

Overcoming the Stablecoin Trilemma: Designing and Simulating a Decentralized, Stable, and Capital-Efficient Peg Maintenance System

Mikhail Kartavchenko¹, Maksim Izmaylov², Alexander Kartavchenko³, Nikita Svirskii⁴, Fefor Zaharov¹, Victoria Artemenko⁵, Evelina Illarionova¹, Lyubov Kolotushkina¹, Semen Krivonosov¹

¹Governor's Physics and Mathematics Lyceum No. 30, Capestone, Saint-Petersburg, Russia

²Peter the Great St. Petersburg Polytechnic University, Saint-Petersburg, Russia

³Toptal LLC, Saint-Petersburg, Russia

⁴Dar es Salam American School, Capestone, Rabat, Morocco

⁵Gazprom School, Capestone, Saint-Petersburg, Russia

Email: michaelkar@capestone.club, izmajlov_mk@spbstu.ru, alekart30@gmail.com, nikitasvr@capestone.club, fdrzah@capestone.club, victoriaart@capestone.club, evelinail@capestone.club, kolotlyb@capestone.club, semkriv@capestone.club

How to cite this paper: Kartavchenko, M., Izmaylov, M., Kartavchenko, A., Svirskii, N., Zaharov, F., Artemenko, V., Illarionova, E., Kolotushkina, L., & Krivonosov, S. (2024). Overcoming the Stablecoin Trilemma: Designing and Simulating a Decentralized, Stable, and Capital-Efficient Peg Maintenance System. *Open Journal of Business and Management*, 12, 4413-4440.

<https://doi.org/10.4236/ojbm.2024.126222>

Received: September 29, 2024

Accepted: November 25, 2024

Published: November 28, 2024

Copyright © 2024 by author(s) and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution-NonCommercial International License (CC BY-NC 4.0).

<http://creativecommons.org/licenses/by-nc/4.0/>



Open Access

Abstract

The Stablecoin Trilemma—balancing stability, decentralization, and capital efficiency—remains a significant challenge in the decentralized finance (DeFi) landscape. This study introduces an innovative stablecoin system, USC, designed to overcome the inherent limitations of existing models by simultaneously achieving stability, decentralization, and capital efficiency. The USC system leverages a combination of decentralized assets, including the hypothetical native token CAPE and ETH, integrated through key components such as the Treasury, Treasury Credit Module, CAPE Bonder, and Merchant. A comprehensive Python-based simulation spanning 298 days was conducted to evaluate the system's performance under varying market conditions. ETH price movements were meticulously modeled using historical data from the Binance/USDT trading pair, with specific periods of daily growth and decline to test the system's resilience. Additionally, CAPE price determination was simulated using a hypothetical liquidity pool. The simulation results demonstrated that the USC system effectively maintained the usc_support metric above the critical threshold of 1 throughout the study period, indicating that each USC remained fully backed by the underlying assets. The Treasury Credit Module played a crucial role by offering a stable 10% Annual Percentage Rate (APR), incentivizing participants to lock their USC and thereby reducing the circulating supply during asset depreciation periods. Furthermore, the Merchant component

provided a substantial safety net with a borrowing capacity of up to 67,315 ETH (approximately \$160.6 million at the simulation's final ETH price), which is nearly three and a half times the Treasury's holdings. This extensive liquidity provision enhanced the system's ability to maintain stability during significant market downturns. However, the study acknowledges several key limitations, including the lack of realistic participant behavior modeling, the absence of external shocks such as regulatory changes or technological failures, and the lack of empirical validation through real-world testing. These limitations highlight the need for further research and development to address behavioral complexities, incorporate unforeseen variables, and validate the system's performance in live environments. In conclusion, the USC stablecoin system presents a promising framework for resolving the Stablecoin Trilemma by integrating decentralized assets and innovative financial mechanisms. While the simulation results are encouraging, demonstrating robust stability, the system's practical implementation will require overcoming significant technical, regulatory, and operational challenges.

Keywords

Stablecoin Trilemma, Decentralization, Capital Efficiency, Stability, USC Stablecoin, Economic Simulation, Peg Maintenance System

1. Introduction

1.1. Problem Identification

The concept of stablecoins has revolutionized the world of cryptocurrency by offering a digital currency that strives to maintain a stable value. However, achieving stability while also ensuring decentralization and capital efficiency has proven to be a significant challenge. This issue is encapsulated in what is known as the Stablecoin Trilemma—a problem that forces stablecoins to compromise between these three essential qualities. As a result, existing stablecoins often sacrifice one of these qualities:

- Centralization: Popular stablecoins like USDC and USDT maintain stability but rely on centralized fiat reserves, sacrificing decentralization.
- Capital Inefficiency: Stablecoins like DAI use over-collateralized crypto loans, ensuring decentralization and stability but locking up excessive capital.
- Instability: Algorithmic stablecoins like Terra's UST attempt to maintain stability through supply adjustments without full collateral backing, leading to potential instability.

1.2. Why It's Important to Solve

Failing to address the Trilemma limits the potential of stablecoins and DeFi. Stablecoins that rely on centralized backing are susceptible to regulatory risks, transparency issues, and potential central points of failure. Systems that use over-collateralized loans are capital inefficient, hindering liquidity and widespread

adoption. Algorithmic stablecoins present significant risks due to their dependence on self-correcting mechanisms that can fail under extreme market conditions, causing instability and loss of investor confidence. Solving the Stablecoin Trilemma is crucial for the future of DeFi. A stablecoin that is stable, decentralized, and capital efficient can lead to wider adoption, greater liquidity, and restlessness in decentralized systems, paving the way for a truly decentralized financial ecosystem (Adams & Van Den Bergh, 2020; Allen & Wysocki, 2019).

