Mitigando Slippage em Serviços RFQ

Luan Martins Felix[1], **Daniel Sadoc Menasché**[2]

[1] Parfin – Parity Financial Ltd.
London, U.K.

[2]Instituto de Computação – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68.530 – 21941-590 – Rio de Janeiro – RJ – Brasil

luan.martins@parfin.io, sadoc@dcc.ufrj.br

Abstract. *DThis study describes an approach to mitigating the slippage issue in RFQ services. Slippage refers to the difference between the expected price and the executed price in a quote for buying or selling a certain amount of a currency pair. The two main strategies considered were spread adjustment and quote cancellation. The study involves collecting and processing data to statistically infer the optimal spread, which simultaneously maintains a competitive price, reduces company losses, and complies with the SLA. The result is the establishment of a simple and reasonable heuristic for approaching the problem and a webapp for visualizing the proposed heuristic.*

Resumo. *O presente trabalho descreve um estudo voltado para a mitigação do problema de slippage em serviços RFQ. Slippage refere-se à diferença entre o preço esperado e o preço executado em uma cotação de compra ou venda de uma determinada quantia em um par de moedas. As duas principais estratégias consideradas foram o ajuste do spread e o cancelamento de cotações. O estudo envolve a coleta e o processamento de dados para inferir estatisticamente o spread ideal que, simultaneamente, mantém um preço competitivo, reduz os prejuízos da empresa e respeita o SLA. O resultado é o estabelecimento de uma heurística simples e razoável para abordar o problema e um webapp para visualizar a heurística proposta.*

1. Introdução

Desde sua criação, o mercado de criptomoedas tem se destacado por sua natureza inovadora e dinâmica, sendo a volatilidade de preços uma manifestação da constante adaptação às transformações, tanto tecnológicas quanto econômicas. Especificamente, serviços *Web3* que atuam sob o modelo *RFQ* (*Request For Quote*), no qual são oferecidas cotações de câmbio de moedas e criptomoedas com validade limitada aos usuários, são diretamente impactados por essa volatilidade, resultando em *slippage* — a diferença entre o preço esperado, prometido ao usuário, e o preço executado, pago pelo serviço. Apesar de pequeno em transações individuais, o *slippage* pode ter impacto significativo em volumes diários de centenas de milhões de dólares, tornando sua mitigação essencial. Este estudo busca desenvolver uma heurística simples e razoável para mitigar esse problema e atingir essa economia.

Nosso objetivo é desenvolver uma heurística capaz de mitigar o impacto do *slippage* para empresas que oferecem serviços *RFQ*. A proposta é estabelecer um ponto de

partida que possa ser testado, validado e aprimorado como uma prova de conceito para implementação em serviços reais. Dada a complexidade do mercado financeiro e do problema em questão, esta abordagem simplificada foi escolhida como um primeiro passo rumo a uma solução mais elaborada. Embora este estudo seja motivado por um serviço da Parfin [Parfin 2025], suas estratégias são aplicáveis a qualquer serviço *RFQ*.

A literatura existente concentra-se principalmente em outro modelo de mercado, o *AMM* (*Automated Market Makers*, vide Seção 3): [Aldridge 2022] analisa a diferença entre *trading* tradicional e cripto, investigando como melhor estimar *slippage* para maximizar lucros; [Lim 2022] propõe um novo protocolo baseado no *Uniswap V3*, utilizando aprendizado de máquina para reduzir o *slippage* enfrentado por *traders* de criptomoedas; [Shah et al. 2020] apresenta a Slipswap, uma aplicação web (também disponível via *API*) que otimiza a conversão entre criptomoedas, reduzindo perdas por *slippage* em mercados altamente voláteis. Até onde sabemos, não há estudos que abordem a mitigação do *slippage* no modelo *RFQ*.

Roteiro. O restante deste artigo está organizado da seguinte forma. Na Seção 2, abordamos a literatura e o estado da arte vigente. Na Seção 3, aprofundamos o funcionamento de serviços *RFQ* e propomos estratégias para mitigar o *slippage*. Em seguida, na Seção 4 descrevemos a coleta e o processamento dos dados necessários para inferir estatisticamente o *spread* ideal. Após isso, na Seção 5 elaboramos, analisamos e validamos a estratégia obtida com base nas amostras coletadas. Na Seção 6, desenvolvemos sobre a implementação do *WebApp*. A Seção 7 conclui com reflexões sobre desenvolvimentos futuros para aprimorar esta abordagem.

2. Trabalhos Relacionados e Estado da Arte

A literatura acadêmica que trata diretamente da mitigação de *slippage* em serviços baseados no modelo *RFQ* ainda é incipiente. Até onde temos conhecimento, este é o primeiro estudo que se dedica especificamente à análise quantitativa do *slippage* em plataformas *RFQ* no mercado de criptoativos, tendo como principal motivação um serviço real oferecido pela Parfin. Embora este artigo se baseie no trabalho de conclusão de curso do primeiro autor [Martins 2023], ele o expande significativamente, tanto na coleta de dados quanto na formalização das estratégias propostas.

Por outro lado, o problema de *slippage* tem sido abordado com mais frequência no contexto dos *Automated Market Makers*, modelo dominante em protocolos de finança descentralizada *(DeFi)*. Aldridge [Aldridge 2022] discute as limitações do modelo *AMM* em termos de perdas por *slippage* e propõe formas de estimar essa variável de forma mais precisa. Lim [Lim 2022] vai além, ao utilizar aprendizado de máquina com reforço profundo para antecipar movimentos de mercado e adaptar dinamicamente a arquitetura do *AMM*, buscando reduzir o impacto do *slippage* em negociações de criptoativos. Shah et al. [Shah et al. 2020] apresentam o *SlipSwap*, um serviço que otimiza a conversão entre tokens em ambientes altamente voláteis, também com o objetivo de mitigar perdas por *slippage*.

