{k} g_i, \hat{p}\right) \\ &= \text{liability}\left(\sum_{i=0}^{k} \alpha^{iu} g^{(j)}, \hat{p}\right) \\ &= \sum_{i=0}^{k} \alpha^{iu} \text{liability}(g^{(j)}, \hat{p}).\end{aligned}$$

From Proposition 8, we can see that

$$\left(x_1 - r_1 + \left(\sum_{i=0}^{k} \alpha^{iu}\right)\sqrt{\frac{1-b_u}{b_u}}\right) \cdot \left(x_2 - r_2 + \left(\sum_{i=0}^{k} \alpha^{iu}\right)\sqrt{\frac{a_u}{1-a_u}}\right) = \left(\sum_{i=0}^{k} \alpha^{iu}\right)^2$$

and

$$\left(x_1 + \sum_{i=0}^{k} \alpha^{il}\sqrt{\frac{1-b_l}{b_l}}\right) \cdot \left(x_2 + \sum_{i=0}^{k} \alpha^{il}\sqrt{\frac{a_l}{1-a_l}}\right) = \left(\sum_{i=0}^{k} \alpha^{il}\right)^2,$$

proving what we need.

The last fact we want to show to complete the proof is that $r^i = \frac{\alpha^i}{\alpha} r$ for Uniswap V3, where $\alpha_i = \sum_j \alpha^{ij}$, $\alpha = \sum_j \sum_{o=0}^{k} \alpha^{oj}$ for j summing over baskets B^l to B^u. We see that

$$\begin{aligned}\mathbf{r}^i &= \text{liability}(g_i, \hat{p}) - \mathbf{q}^i \\ &= \alpha^i(\text{liability}(g^{(j)}, \hat{p}) - \text{liability}(g^{(j)}, p)) \\ &= \frac{\alpha^i}{\alpha}\left(\text{liability}(g, \hat{p}) - \mathbf{q}\right) \quad \text{where } g = \sum_{i=0}^{k} g_i \text{ and } \mathbf{q} = \sum_{i=0}^{k} \mathbf{q}^i \\ &= \frac{\alpha^i}{\alpha}\mathbf{r},\end{aligned}$$

as desired. □

D ON NEW PROTOCOLS

In some cases, the generalized bucketing scheme we provide in § 7.3 is still overly restrictive on the expressivity of LPs, depending on the size of the price intervals. It may also be unnatural for LPs to specify their liquidity allocation via discontinuous liquidity functions. Fortunately, our general protocol easily allows one to generate more expressive restricted protocols, for particular families of C (equivalently g for two assets) functions which still allow for efficient computations. For example, one could use "soft" liquidity buckets where liquidity continuously "fades" in and out around a target price a_j.

In the case of n assets, one can generalize the idea of buckets, though now the task of partitioning the $d = n - 1$-dimensional price space becomes nontrivial. One promising approach would be to partition into a cell complex of convex polytopes, specifically a power diagram, [Aurenhammer, 1987]. One could define indicator functions as above to allow LPs to allocate liquidity uniformly (or according to some base shape) across each of these regions. We also give a piecewise linear protocol which could be similarly adapted to these polyhedral regions, where the linear pieces align with the regions.

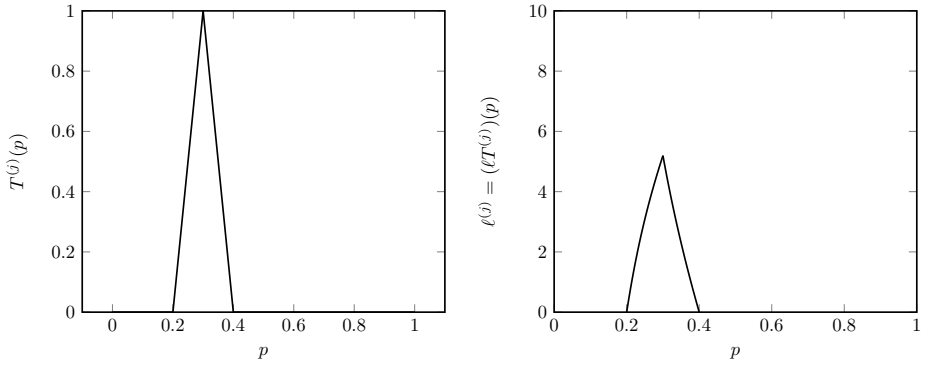


Fig. 2. $T^{(j)}(p)$ and $\ell^{(j)}(p)$ for $a_j = 0.3$

D.1 Soft buckets allowing richer functions than discrete buckets

Define a set $a_0 < a_1 = 0 < a_2 < \cdots < a_k = 1 < a_{k+1}$. Our "buckets" B_j will be supported on the interval $[a_{j-1}, a_{j+1}]$. To capture the continuous fading, define the triangular function $T^{(j)} : [0,1] \to [0,1]$ as $T^{(j)}(p) = \left(\frac{p-a_{j-1}}{a_j - a_{j-1}}\right)\mathbf{1}_{p\in[a_{j-1},a_j]} + \left(\frac{a_{j+1}-p}{a_{j+1}-a_j}\right)\mathbf{1}_{p\in[a_j,a_{j+1}]}$. The corresponding base liquidity functions are then $\ell^{(j)} = (\ell T^{(j)})(p)$ where $\ell(p) = 2(p(1-p))^{-\frac{3}{2}}$, to again use the same base "shape" as the constant product invariant $\varphi_\alpha(\mathbf{x}) = x_1 x_2 = \alpha^2$. Let $g^{(j)}$ then be the corresponding base generating function for $\ell^{(j)}$.