1.3. Goal of the Study

The primary goal of this study is to solve the Stablecoin Trilemma by designing a stablecoin system that simultaneously achieves stability, decentralization, and capital efficiency without compromise. By introducing the innovative USC stablecoin system—which integrates decentralized assets, economic incentives, and inherent stability mechanisms—we aim to provide a practical solution that addresses the inherent trade-offs faced by existing stablecoins. Successfully overcoming the Trilemma would not only validate the proposed system but also offer a blueprint for the next generation of stablecoins, fostering sustainable growth in decentralized finance ecosystems (Puschmann, 2017).

1.4. The Stablecoin Trilemma Explored

Defining Stability, Decentralization, and Capital Efficiency

- **Stability:** The ability of a stablecoin to maintain a consistent value, usually pegged to a fiat currency like the US dollar, minimizing volatility and serving as a reliable medium of exchange and store of value.
- **Decentralization:** Operating without reliance on centralized authorities or intermediaries, ensuring that control and governance are distributed among a network of participants.
- **Capital Efficiency:** The effective use of capital where minimal assets are required to achieve a desired level of stability, avoiding the need for over-collateralization that locks up excessive funds (Stablecoins.wtf, n.d.).

1.5. Challenges in Achieving All Three

The Stablecoin Trilemma posits that existing stablecoins can typically achieve only two of the three qualities:

- **Stable and Capital Efficient but Not Decentralized:** Stablecoins like USDC and USDT maintain stability and are capital efficient by being fully backed by fiat reserves. However, they rely on centralized entities, compromising decentralization.
- **Stable and Decentralized but Not Capital Efficient:** Decentralized stablecoins like DAI achieve stability without central authorities but require over-collateralization, leading to capital inefficiency.
- **Decentralized and Capital Efficient but Not Stable:** Algorithmic stablecoins attempt to maintain a peg through supply adjustments without full collateral backing. While they are decentralized and capital efficient, they often fail to

maintain long-term stability, as evidenced by the collapse of Terra's UST.

2. Methods

2.1. Approach to Solving the Problem

To overcome the Stablecoin Trilemma, the USC system introduces a novel approach that balances decentralization, capital efficiency, and stability through a set of interconnected components.

2.1.1. Decentralization

No fiat money or centralized stablecoins are used to back the stablecoin's value, ensuring a purely decentralized system. Instead, USC is backed by decentralized crypto assets: a mix of CAPE (a hypothetical native token of a hypothetical purpose-built blockchain) and Ethereum (ETH). This ensures that control over the system is fully decentralized, with no reliance on centralized financial institutions or reserves.

2.1.2. Capital Efficiency

The system achieves capital efficiency by allowing users to directly exchange assets with the Treasury at a 1:1 exchange rate—issuing 1 USC for every \$1 worth of the specified mix of CAPE and ETH provided. This eliminates the need for over-collateralized loans typical in other stablecoins like DAI. Furthermore, the system creates an artificial buffer zone—an over-collateralization achieved internally rather than by individual users. This buffer is established through:

- Deposits into the Treasury Credit Module: Participants lock their USC to earn interest, reducing the circulating supply and enhancing the system's collateralization.
- Loans from the Merchant: The Merchant can inject additional capital into the system by providing loans backed by its staked ETH reserves, stabilizing USC during market volatility.

By combining the 1:1 exchange rate with these internal mechanisms, the USC system maintains capital efficiency for users while ensuring sufficient over-collateralization exists—aiming for a USC backing level of 1.5 (Catalini & Gans, 2016).

2.1.3. Stability

A key focus is stability, achieved through the creation of a buffer zone in the stablecoin's backing value. This buffer helps ensure that each USC remains backed even during significant downturns in assets backing the coin. Additionally, instant liquidity injections through loans provided by the Merchant offer immediate support to stabilize the system during high volatility, making the system resilient to market shocks.

2.2. System Components

2.2.1. Treasury Module

The Treasury is responsible for issuing and redeeming USC. Users exchange a mix of 60% CAPE and 40% ETH with the Treasury to receive USC. The exchange rate is 1 USC for every \$1 worth of the mix, with small fees applied.

2.2.2. Treasury Credit Module

The Treasury Credit Module allows participants to lock their USC for 90 days in exchange for an APR of up to 10%. This helps reduce the circulating supply of USC, alleviating pressure on the Treasury to back every stablecoin. The rewards pool that funds this APR comes from two sources: the CAPE Bonder and profits from Merchants. The module plays a critical role in maintaining the stability of the system by creating a buffer zone in backing value.

2.2.3. CAPE Bonder

The CAPE Bonder enables participants to bond CAPE at a 5% discount in exchange for a mix of ETH and CAPE (in the same proportion as in the Treasury). After a lock period of five days, participants can sell the CAPE for profit or loss depending on its market price. The CAPE and ETH provided by participants are then exchanged for USC in the Treasury, and the resulting USC is transferred to the Treasury Credit Module's rewards pool to support the APR ([Olympus Dao Docs](#)).

2.2.4. Merchant

The Merchant's primary function is to facilitate the flow of ETH from external blockchains into the internal system by issuing Synthetic ETH on the internal blockchain. Users deposit their ETH into the Merchant from an external blockchain like Ethereum, and in return, the Merchant mints Synthetic ETH on the internal blockchain. While Synthetic ETH operates within the system, the Merchant manages the "real" ETH externally through reserves and staking.

2.3. Functions of the Merchant

- **ETH Management:** Upon receiving external ETH, the Merchant allocates 30% of it to a reserve pool and stakes the remaining 70%. Staking the ETH generates daily interest, which is managed in two ways: it is both reinvested in real ETH to grow the reserves and the staked pool, and allocated to the Treasury Credit Module's rewards pool in the form of Synthetic ETH (Merchant treats ETH that is received as a profit from staking the same way as it was ETH that was brought by users).
- **Lending to the Treasury:** When additional backing is required for USC, the Merchant plays a pivotal role in stabilizing the system by taking an over-collateralized loan using its staked ETH as collateral. The Merchant can borrow up to 30% of its staked ETH pool. The acquired ETH flows through the Merchant before being sent to the Treasury in the form of Synthetic ETH, thus increasing both the reserves and staking pool in real ETH.