Contudo, o cenário é notavelmente distinto no modelo *RFQ*. Em vez de negociações contínuas baseadas em curvas de liquidez, o modelo *RFQ* oferece cotações com tempo de vida limitado, o que exige estratégias específicas para lidar com oscilações rápidas de mercado. Diante da ausência de estudos acadêmicos voltados para esse con-

texto, observamos que, na prática, empresas que atuam com *RFQ* adotam abordagens empíricas centradas na aplicação de *spreads* fixos. Com frequência, utilizam valores como 1 *basis point* (bp) — ou 0,01% — aplicados simetricamente às cotações. Esses valores são calibrados de maneira manual ou semi-automatizada, com base em observações operacionais como o índice de rejeição de ordens, lucro médio por transação, flutuações de volatilidade e até preferências específicas de determinados clientes [Wintermute 2023].

Apesar da simplicidade dessa abordagem, empresas mais sofisticadas vêm adotando soluções técnicas para mitigar os riscos associados à volatilidade e à latência de execução. Uma dessas estratégias é a utilização de *latency buffers*, que ajustam o *spread* com base na latência esperada entre a cotação e sua execução [Cartea et al. 2015, Bouchaud et al. 2018]. Essa prática é comum entre instituições de alta frequência como a Jane Street [Street 2023], que buscam proteger suas cotações contra *latency arbitrage*.

Outra frente explorada é o uso de modelos de previsão de volatilidade de curto prazo, comumente baseados em GARCH [Engle 1982] ou técnicas de aprendizado de máquina [Dixon et al. 2020]. Empresas como a Coinbase relatam o uso de modelos preditivos para ajustar dinamicamente preços e riscos de execução [Coinbase 2022], o que se alinha à proposta deste estudo de tornar o *spread* uma variável responsiva às condições reais de mercado.

Além disso, alguns *market makers* institucionais adotam modelos de impacto de mercado, estimando como a própria execução de ordens pode afetar os preços. A literatura clássica de Almgren e Chriss [Almgren and Chriss 2000] serve de base teórica para essas abordagens, e empresas como a Paradigm adaptaram esses conceitos ao mercado de criptoativos [Research 2023].

Também têm ganhado espaço estratégias baseadas em simulações, como Monte Carlo sobre microestruturas de livros de ordens [Gould et al. 2013, Aoyagi and Ibuka 2021]. A provedora de dados Kaiko, por exemplo, aplica tais métodos para estimar *slippage* e custos de execução em seus relatórios de análise de mercado [Kaiko 2023].

Por fim, abordagens mais experimentais incluem algoritmos de aprendizado por reforço e *multi-armed bandits*, aplicados à calibragem de *spreads* com base em feedback contínuo de lucro e perda [Spooner et al. 2018, Nevmyvaka et al. 2006]. Embora a literatura acadêmica já aborde essas técnicas, empresas como a Radix Trading vêm explorando sua aplicação prática para estratégias de *market making* adaptativo [Trading 2023].

Apesar dessas técnicas promissoras, não identificamos na literatura científica atual estudos que documentem ou avaliem quantitativamente essas estratégias no contexto de serviços *RFQ* para criptoativos. Esse cenário reforça o ineditismo e a relevância do presente trabalho, que propõe uma heurística estatística sistematizada e de fácil implementação, capaz de otimizar o *spread* aplicado em tempo real com base em dados históricos recentes. Ao preencher essa lacuna, o trabalho não apenas contribui para o avanço da literatura na área, como também oferece uma solução prática e aplicável ao ecossistema Web3.

3. Entendendo os serviços RFQ

O modelo *RFQ* tem ganhado destaque em ambientes de negociação institucional, especialmente no contexto de ativos digitais, por combinar rapidez na execução com controle sobre preços. Ao contrário de modelos como *order books* ou *AMMs*, os serviços *RFQ* oferecem cotações com validade limitada, exigindo respostas rápidas dos usuários e decisões estratégicas por parte das plataformas. Nesta seção, detalhamos o funcionamento desses serviços, diferenciando-os dos demais modelos de mercado e estabelecendo os fundamentos para o desenvolvimento de estratégias de mitigação de *slippage*.

3.1. Analisando os modelos de mercado

Na *Web3*, existem três modelos de mercado predominantes: *Order Book*, *AMM* (*Automated Market Makers*) e *RFQ* (*Request for Quote*). O *Order Book* registra todas as ordens de compra e venda, permitindo negociações mais precisas, mas exigindo mais liquidez. O *AMM* usa um sistema automático para calcular preços com base na oferta e demanda, sem precisar de compradores e vendedores diretos. Já o modelo *RFQ* permite que compradores peçam preços antes de fechar uma negociação, garantindo melhores ofertas. *Exchanges* funcionam como casas de câmbio para moedas tradicionais e criptomoedas e geralmente operam com o modelo de *Order Book*, permitindo negociações diretas entre compradores e vendedores. Protocolos de finanças descentralizadas, como Uniswap, utilizam *AMM*s, onde os preços são ajustados automaticamente com base na liquidez disponível. Já plataformas institucionais e de grandes negociações adotam o modelo *RFQ*, garantindo melhores preços ao permitir que compradores solicitem cotações antes de fechar uma transação. Esse estudo escolhe focar em serviços de modelo *RFQ*.

Serviços *RFQ* são complexos e contam com diversas camadas e estratégias para melhor atender seus clientes, como múltiplos provedores e pares de moedas. Para o propósito deste estudo, adotamos uma abordagem simplificada. Consideraremos apenas um provedor, a corretora Bitstamp (vide Seção 4); trabalharemos com o par BTC/USD; a vida útil das cotações oferecidas pelo serviço será de 5 segundos; e assumiremos um acordo de nível de serviço (*Service Level Agreement*, *SLA*), no qual cotações podem ser canceladas, mas não em uma proporção superior a 1%, a fim de preservar a satisfação dos usuários.