Now letting $\mathcal{G}_{\text{init}} = \{\sum_j \alpha_j g^{(j)} \mid \forall j\, \alpha_j > 0\}$ and $\mathcal{G}_{\text{LP}} = \{\sum_j \alpha_j g^{(j)} \mid \forall j\, \alpha_j \geq 0\}$, Protocol 3 gives a new liquidity provisioning protocol. While the corresponding computations appear to be essentially as light-weight as Uniswap V3, the liquidity function is better behaved. We can characterize the possible liquidity functions $\{g'' \mid g \in \mathcal{G}_{\text{LP}}\}$ as the functions $f\ell$ where f is an arbitrary nonnegative continuous function that is affine on each interval $[a_j, a_{j+1}]$. (Simply take $\alpha_j = f(a_j)$.) Thus, we have in particular that liquidity is always continuous in the price in this new protocol. One can additionally innovate from here, adding more flexibility, while taking care to keep the various computations manageable.

D.2 Piecewise linear market maker

To demonstrate the robustness of our protocol, in this section we consider a market where the liquidity curves ℓ are not well-defined. Yet, we show that our general scheme still gives rise to a sensible market maker.

Consider a market maker that restricts the possible prices to $\{p^i\}_{i=1}^m$ where $0 < a_1 < \cdots < a_m < 1$. Consider a $(g^{(j)})'$ of the form

$$(g^{(j)})'(p) = \begin{cases} a_j - 1 & p \leq a_j \\ [a_j - 1, a_j] & p = a_j \\ a_j & p \geq a_j \end{cases}.$$

$g'(p) = \sum_j \alpha_j (g^{(j)})'(p)$ where $\alpha_j = \sum_i \alpha^{ij}$.

Observe that liquidity is infinity at each price. As prices only move discretely in this market, the market needs to be parameterized by reserves held / liability vector $\mathbf{q}$. Hence the state of the market is $\{\alpha_{ij}\}_{i\in\{0,\cdots,k\}, j\in\{1,\cdots,m\}}, \{\mathbf{q}^i\}_{i\in\{0,\cdots,k\}}$.

If the current market reserves are $\mathbf{q}$, the price in this market can be derived from the below formula

$$\text{price}(g, \mathbf{q}) = a_{j^*} \text{ where } j^* = \arg\max_{j'} \text{ s.t. } \left\{ \mathbf{q} - \sum_{j=1}^{m} \alpha_j (a_j - 1) + \sum_{j=1}^{j'-1} \alpha_j \geq 0 \right\}.$$

For a given $\mathbf{q}$, let there exist $y \in [0, 1)$ such that $\mathbf{q} = \sum_j \alpha_j a_j - \sum_{j=j^*}^{m} \alpha_j + y\alpha_{j^*}$. We can maintain this relative liquidity in a bucket after changing the liquidity curves.

For the ModifyLiquidity function, let LP i wants to change its liquidity levels from α_{ij} to α'_{ij} for price bucket j. The new reserves that LP i needs to deposit is given by $\hat{\mathbf{q}} - \mathbf{q} = (\alpha'_{ij} - \alpha_{ij})(a_j - \mathbf{1}_{j \geq j^*} + y\mathbf{1}_{j=j^*})$

ExecuteTrade takes in a trade r, computes the new price $\hat{p}$ corresponding to the new liability vector $\mathbf{q} + \mathbf{r}$ using the above given formula.

E ON INCOMPLETE MARKETS

In the main body, we stated our general framework in terms of complete markets. Here, we describe how Protocol 1 works for incomplete markets. We follow Abernethy et al. [2013] to define an incomplete market with k securities by specifying a function $\phi : \{1, \ldots, n\} \to \mathbb{R}^k$ where the ith security is ϕ_i. In Abernethy et al. [2013], cost functions are given by the convex conjugate $\hat{C} : \mathbb{R}^k \to \mathbb{R}$ of some $F : \Pi \to \mathbb{R}$ where $\Pi = \text{conv}(\{\phi(i) : i \in \{1, \ldots, n\}\})$ is the price space (where F is denoted by R in their paper). Alternatively, we can have securities live in $\mathbb{R}^n$, and *extend* F to $G : \Delta_n \to \mathbb{R}$ with

$$G(\mathbf{p}) = F(E_{\mathbf{p}}\phi), \quad E_{\mathbf{p}}\phi = \sum_{i=1}^{n} p_i \phi_i,$$

where $E_{\mathbf{p}}$ is the expected payoff of the securities. Then, we can take the convex conjugate to get a cost function C for Protocol 1. We can observe that G is flat along directions in which the expected payoff is constant, so C has zero liquidity along these dual directions. Hence, the share vectors $\mathbf{q}$ are effectively constrained to span $(\{\phi_1, \ldots, \phi_k\})$. In this sense, incomplete markets are a special case of Protocol 1.

It is worth noting that it is easier to work in the lower-dimensional space $\mathbb{R}^k$ directly, as Abernethy et al. [2013] also do. For our purposes, the protocols and equivalence results are easier to state, and Abernethy et al. [2013] provide similar equivalences but with stronger assumptions on F and $\hat{C}$, namely strict convexity of F. Since we do not require G to be strictly convex in Theorem 1, our equivalence theorem applies to the incomplete market setting.