Why the Merchant Is Essential

The Merchant provides an additional layer of defense against volatility by managing real-world ETH assets outside the internal blockchain. Its ability to stake ETH not only generates returns but also bolsters the system's stability during market fluctuations. By reinvesting staking rewards and contributing to the Treasury's

backing, the Merchant enhances the system's resilience and capacity to maintain the USC peg.

2.4. The Role and Nature of CAPE in the USC System

2.4.1. CAPE: A Hypothetical Native Token

CAPE is introduced as a hypothetical native token specific to a hypothetical purpose-built blockchain designed exclusively for the USC system. It is important to note that CAPE and the associated blockchain are conceptual and have not been implemented in reality for this study. The recommendation to build the USC stablecoin on a separate blockchain stems from the necessity to incorporate unique features and mechanics that may not be feasible on existing platforms.

2.4.2. Issuance and Governance of CAPE

CAPE is envisioned to be issued through a transparent and decentralized protocol on the blockchain. Its supply would be governed by predefined algorithms or community-driven governance models, ensuring no central authority controls its issuance, aligning with the system's goal of maintaining decentralization (Wood, 2014; Schär, 2021).

2.4.3. Considerations for CAPE Issuance

It is crucial to carefully design the issuance mechanisms of CAPE to maintain its price stability, which is vital for the overall stability of the USC system. The CAPE Bonder module already introduces new CAPE into circulation by offering it at a discount to participants. While this incentivizes engagement and supports the Treasury Credit Module's rewards pool, it can exert downward pressure on CAPE's market price if not properly managed. Excessive issuance of CAPE, whether through the CAPE Bonder or other mechanisms, could lead to an oversupply, negatively impacting its price. Since CAPE is a significant part of USC's backing assets, a decline in CAPE's value could weaken the backing value and threaten the stability of USC itself.

2.4.4. Utility of CAPE beyond Backing USC

- Beyond serving as one of the assets backing USC, CAPE would have utility within the blockchain's DeFi ecosystem: Governance Participation: CAPE holders might have voting rights on protocol upgrades and governance decisions, fostering community involvement.
- Access to DeFi Services: CAPE could be used within various DeFi applications on the blockchain, such as lending, borrowing, and liquidity provision.

2.5. Necessity of a Purpose-Built Blockchain for the USC System

- Centralized Management of Real ETH Assets: By requiring users to interact with the system through the Merchant, the system ensures that all real ETH assets are managed by the system itself, allowing for effective staking, borrowing, and liquidity provision.
- Advantages of a Purpose-Built Blockchain for CAPE Demand: By creating a separate, purpose-built blockchain with CAPE as its native token, the USC

system can significantly enhance CAPE's demand. CAPE would be required for every operation within the blockchain, ensuring consistent and widespread use. This intrinsic demand supports CAPE's price stability and reduces the risk of price volatility affecting the stablecoin's backing.

2.6. Simulation Methodology and Assumptions

2.6.1. Simulation Framework

A Python-based simulation was conducted over a 298-day period to evaluate how effectively the system could maintain stability under varying market conditions. The simulation included three primary capital flows into different parts of the system, processed daily:

- **ETH to CAPE Bonder (eth_to_bonder = 20)** This flow represents the amount of ETH that participants allocate daily to purchase CAPE bonds through the CAPE Bonder module.
- **ETH to Treasury Credit Module (eth_to_bank = 100)** This flow represents the daily amount of ETH that participants deposit into the Treasury Credit Module, effectively converting their ETH into USC and locking it for a specified period to earn interest.
- **ETH for External Purposes (eth_for_third_flow = 100)** This flow accounts for the daily amount of ETH that participants convert into USC for purposes outside of the system's internal modules, such as trading, sending, storing, or interacting with decentralized applications (dApps) within the blockchain's ecosystem or outside of it.

2.6.2. Price Determination of ETH and CAPE

ETH price data was based on historical data from the Binance/USDT trading pair. The ETH price was recalculated using specific formulas that estimate the average growth or decline over predefined time periods to simulate market conditions and challenge the system's stability mechanisms. However, it is important to acknowledge that the ETH price data generated using our formulas may contain a degree of error due to the simplifications in our modeling approach, which could impact the accuracy and precision of the simulation results (**Figure 1**).

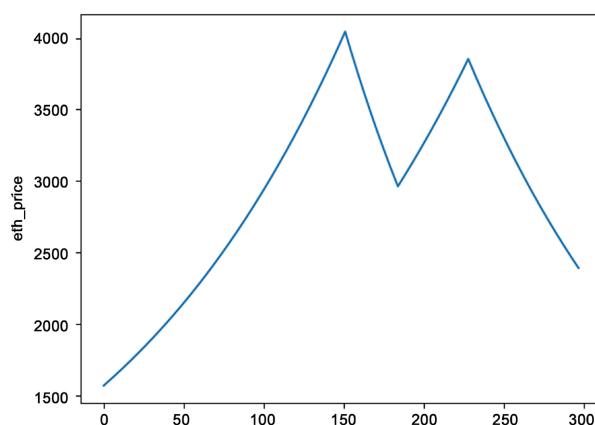


Figure 1. Simulated ETH price dynamics.