Destacamos três pontos sobre o modelo *RFQ*. Primeiro, as cotações podem ser pensadas como ofertas de tempo limitado feitas pelo provedor do serviço ao usuário. Ou seja, cada cotação representa uma oportunidade de compra ou venda de um par de moedas por um preço fixo durante uma janela de tempo. Segundo, o serviço age como uma ponte entre o usuário e o melhor provedor disponível no momento da compra, dependendo da operação em questão. A transação só é efetivamente realizada quando a cotação é aceita. Terceiro, durante a janela de tempo da cotação, o preço oferecido permanece fixo, mesmo que o preço de mercado varie. Denominamos o preço inicialmente oferecido como preço esperado (P_{esp}) e o preço no momento em que a cotação foi aceita como preço executado (P_{exec}).

3.2. Entendendo *slippage*

Agora, podemos definir *slippage* como a diferença entre o preço esperado P_{esp} e o preço executado P_{exec}, isto é, $\textit{slippage} = P_{\text{esp}} - P_{\text{exec}}$. O *slippage* pode ou não ser um problema

para a empresa que oferece serviços *RFQ*. Por exemplo, em uma cotação de compra, se o *slippage* for positivo, o usuário terá comprado a um preço superior ao que o serviço pagou para executá-la, gerando lucro para a empresa. Já se o *slippage* for negativo, o usuário terá comprado a um preço inferior ao pago pela empresa, gerando prejuízo. Cenários análogos se aplicam a cotações de venda. A Tabela 1 evidencia essas situações.

As soluções para mitigar o *slippage* que exploraremos são o uso de *spread* e o cancelamento de cotações. *Spread*, ou margem, consiste em ajustar o P_{esp} para reduzir a discrepância com o P_{exec}. Por exemplo, em uma cotação de compra, buscamos um valor M ideal tal que $P_{\text{esp}} + M \geq P_{\text{exec}}$. Um valor de M muito alto resultaria em um preço não competitivo, enquanto um valor muito baixo manteria o prejuízo. Já o cancelamento de cotações refere-se à possibilidade de o serviço cancelar uma cotação aceita pelo usuário. No entanto, conforme discutido, essa prática não pode exceder 1% das cotações. Para ilustrar melhor o funcionamento do modelo *RFQ*, a Figura 1 apresenta fluxos de compra de 1 BTC/USD utilizando o serviço, considerando tanto o cancelamento de cotações quanto o uso de *spread*.

Tabela 1. Avaliação dos possíveis cenários de *slippage*

	Positivo ($P_{\text{esp}} \geq P_{\text{exec}}$)	Negativo ($P_{\text{esp}} < P_{\text{exec}}$)
Compra	Lucro	Prejuízo
Vende	Prejuízo	Lucro

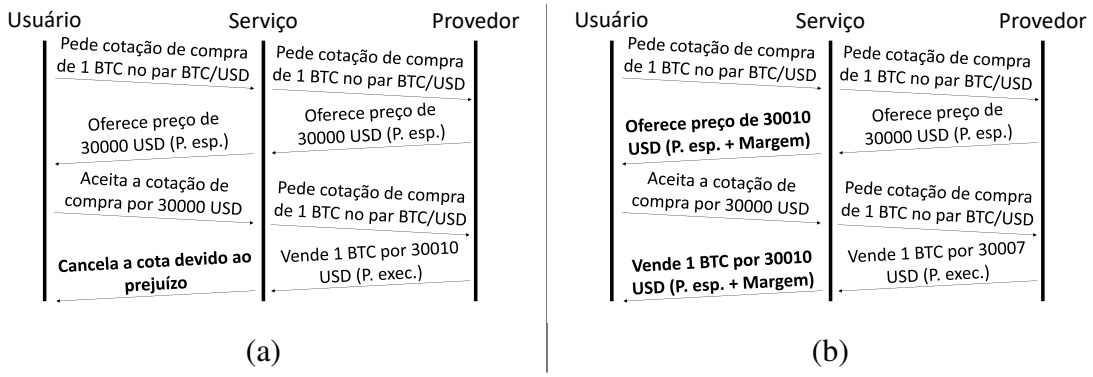


Figura 1. Fluxos do serviço *RFQ*, com duas soluções contra *slippage*: (a) cancelamento de transações desfavoráveis e (b) margem (*spread*) para evitar perdas.

4. Coletando e Processando Dados

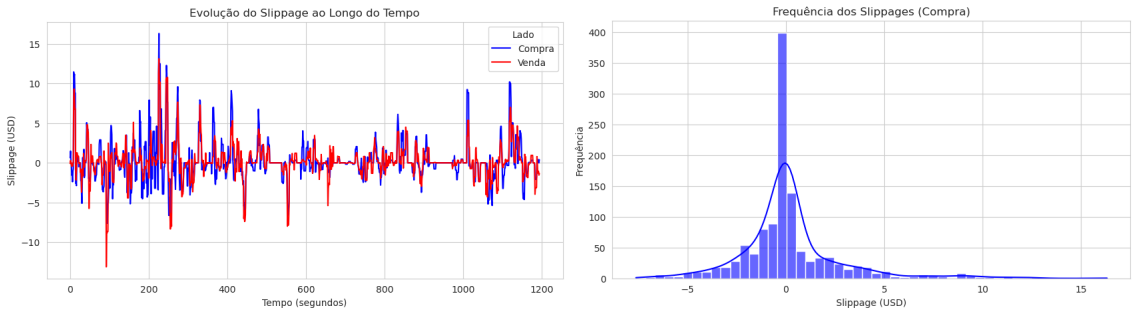
4.1. Escolhendo o provedor

Como mencionado anteriormente, escolhemos a Bitstamp [Bitstamp 2025c] como nossa provedora. A decisão foi baseada na qualidade da documentação e na facilidade de uso da sua *API* [Bitstamp 2025b], que fornece acesso ao livro de ordens atualizado em tempo real via *WebSocket*. Embora *exchanges* como Coinbase e Binance também disponibilizem APIs robustas para consulta de dados de mercado, a API da Bitstamp se destacou pela simplicidade de integração e pela clareza na estrutura das respostas, reduzindo a complexidade da implementação. Além disso, a Bitstamp fornece um *WebApp* [Bitstamp 2025a]

que permite visualizar o comportamento do livro de ordens em tempo real, facilitando a validação dos dados coletados.

4.2. Simulando o serviço

Simularemos o serviço seguindo as restrições definidas anteriormente. Ou seja, geraremos cotações a cada 5 segundos para um usuário que deseja comprar 1 BTC por USD e analisaremos qual seria o *spread* necessário para cobrir um possível prejuízo, caso este ocorra, conforme ilustrado na Tabela 1. Devido à alta volatilidade, que pode ser visualizada no *WebApp* da Bitstamp, assumiremos que, a cada segundo, o *slippage* variará com alta probabilidade. Assim, a cada segundo da amostra, calcularemos a diferença entre o preço naquele segundo e o preço após 5 segundos. Com esse mecanismo, conseguimos calcular o *slippage* para qualquer segundo dentro da amostra. Para ilustrar, a Figura 2 apresenta visualizações dos *slippages* calculados sobre uma amostra ao longo de 20 minutos.



(a) (b)

Figura 2. Visualização dos *slippages* de uma amostra de 20 minutos: (a) Ao longo do tempo e (b) por meio da frequência.

A coleta de dados foi realizada em tempo real, conectando-se ao *WebSocket* da Bitstamp e registrando o livro de ordens atualizado continuamente. As informações foram armazenadas em arquivos JSON, organizados em diferentes tamanhos de amostra, abrangendo períodos que variam de alguns minutos a várias horas, permitindo uma análise mais ampla da variação do mercado. Para processamento e visualização dos dados, foi utilizado um *notebook* Jupyter, que contém tanto a geração dos gráficos apresentados neste estudo quanto as análises realizadas. Embora o *notebook* não esteja estruturado formalmente, ele serve como um rascunho interativo, permitindo a replicação e exploração dos resultados obtidos. O código fonte pode ser encontrado em [Luan Martins 2025b].

5. Calculando a Margem Ideal

5.1. Definindo o intervalo ideal

Dado que conseguimos uma estratégia para calcular os *slippages* de forma consistente, uma primeira abordagem ingênua é utilizar o valor médio desses *slippages* como margem e ignorar completamente a possibilidade de cancelamento de cotações. Para calcular essa média, é necessário definir exatamente o intervalo de tempo sobre o qual ela será computada. Assim como no intervalo de 1 segundo entre os cálculos de *slippages* nos dados coletados, a escolha do intervalo ideal para o cálculo das margens é estatisticamente desafiadora.

Dessa forma, estimamos um intervalo médio com base em observações empíricas. A Figura 2 sugere que a distribuição dos *slippages* se aproxima de uma normal. Essa hipótese é plausível, dado que, apesar das oscilações de curto prazo, estudos indicam que os retornos de ativos financeiros tendem a exibir comportamento aproximadamente estacionário e simétrico no longo prazo (e.g., vide nota de rodapé 2 em [Cont 2001]). No entanto, para avaliar rigorosamente essa hipótese, realizamos testes formais de normalidade sobre os dados, vide Subseção 5.2.

A segunda observação é que, quanto menor o intervalo, mais influenciado pelo passado imediato o *spread* será, tornando-se mais volátil. Por outro lado, quanto maior o intervalo, mais normalizada será a variação da moeda, fazendo com que o *spread* tenda a zero. Essa relação pode ser visualizada nas Figuras 5 e 6, que apresentam distorções significativas, dificultando a modelagem estatística. Assim, um meio-termo adequado, adotado a título de simplificação, é um intervalo de duas horas. Ou seja, para cada instante t da amostra, utilizamos as duas horas anteriores para calcular o *slippage* médio, e a margem será definida por esse valor. Em cenários positivos, há maior lucro para a empresa; em cenários negativos, a margem cobre o prejuízo.

5.2. Analisando a distribuição dos *slippages*

Aplicamos os testes de Kolmogorov-Smirnov (KS-Test), Anderson-Darling e D'Agostino-Pearson, e todos rejeitaram a hipótese nula de que os *slippages* seguem uma distribuição normal, para todas as amostras coletadas. Apesar disso, ao analisarmos a Figura 3, que exibe um histograma comparado a uma curva normal teórica e um *QQ Plot* para uma amostra de duas horas, percebemos que o comportamento dos *slippages* é razoavelmente próximo ao de uma distribuição normal, mesmo que não satisfaça estritamente os critérios estatísticos de normalidade. Dado esse comportamento visualmente semelhante e a tendência dos mercados financeiros a apresentarem padrões estatísticos previsíveis ao longo do tempo, assumiremos que podemos aplicar estatísticas da curva normal sobre a distribuição dos *slippages*. Essa aproximação nos permite utilizar métricas estatísticas familiares, como média e desvio padrão, para modelar a variação dos *slippages* e definir estratégias baseadas nesse comportamento.