Specifically, ETH was modeled to:

- Increase in price by 0.63% daily from $t = 1$ to $t = 151$ (October 12th, 2023 to March 11th, 2024)
- Decrease in price by 0.94% daily from $t = 152$ to $t = 184$ (March 12th, 2024 to April 13th, 2024)
- Increase in price by 0.60% daily from $t = 185$ to $t = 228$ (April 13th, 2024 to May 27th, 2024)
- Decrease in price by 0.69% daily from $t = 229$ to $t = 298$ (May 28th, 2024 to August 5th, 2024)

For CAPE price determination, the simulation utilized a hypothetical liquidity pool with ample initial assets, ensuring that CAPE's price was accurately and smoothly determined within the model. In a real-world scenario, price oracles would likely be employed to obtain reliable CAPE and ETH price data. However, for the purposes of this simulation, using a hypothetical liquidity pool was the most straightforward method to achieve consistent and realistic price dynamics.

2.6.3. Assumptions Regarding Participant Behavior

- Participants are assumed to sell their bonded CAPE immediately upon the expiration of the lock period, regardless of the prevailing CAPE price. Additionally, participants are expected to deposit into the Treasury Credit Module whenever feasible.
- Consistent Participation: The simulation presumes ongoing engagement from participants, with fixed capital inflows reflecting sustained interest in the system (explanation of this assumption is provided in the section below).

2.6.4. Flows of ETH into the System Are Fixed and Always Positive

The simulation assumes that demand for deposits into the Treasury Credit Module remains consistently positive under most market conditions. This assumption is based on the system offering a 10% APR, which is higher than most traditional finance alternatives, ensuring that participants continue depositing even during market downturns. During periods of market decline, demand for these deposits is expected to be particularly strong, as USC is designed to remain stable, making it an attractive, non-volatile alternative when other cryptocurrencies and their projects' APRs are falling. Conversely, when the market is growing and alternative projects may offer higher APRs, USC's stability—designed to mimic the value of the dollar—still appeals to participants who seek security over speculative returns. The simulation also considers that testing for a bank run scenario is unnecessary, as nearly any financial system would struggle under such conditions. However, the USC system is designed with a buffer zone, indicated by the usc_support metric, which provides protection against up to a 25% drop in both ETH and CAPE prices without triggering panic. If USC maintains its \$1 peg at all times, there is no rational incentive for participants to initiate a bank run. In the rare event that the usc_support metric drops below 1, Merchants would promptly recover the system's stability by issuing loans, and the high demand for deposits in the

Treasury Credit Module would continue to restore the system's balance. Thus, the assumption is that the system is resilient against typical market fluctuations, and panic-induced bank runs are highly unlikely in the simulated environment.

2.6.5. No Smart Contract Failures or External Shocks

The simulation assumes that all smart contracts function without any technical failures or exploits. Furthermore, it assumes that no external shocks, such as regulatory changes or large-scale market disruptions beyond ETH price fluctuations, impact the system's operations.

2.6.6. Assumption Regarding Market Volatility

The simulation did not incorporate scenarios involving extreme ETH price downturns. This decision was based on evidence from studies indicating a trend of decreasing volatility in cryptocurrency markets over recent years. As such, the simulation focused on replicating recent ETH price movements, which are less volatile than those observed in the earlier history of cryptocurrencies. By modeling price fluctuations that reflect current market conditions, the study aims to provide a realistic assessment of the USC system's performance under typical volatility levels. This assumption acknowledges that while extreme market crashes are possible, they are less representative of the current market environment and were thus excluded from the simulation's scope ([Fidelity Digital Assets, 2022](#)).

3. Results

3.1. USC Support Metric

The usc_support metric varied over the course of the simulation. Initial values exceeded the system's desired level of 1.5 due to an increase in the price of ETH. The metric subsequently dropped below 1.5 during periods of ETH price decline but remained above the critical threshold of 1, indicating full backing of USC (see [Figure 2](#)).

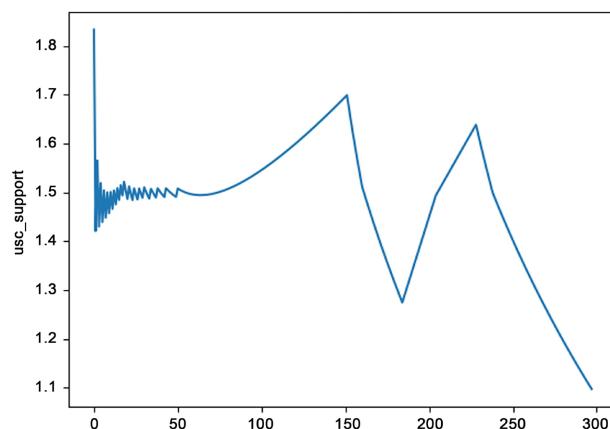
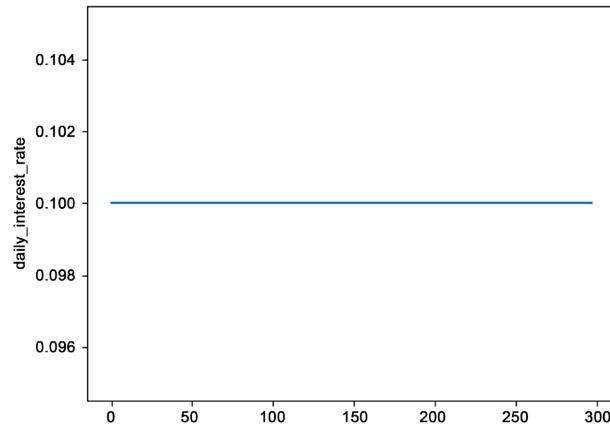


Figure 2. USC support metric fluctuations.

3.2. Treasury Credit Module Interest Rate

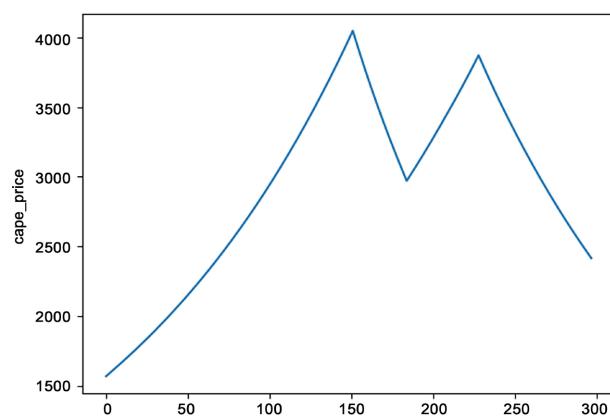
The interest rate within the Treasury Credit Module remained stable at 10% APR for the duration of the simulation. No deviations from this rate were observed throughout the simulation period (see [Figure 3](#)).