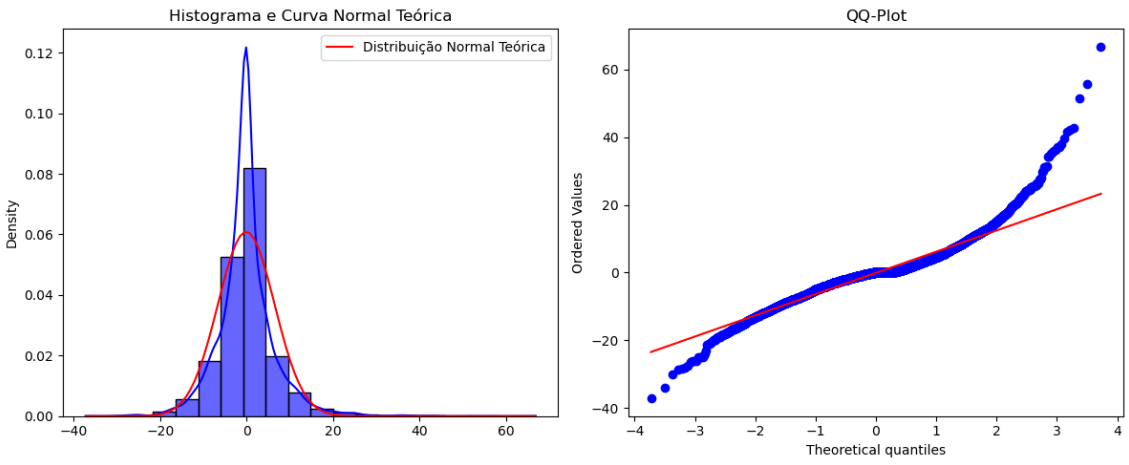


Figura 3. Comparação de uma amostra de 2 horas de *slippages* com a curva normal teórica

Por completude, apesar de seguirmos o estudo assumindo a distribuição normal, realizamos uma investigação sobre qual distribuição estatística melhor se ajusta aos *slippages*. Para isso, testamos diversas distribuições, incluindo Normal, Log-Normal, Cauchy, T-Student, Laplace, Beta, Gumbel, Pareto e Exponencial Generalizada, utilizando o KS-Test para avaliar a aderência de cada uma. Observamos que os resultados variam conforme o tamanho da amostra: amostras pequenas (de minutos) apresentaram melhor ajuste a uma distribuição Cauchy, enquanto amostras maiores (de horas) mostraram um comportamento mais próximo da distribuição T-Student. A Figura 4 ilustra os testes realizados para uma amostra de duas horas, onde a T-Student se mostrou a melhor aproximação.

Esse comportamento pode ser explicado pelas características dessas distribuições. A distribuição Cauchy é conhecida por suas caudas longas, o que significa que ela acomoda melhor eventos extremos, como grandes variações repentinas de preço que ocorrem em pequenas janelas de tempo. Já a distribuição T-Student, embora também possua caudas mais largas do que a normal, converge para uma distribuição normal conforme o tamanho da amostra aumenta. Isso explica por que, ao analisarmos janelas de tempo maiores, os *slippages* começam a se comportar mais como uma T-Student, sugerindo que a volatilidade do mercado tende a suavizar ao longo do tempo.

5.3. Definindo a estratégia final

A abordagem completa considera o cancelamento de cotações. Para isso, precisamos definir um valor que garanta que apenas 1% das cotações sejam canceladas. Como assumimos que os *slippages* seguem uma distribuição normal, podemos calcular a média μ e o desvio padrão σ para modelar sua distribuição estatística. A distribuição normal é uma das mais importantes na estatística, pois modela uma ampla gama de fenômenos naturais, incluindo variações de preços no mercado financeiro. Sua forma característica é a de um sino simétrico, onde a média μ define o centro da distribuição e o desvio padrão σ determina sua dispersão. Em uma distribuição normal padronizada, aproximadamente 68% dos valores estão dentro de 1σ da média, 95% dentro de 2σ e 99% dentro de 2.58σ. Isso significa que, ao escolher $z = 2.58$, garantimos que apenas 1% dos eventos estão além desse limite, tornando essa uma escolha adequada para definir um *spread* que minimize o cancelamento de cotações.

Dessa forma, podemos determinar o valor ótimo do *spread* utilizando a fórmula: $\textit{Spread} = \mu + 2.58 \cdot \sigma$. Esse cálculo assegura que as cotações canceladas permanecerão dentro do limite estipulado de 1%, reduzindo perdas associadas a ajustes excessivos de preços. Finalmente, ao aplicar essa abordagem sobre as amostras coletadas, os resultados foram satisfatórios. Por exemplo, a Figura 7 visualiza o impacto dessa estratégia em termos de lucros e prejuízos para a empresa. Nesse cenário, com 1 BTC sendo comprado a cada segundo, isso totalizaria um volume de centenas de milhões de dólares. Aplicando a heurística, a empresa manteria a taxa de rejeição em 0.75% e lucraria cerca de cem mil dólares, utilizando um *spread* médio de 9.78 dólares.

Além das análises realizadas, desenvolvemos um webapp [Luan Martins 2025a] que permite visualizar, em tempo real, o comportamento do *slippage* médio e simular pedidos de cotação de compra e venda. O código fonte da aplicação pode ser encontrado em [Luan Martins 2025c]. Os dados ao vivo da Bitstamp são consumidos para calcular a média e o desvio padrão dos *slippages*, possibilitando a estimativa do *spread* que seria

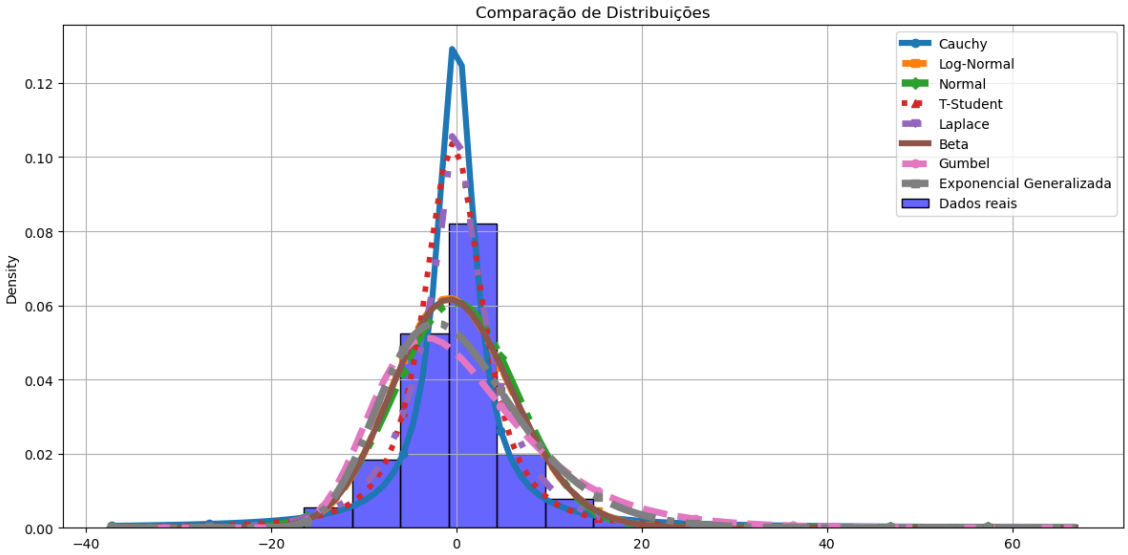


Figura 4. Testes de aderência de diferentes distribuições com uma amostra de 2 horas de slippages

aplicado pela nossa solução conforme descrito anteriormente. A Seção 6 aprofunda sobre a construção do *WebApp*.

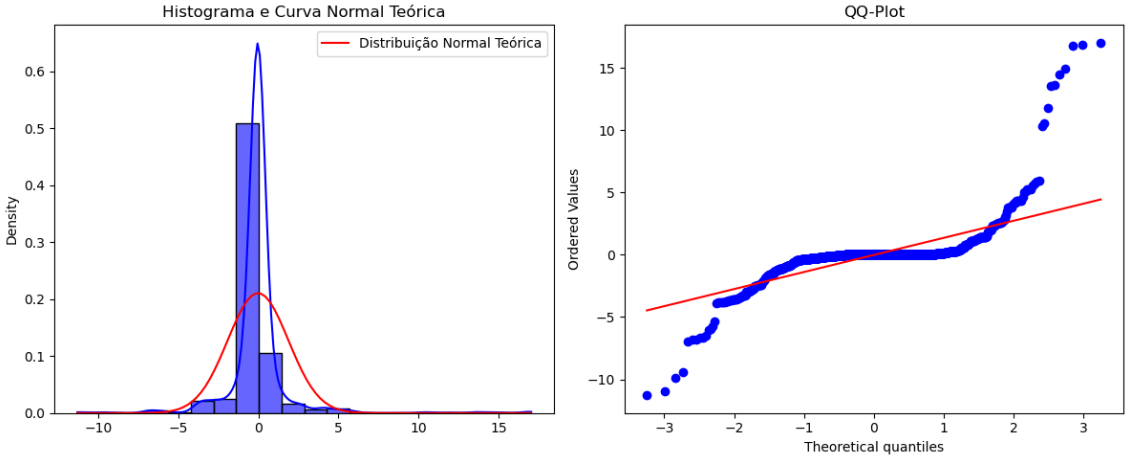


Figura 5. Comparação de uma amostra de 20 minutos de *slippages* com a curva normal teórica

6. Implementando o *WebApp*

Para complementar a análise estatística apresentada neste estudo, desenvolvemos um *WebApp* que permite visualizar em tempo real o comportamento do *slippage* e simular pedidos de cotação de compra e venda. A aplicação serve como uma ferramenta prática para demonstrar os conceitos discutidos, permitindo que usuários interajam diretamente com os dados da Bitstamp e observem o impacto da estratégia de mitigação de *slippage*. Além de calcular a média e o desvio padrão dos *slippages* com base na janela de tempo em que a aplicação permanece aberta, o WebApp também estima o *spread* aplicado de acordo com a heurística proposta, tornando o processo mais intuitivo e acessível.

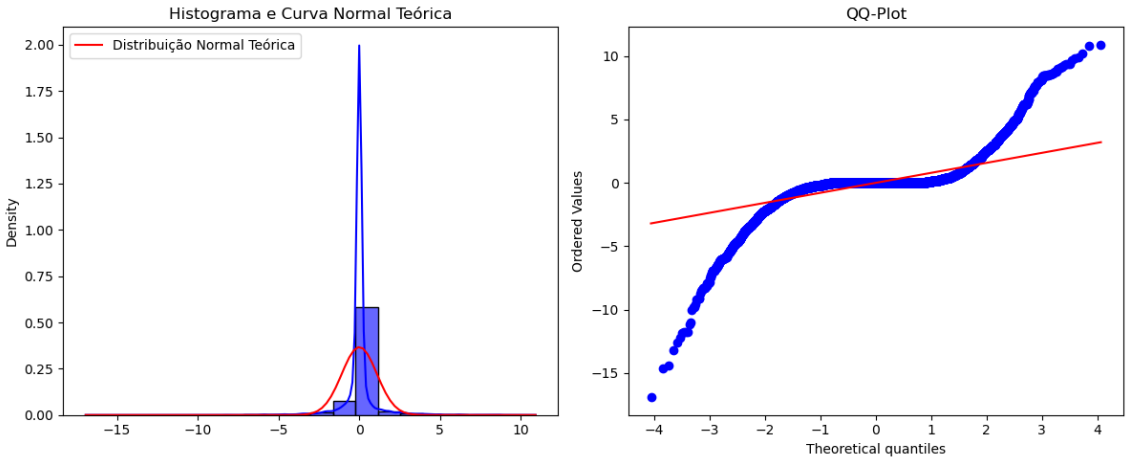


Figura 6. Comparação de uma amostra de 8 horas de *slippages* com a curva normal teórica