[Figure 3](#). Treasury credit module APR stability.

3.3. Merchant and Treasury Holdings at the End of the Simulation

- The Merchant held 320551.43 ETH, with the price of ETH at \$2389.74.
- The Merchant earned 20644.76 ETH through staking operations during the simulation period.
- The Treasury Credit Module had 17410324.95 USC available for withdrawal at any given moment, representing USC from deposits that had completed the 90-day lock period.
- The Treasury held 19102.17 ETH and 28525.50 CAPE. The final price of CAPE was \$2414.78 (see [Figure 4](#) for the CAPE/USD price graph).



[Figure 4](#). CAPE/USD price trends.

3.4. Borrowing Capacity from the Merchant

Based on the Merchant's holdings at the conclusion of the simulation, the Treasury could borrow up to 67,315 ETH if the usc_support metric fell below 1.

3.5. USC Held by Users for External Purposes

A total of 86923148.87 USC was held by participants outside the stablecoin system, indicating the volume of USC being used for external purposes such as trading or other activities.

4. Discussion

4.1. Interpretation of Results

The simulation results demonstrate that the proposed USC stablecoin system effectively addresses the Stablecoin Trilemma by achieving stability, decentralization, and capital efficiency simultaneously. Throughout the 298-day simulation period, the usc_support metric fluctuated but consistently remained above the critical threshold of 1. This indicates that at all times during the simulation, each USC was fully backed by the underlying assets (ETH and CAPE). The maintenance of the usc_support metric above 1 demonstrates the system's ability to withstand market downturns without compromising the full backing of USC. This consistent full backing is essential for maintaining user trust and preventing scenarios like bank runs, where a loss of confidence could lead to mass redemptions and destabilize the system. The Treasury Credit Module played a vital role in this stability by offering a stable interest rate of 10% APR. This reduced the circulating supply of USC during periods when backing assets were decreasing in value, helping to maintain the usc_support metric above 1. Moreover, the system did not need to activate emergency measures such as borrowing from the Merchant, indicating that the internal mechanisms and buffer zones were sufficient to maintain stability. However, the Merchant's potential capacity to provide additional liquidity represents a significant safety net, enhancing the system's resilience. Specifically, the Treasury could borrow up to 67,315 ETH, equivalent to approximately \$160.6 million at the simulation's final ETH price of \$2389.74. In comparison, the Treasury's holdings at the end of the simulation totaled 19102.17 ETH, valued at around \$45.6 million. This means that the borrowing capacity from the Merchant is nearly three and a half times the Treasury's current ETH holdings, providing a robust cushion to maintain the USC stablecoin's backing and stability in the event of significant market downturns.

4.2. Key Limitations of the System

The proposed USC stablecoin system, while innovative, presents several key limitations that must be addressed to ensure its successful implementation and long-term viability. These limitations span technical, regulatory, economic, and operational aspects, each posing unique challenges:

1) Technical Implementation Challenges: Developing the USC system involves creating complex smart contracts and a new blockchain infrastructure tailored to its specific needs. Smart contracts must be meticulously coded to handle intricate financial operations, asset exchanges, and incentive mechanisms without introducing vulnerabilities. Additionally, building a purpose-built blockchain

requires significant expertise to ensure scalability, interoperability, and robustness against potential attacks. Any flaws in the smart contracts or blockchain protocol could lead to severe financial losses and undermine user trust (Ernst & Young, 2021; Buterin, 2014).

2) **Regulatory and Legal Risks:** The decentralized nature of the USC system is likely to attract scrutiny from regulatory bodies across different jurisdictions. Navigating the **complex** and evolving landscape of cryptocurrency regulations poses a significant challenge. Issues such as compliance with Anti-Money Laundering (AML) and Know Your Customer (KYC) regulations, classification of CAPE and USC as securities or commodities, and adherence to data protection laws can vary widely between countries. Furthermore, implementing necessary compliance measures could inherently compromise the system's decentralization. For instance, enforcing AML/KYC typically requires centralized data collection and verification processes, which contradicts the USC system's objective of distributing control among participants without relying on central authorities. Failure to comply with these regulations could result in legal penalties, restrictions on operations, or forced alterations to the system's design, thereby hindering its global adoption and functionality. Additionally, the need to balance regulatory compliance with maintaining decentralization presents a complex dilemma, potentially necessitating compromises that could undermine the system's foundational principles of trustlessness and distributed governance (Zhu, 2023; Van Valkenburgh, 2021; Schoenmakers & Mastrogiacomo, 2020; Kaal & Milne, 2020).

3) **Economic Sustainability of Incentives:** Maintaining a 10% APR through the Treasury Credit **Module** raises concerns about the long-term economic sustainability of the incentive structure. High interest rates require substantial and consistent revenue streams to fund the rewards. Market fluctuations, decreased staking returns, or reduced capital inflows could jeopardize the ability to sustain these incentives. If the system cannot continuously offer attractive returns, participant engagement may decline, weakening the stability mechanisms that rely on active user participation.

4) **Adoption and Network Effects:** Achieving widespread adoption is crucial for the USC system's success, as network effects significantly enhance the utility and stability of a **stablecoin**. However, penetrating a market dominated by established stablecoins like USDC, USDT, and DAI is challenging. Building a robust user base requires extensive marketing, partnerships, and trust-building measures. Without sufficient adoption, the system may struggle to achieve the necessary liquidity and capital flows to maintain the USC peg and provide effective incentives, limiting its impact on the decentralized finance ecosystem.