O *WebApp* foi desenvolvido como uma aplicação simples baseada em HTML, CSS e JavaScript, garantindo leveza e acessibilidade. O *frontend* é responsável por exibir as informações em tempo real, utilizando JavaScript para consumir os dados da *API* da Bitstamp e atualizar dinamicamente os valores de *slippage* e *spread*. Como não há necessidade de um *backend* dedicado, a aplicação é hospedada no GitHub Pages, permitindo fácil acesso sem configuração de servidores.

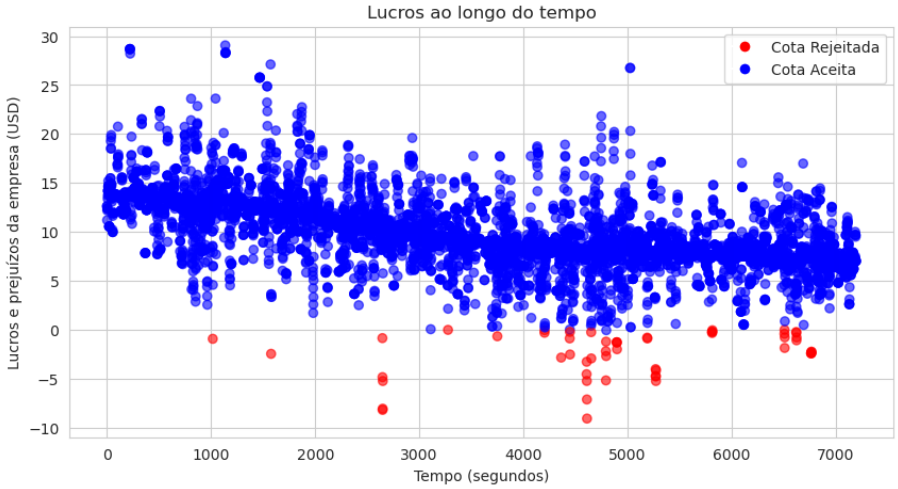


Figura 7. Lucros e prejuízos da empresa ao longo de 2 horas de uma amostra

Vamos descrever a estrutura do *WebApp*, visualizada na Figura 8. No topo, exibe-se a data da última atualização do livro de ordens da Bitstamp, garantindo que os dados estejam sempre sincronizados com o mercado. Abaixo, a interface é dividida em duas colunas: à esquerda, informações relacionadas às ordens de compra de BTC, e à direita, informações sobre as ordens de venda. Cada coluna começa com uma lista das 10 melhores ofertas do livro de ordens naquele momento. Em seguida, cada coluna exibe o preço atual para comprar ou vender 1 BTC, calculado dinamicamente com base no livro de ordens. Vale notar que a compra é realizada a partir das ordens de venda, enquanto a venda ocorre para as ordens de compra. Logo abaixo, são apresentados os preços médios para

compra e venda, considerando todas as cotações registradas enquanto a aplicação esteve aberta. Além disso, os *slippages* são calculados com base nesses preços armazenados em memória, permitindo acompanhar a variação do mercado em tempo real. Utilizando esses valores, o *spread* aplicado segue a heurística proposta, sendo calculado como a média somada a 2.58 vezes o desvio padrão, conforme discutido anteriormente. Na sequência, botões permitem simular cotações de compra e venda, aplicando o modelo desenvolvido. Por fim, no rodapé de cada coluna, uma lista exibe o histórico de preços de compra e venda do BTC armazenados em memória, registrando os valores a cada segundo desde a abertura da aplicação.

Figura 8. Captura de tela do Webapp

Para visualizar uma simulação de cotação de compra realizada no *WebApp*, utilizaremos um exemplo: Às 13:31:05, foi feito o pedido de cotação, exibindo o preço esperado para a compra de 1 BTC naquele momento: 87004.54. Simultaneamente, foi calculado o *spread* ideal para essa transação, resultando em um valor de 23.79. Cinco segundos depois, às 13:31:10, o preço executado foi registrado em 87012.43, resultando

em um *slippage* de -7.89, ou seja, um prejuízo para o serviço caso não houvesse ajuste de margem. No entanto, ao aplicar o *spread* calculado, o *slippage* ajustado passou a ser 15.90, garantindo lucro na operação. Vale destacar que o valor da margem aplicada, equivalente a apenas 0.027% do preço da transação, é insignificante do ponto de vista da competitividade, mas suficiente para mitigar o risco de perdas e estabilizar os resultados do serviço.

As Listings 1 e 2 ilustram trechos-chave do código do *WebApp*, responsáveis pelo consumo dos dados da Bitstamp e pelo cálculo do *spread*. A Listing 1 mostra a inicialização da conexão com a *API WebSocket* da Bitstamp, onde a aplicação assina o canal `order_book_btcusd` para receber atualizações em tempo real do livro de ordens. Já a Listing 2 apresenta a lógica para calcular o *spread* com base nos *slippages* registrados. Para isso, é definida uma janela de tempo de 5 segundos (`TIME_WINDOW`), sobre a qual os *slippages* são calculados e armazenados em um *array*. Com esses valores, são computadas a média e o desvio padrão, e o *spread* final é determinado aplicando a heurística baseada na tabela z, com $z = 2.58$ para cobrir 99% dos casos. Esses trechos de código ilustram a implementação prática da metodologia discutida ao longo do artigo, e o código completo pode ser encontrado em [Luan Martins 2025c].

```
const bitstamp = new WebSocket("wss://ws.bitstamp.net/");

bitstamp.onopen = () => {
  const data = {
    event: "bts:subscribe",
    data: {
      channel: "order_book_btcusd",
    },
  };

  bitstamp.send(JSON.stringify(data));
};
```

Listing 1. Inscrevendo no canal *WebSocket* da Bitstamp

```
const TIME_WINDOW = 5;
const slippages = [];
for (let i = 0; i < prices.length - TIME_WINDOW; i++) {
const slippage = prices[i] - prices[i + TIME_WINDOW];
slippages.push(slippage);
}

const slippageAverage = average(slippages);
const slippageVariance = variance(slippages);
const slippageSd = Math.sqrt(slippageVariance);
const z = 2.58;
const spread = slippageAverage + z * slippageSd;
```

Listing 2. Calculando as estatísticas e o *spread*

7. Conclusão

Por meio de uma extensa coleta e processamento de dados, além da aplicação de diversas heurísticas simples e razoáveis, conseguimos estabelecer uma prova de conceito capaz de determinar, em tempo real, um *spread* razoável para uma cotação, respeitando todas as restrições estabelecidas: A escolha do par BTC/USD, da corretora Bitstamp, da janela de 5 segundos e, principalmente, das premissas baseadas em observações, como a segmentação das amostras e a suposição de que os *slippages* seguem uma distribuição normal. Apesar das simplificações, o estudo atingiu seu objetivo ao propor uma heurística simples e razoável, validada com dados reais.

Trabalhos futuros podem explorar três principais vertentes: Primeiro, utilizar algoritmos ou técnicas como aprendizado de máquina para determinar o intervalo de tempo mais adequado para o cálculo do *slippage* médio. Segundo, muitos dos parâmetros fixados devem ser flexibilizados. Entre os aspectos a serem explorados estão: testar diferentes pares de moedas, coletar dados de diferentes corretoras e avaliar o impacto de distintas janelas de tempo nas cotações. Terceiro, integrar a solução ao serviço, permitindo o cálculo dinâmico do *spread* ideal em tempo real, e avaliar os resultados.

Agradecimentos

Este projeto foi parcialmente financiado pela FAPERJ (E-26/204.268/2024), CNPq e CAPES.

Referências

Aldridge, I. (2022). Slippage in AMM markets. *SSRN*.

Almgren, R. and Chriss, N. (2000). Optimal execution of portfolio transactions. *Journal of Risk*, 3(2):5–40.

Aoyagi, Y. and Ibuka, N. (2021). Simulation-based estimation of execution cost in crypto market using monte carlo methods. *arXiv preprint arXiv:2109.12549*.

Bitstamp (2025a). Bitstamp live order book. `https://www.bitstamp.net/s/webapp/examples/order_book_v2.html`. [Online; accessed 20-March-2025].

Bitstamp (2025b). Bitstamp public api (v2). `https://www.bitstamp.net/api/`. [Online; accessed 20-March-2025].

Bitstamp (2025c). Buy & trade with ease on the trusted crypto exchange. `https://www.bitstamp.net/`. [Online; accessed 20-March-2025].

Bouchaud, J.-P., Bonart, J., Donier, J., and Gould, M. (2018). *Trades, Quotes and Prices: Financial Markets Under the Microscope*. Cambridge University Press.

Cartea, Á., Jaimungal, S., and Penalva, J. (2015). *Algorithmic and High-Frequency Trading*. Cambridge University Press.

Coinbase (2022). Machine learning at coinbase. [Online; acessado em abril de 2025].

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative finance*, 1(2):223.

Dixon, M., Halperin, I., and Bilokon, P. (2020). *Machine Learning in Finance: From Theory to Practice*. Springer.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007.

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., and Howison, S. D. (2013). Limit order books. *Quantitative Finance*, 13(11):1709–1742.

Kaiko (2023). Execution strategies and slippage in crypto markets. [Online; acessado em abril de 2025].

Lim, T. (2022). Predictive crypto-asset automated market making architecture for decentralized finance using deep reinforcement learning. *arXiv*, (2211.01346).

Luan Martins (2025a). Mitigando slippage. `https://muanlartins.github.io/mitigating-slippage`. [Online; accessed 26-March-2025].

Luan Martins (2025b). Repositório do notebook jupyter. `https://muanlartins.github.io/slippage-problem`. [Online; accessed 26-March-2025].

Luan Martins (2025c). Repositório do webapp mitigando slippage. `https://github.com/muanlartins/mitigating-slippage`. [Online; accessed 26-March-2025].

Martins, L. (2023). Entendendo e evitando o slippage problem na web3.0. *Pantheon*.

Nevmyvaka, Y., Feng, Y., and Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680.

Parfin (2025). Empowering you with full life cycle digital asset solutions to shape the future of finance. `https://parfin.io/`. [Online; accessed 20-March-2025].

Research, P. (2023). Estimating market impact in crypto markets. [Online; acessado em abril de 2025].

Shah, J., Javare, P., and Khetan, D. (2020). Slipswap: Reduce the slippage that is incurred during the swap of tokens using algorithmic analysis. *IEEE*, pages 131–134.

Spooner, T., Savani, R., and Megson, S. (2018). Market making via reinforcement learning. *arXiv preprint arXiv:1804.04216*.

Street, J. (2023). Latency arbitrage and quote protection. [Online; acessado em abril de 2025].

Trading, R. (2023). Rl for mm strategies. [Online; acessado em abril de 2025].

Wintermute (2023). Crypto liquidity 101. [Online; acessado em abril de 2025].