5) **Dependence on the CAPE Token:** The USC system's reliance on the CAPE token introduces risks associated with its hypothetical nature. As CAPE is not an established asset, its market acceptance, liquidity, and price stability remain uncertain. If CAPE fails to gain traction or experiences significant volatility, it could undermine the USC's backing ratio, compromising the stablecoin's stability and

user confidence.

6) Complexity and User Understanding: The USC system's multifaceted design, involving different components, presents a steep learning curve for users. Navigating these interconnected mechanisms requires a deep understanding of decentralized finance principles, smart contract interactions, and the specific operational workflows of the system. This complexity may deter less technically proficient users, limiting the system's accessibility and broader adoption. Simplifying user interfaces and providing comprehensive educational resources are essential to mitigate this barrier.

7) Security Risks: Despite rigorous design, the USC system remains susceptible to security threats such as smart contract exploits, hacking attempts, and technical failures. Vulnerabilities in the smart contracts could be exploited by malicious actors to drain funds or manipulate the system's mechanisms, leading to financial losses and reputational damage. Ensuring robust security measures, including regular audits, formal verification, and continuous monitoring, is imperative to protect the system from potential breaches and maintain user trust.

8) Interdependence of System Components: The USC system's components exhibit a high degree of interdependence, meaning that a failure or inefficiency in one **component** can initiate cascading effects throughout the entire system. For example, a malfunction in the CAPE Bonder could disrupt the allocation of funds to the Treasury Credit Module's rewards pool. This disruption may lead to a shortage of funds, resulting in a decrease in the APR offered to participants. A reduced APR could discourage users from making deposits into the Treasury Credit Module, thereby preventing a decrease in the circulating supply of USC and weakening the system's backing. Similarly, issues within the Merchant component could impair the provision of liquidity, making it difficult or impossible to execute emergency measures if the usc_support metric falls below 1. Such liquidity impairments would undermine the system's ability to stabilize USC during periods of market stress. Therefore, ensuring seamless integration and implementing robust fail-safes across all components are critical to preventing systemic failures and maintaining the overall stability of the USC system.

9) Dependence on a Developed DeFi Ecosystem: The effectiveness and stability of the USC **system** rely heavily on the existence of a vibrant and well-developed decentralized finance ecosystem within its own blockchain. This ecosystem encompasses lending platforms, decentralized exchanges (DEXs), and other DeFi protocols that provide essential utility and liquidity for both CAPE and USC. Without a comprehensive suite of DeFi services, funds may begin to exit the system, leading to a depletion of capital within the Merchant component. This outflow undermines the effectiveness of ETH staking, as reduced funds limit the Merchant's ability to generate staking rewards and maintain sufficient liquidity. Consequently, the system's capacity to provide emergency funds in scenarios where the usc_support metric drops below 1 is significantly impaired. Ensuring that funds remain within the system is crucial for maintaining the Merchant's liquidity

and the overall stability of USC. Therefore, developing a robust DeFi ecosystem within the USC blockchain is essential to retain capital within the Merchant, sustain effective staking operations, and ensure the system can respond adequately to fluctuations in the backing ratio.

4.3. Unobvious Advantages of the USC System

1) **Compounding Effect of the Merchant Mechanisms:** The Merchant component facilitates real ETH flows from staking profits and over-collateralized loans. They are **strategically** reinvested to compound the system's assets. Specifically, 70% of the profits generated from staking real ETH (or real ETH that was received as a result of receiving a loan) are reinvested back into staking, thereby expanding the staking pool and increasing yield generation. Simultaneously, 30% of the profits (or real ETH that was received as a result of receiving a loan) are allocated to reserve pools to ensure liquidity and provide a safety net for withdrawals. Additionally, the system mints Synthetic ETH (corresponding to newly acquired real ETH from staking profits or loans), which is either allocated to the Treasury Credit Module's rewards pool or directly to the Treasury. This dual usage approach not only amplifies the system's asset base but also reinforces the stability and incentivization mechanisms essential for maintaining the USC peg.

2) **Enhanced Borrowing Capacity through the Merchant:** The Merchant plays a pivotal role in augmenting the Treasury's borrowing capacity, which is intrinsically linked to the size and robustness of the blockchain's DeFi ecosystem. While the Treasury manages the reserves required to back USC, the Merchant oversees all funds within the blockchain's ecosystem. This division of responsibilities ensures that the Treasury can leverage the Merchant's holdings to access substantial liquidity. As the DeFi ecosystem within the blockchain expands, the Merchant's holdings grow proportionately, directly enhancing the Treasury's ability to borrow funds.

3) **Automatic Growth of Over-Collateralization Buffer during Bull Markets:** In periods of market growth, the USC system benefits from the natural appreciation of underlying assets, thereby enhancing its over-collateralization buffer without necessitating additional deposits. As the value of ETH or CAPE appreciates, the Treasury's holdings increase proportionately, strengthening the over-collateralization buffer. This automatic growth of the buffer zone ensures that each USC remains well-backed, sustaining its peg and reinforcing user confidence without placing additional burdens on Treasury.

4) **Attractive APR in the Treasury Credit Module:** The Treasury Credit Module offers a 10% Annual Percentage Rate (APR), which serves as a compelling incentive for participants to lock their USC and support the system's stability. This high APR is designed to surpass most traditional finance alternatives, encouraging users to deposit their USC into the Treasury Credit Module. The influx of locked funds reduces the circulating supply of USC, alleviating pressure on the Treasury to back each USC in circulation. Moreover, the attractive APR not only draws

sustained participation but also ensures a steady flow of capital into the system, which is crucial for maintaining stability and supporting the system's economic mechanisms.

4.4. Limitations of the Study Process

- Lack of Realistic Participant Behavior Modeling
- No Consideration of External Shocks
- No real-world testing.

4.5. Conclusion

The USC stablecoin system emerges as a promising solution to the longstanding Stablecoin Trilemma, adeptly balancing stability, decentralization, and capital efficiency. Through the innovative integration of its core components—Treasury, Treasury Credit Module, CAPE Bonder, and Merchant—the USC system demonstrates its capability to maintain a fully backed stablecoin, as evidenced by the usc_support metric consistently remaining above the critical threshold of 1 throughout the 298-day simulation period. This consistent backing underscores the system's resilience against market downturns, thereby fostering user confidence and potentially preventing destabilizing phenomena such as bank runs. A pivotal element contributing to this stability is the Treasury Credit Module, which offers an attractive 10% APR to participants, incentivizing the locking of USC and effectively reducing its circulating supply during periods of declining asset values. Additionally, the Merchant component serves as a significant safety net by providing substantial borrowing capacity—up to 67,315 ETH (approximately \$160.6 million)—which is nearly three and a half times the Treasury's ETH holdings. This extensive liquidity provision enhances the system's ability to uphold the USC peg in the face of significant market volatility, thereby reinforcing the system's robustness. Despite these promising outcomes, the study acknowledges several key limitations that must be addressed to ensure the USC system's successful implementation and long-term viability. Furthermore, the study process itself is subject to limitations. Ultimately, the USC stablecoin system offers a compelling framework for the next generation of stablecoins, promising enhanced stability, decentralization, and capital efficiency. With continued innovation and strategic refinement, the USC system has the potential to significantly impact the decentralized finance landscape, paving the way for more resilient and trustworthy stablecoin solutions.

4.6. Future Work

Future research and development efforts should focus on:

- Dynamic Modeling of Participant Behavior: Incorporating models that account for irrational actions, emotional responses, and diverse decision-making processes.
- Incorporation of External Shocks: Evaluating the impact of regulatory changes, technological failures, and macroeconomic events.

- Empirical Validation: Deploying pilot programs or controlled environments to observe real-world performance and gather data.
- Development of the DeFi Ecosystem: Creating initial DeFi services, incentivizing developers, and building partnerships to ensure a vibrant ecosystem within the USC blockchain.
- Economic Sustainability Analysis: Examining the long-term feasibility of incentive structures and exploring adaptive mechanisms.
- Security Enhancements: Continuously improving security protocols, conducting audits, and engaging the community in safeguarding the system.
- Regulatory Engagement: Proactively working with regulators to navigate legal challenges and ensure compliance.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

- Adams, R., & Van Den Bergh, M. (2020). Stablecoins: Regulatory Perspectives and Future Directions. *Journal of Financial Regulation*, 6, 123-145.
- Allen, F., & Wysocki, P. (2019). Decentralized Finance (DeFi): Transforming Traditional Financial Systems. *International Journal of Financial Studies*, 7, 45-60.
- Buterin, V. (2014). *A Next-Generation Smart Contract and Decentralized Application Platform*. Ethereum Whitepaper. <https://ethereum.org/en/whitepaper/>
- Catalini, C., & Gans, J. S. (2016). *Some Simple Economics of the Blockchain*. MIT Sloan Research Paper No. 5191-16.
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2874598
- Ernst & Young (2021). *Blockchain in Financial Services: Driving Innovation and Enhancing Security*. https://www.ey.com/en_gl/blockchain
- Fidelity Digital Assets (2022). A Closer Look at Bitcoin's Volatility.
<https://www.fidelitydigitalassets.com/research-and-insights/closer-look-bitcoins-volatility>
- Kaal, W. A., & Milne, A. (2020). The Blockchain-Based Stablecoin: Opportunities and Challenges. *Journal of Digital Banking*, 5, 157-168.
Olympus Dao Docs. <https://docs.olympusdao.finance/>
- Puschmann, T. (2017). Fintech. *Business & Information Systems Engineering*, 59, 69-76.
<https://doi.org/10.1007/s12599-017-0464-6>
- Schär, F. (2021). Decentralized Finance: On Blockchain- and Smart Contract-Based Financial Markets. *Federal Reserve Bank of St. Louis Review*, 103, 153-174.
<https://doi.org/10.20955/r.103.153-74>
- Schoenmakers, B., & Mastrogiacomo, A. (2020). Regulatory Challenges for Stablecoins and Central Bank Digital Currencies. *Journal of Financial Regulation and Compliance*, 28, 320-333.
- Stablecoins.wtf (n.d.). *The Stablecoin Trilemma*.
<https://stablecoins.wtf/resources/the-stablecoin-trilemma18>
- Van Valkenburgh, C. (2021). Regulating Stablecoins: Navigating the Intersection of Finance and Technology. *Stanford Journal of Blockchain Law & Policy*, 3, 1-45.

- Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. *Ethereum Project Yellow Paper*, 151, 1-32.
- Zhu, K. (2023). Legal Regulation of Stablecoins. *Beijing Law Review*, 14, 1142-1150.
<https://doi.org/10.4236/blr.2023.143060>

Appendix I: Table with Results of the Simulation

Appendix II: Simulation Code

```

import random
import pandas as pd
import seaborn as sns

# USC Stablecoin Simulation by Alexander Kartavchenko & Mikhail Kartavchenko
# Research: "Overcoming the Stablecoin Trilemma with a New Peg Maintenance System"
# Part of the Capstone Research Initiative
#July 2024

class CapePool():
    def __init__(self, eth_amount, cape_amount, fee_percent=0.003):
        #params
        self.fee_percent = fee_percent
        self.eth_amount = eth_amount
        self.cape_amount = cape_amount
        self.constant = self.eth_amount * self.cape_amount

        #state
        self.total_cape_fee = 0
        self.total_eth_fee = 0

    def get_cape_price(self, eth_price):
        """
        Calculate the price of CAPE in terms of ETH in USD
        """
        return self.eth_amount / self.cape_amount * eth_price

    def sell_cape(self, sold_cape_amount):
        """
        Add CAPE tokens to the pool and update ETH amount according to
        (ETH_amount - x)(CAPE_amount + sold_cape_amount) = const
        Return bought ETH amount
        """

        # calculate fee and charge it
        self.total_cape_fee += sold_cape_amount * self.fee_percent
        sold_cape_amount = sold_cape_amount * (1 - self.fee_percent)

        # update pool balance
        self.cape_amount += sold_cape_amount
        updated_eth_amount = self.constant / self.cape_amount
        bought_eth_amount = self.eth_amount - updated_eth_amount
        self.eth_amount -= bought_eth_amount

        return bought_eth_amount

    def sell_eth(self, sold_eth_amount):
        """
        Remove ETH tokens from the pool and update CAPE amount according to
        (ETH_amount + sold_eth_amount)(CAPE_amount - x) = const
        Return bought CAPE amount
        """

        # calculate fee and charge it
        self.total_eth_fee += sold_eth_amount * self.fee_percent
        sold_eth_amount = sold_eth_amount * (1 - self.fee_percent)

        # update pool balance
        self.eth_amount += sold_eth_amount
        updated_cape_amount = self.constant / self.eth_amount
        bought_cape_amount = self.cape_amount - updated_cape_amount
        self.cape_amount -= bought_cape_amount

        return bought_cape_amount

# USC Stablecoin Simulation by Alexander Kartavchenko & Mikhail Kartavchenko
# Research: "Overcoming the Stablecoin Trilemma with a New Peg Maintenance System"
# Part of the Capstone Research Initiative
#July 2024

class Treasury:
    def __init__(self, fee_percent=0.001, eth_fraction=0.4, cape_fraction=0.6):
        #params
        self.fee_percent = fee_percent

```

```

self.cape_fraction = cape_fraction
self.eth_fraction = eth_fraction

# state
self.total_eth_fee = 0
self.total_eth_amount = 0
self.total_cape_amount = 0
self.total_wasted_cape_amount = 0
self.total_printed_cape_amount = 0
self.total_emitted_usc = 0

def buy_usc(self, eth_amount, cape_amount, eth_price, cape_price, fee=True):
    """
    Buy USC using the mixture of CAPE and ETH (the proportion of tokens is controlled outside)
    Return emitted (bought) USC amount
    """
    if fee:
        # calculate fee and charge it
        usd_amount = eth_amount * eth_price + cape_amount * cape_price
        usd_fee = usd_amount * self.fee_percent
        eth_fee = usd_fee / eth_price
        self.total_eth_fee += eth_fee
        eth_amount -= eth_fee

    usd_amount = eth_amount * eth_price + cape_amount * cape_price
    usc_amount = usd_amount

    self.total_eth_amount += eth_amount
    self.total_cape_amount += cape_amount
    self.total_emitted_usc += usc_amount

    return usc_amount

def sell_usc(self, usc_amount, eth_price, cape_price):
    """
    Sell USC for the mixture of CAPE and ETH (the proportion of tokens is controlled by Treasury)
    Return bought ETH and CAPE amount
    """
    usc_fee = usc_amount * self.fee_percent
    usd_fee = usc_fee

    usd_amount = usc_amount
    cape_amount = usd_amount * self.cape_fraction / cape_price
    eth_amount = usd_amount * self.eth_fraction / eth_price

    eth_amount -= usd_fee / eth_price

    self.total_eth_amount -= eth_amount
    self.total_cape_amount -= cape_amount
    self.total_emitted_usc -= usc_amount

    return eth_amount, cape_amount

def deposit_wasted_cape(self, cape_amount):
    """
    Deposit CAPE amount as wasted CAPE
    """
    self.total_wasted_cape_amount += cape_amount

def get_total_balance(self, eth_price, cape_price):
    """
    Calculate total balance (CAPE + ETH)
    Return balance in USD
    """
    return self.total_cape_amount * cape_price + self.total_eth_amount * eth_price

def calc_cape_fraction(self, eth_price, cape_price):
    """
    Calculate current CAPE fraction in total balance (CAPE + ETH)
    Return CAPE fraction
    """
    return self.total_cape_amount * cape_price / self.get_total_balance(eth_price, cape_price)

def calc_usc_support(self, eth_price, cape_price):
    """
    Calculate balance support per 1 emitted USC
    Return USC support
    """
    return self.get_total_balance(eth_price, cape_price) / self.total_emitted_usc

```

```
# USC Stablecoin Simulation by Alexander Kartavchenko & Mikhail Kartavchenko
# Research: "Overcoming the Stablecoin Trilemma with a New Peg Maintenance System"
# Part of the Capstone Research Initiative
#July 2024
```

```
class CapeBonder():
    def __init__(self):
        # params
        self.cape_price_discount = 0.05
        self.lock_period = 5

        # state
        self.collected_cape = 0
        self.collected_eth = 0
        self.bonded_cape_list = []

    def buy_bond(self, eth_amount, eth_price, cape_amount, cape_price):
        """
        Buy bonded CAPE with the discount
        Return bonded CAPE amount
        """
        usd_amount = eth_amount * eth_amount * eth_price +\
            cape_amount * cape_price

        cape_bonded = usd_amount / (cape_price * (1 - self.cape_price_discount))

        self.collected_cape = cape_amount
        self.collected_eth = eth_amount
        self.bonded_cape_list.append(cape_bonded)
        return cape_bonded

    def emit_cape(self):
        """
        Emit CAPE at the corresponding simulation step if lock period has expired
        Return emitted CAPE amount
        """
        # length of bonded_cape_list equals the number of simulation steps
        simulation_step = len(self.bonded_cape_list)
        if simulation_step > self.lock_period:
            # -1 since we order list elements from 0
            return self.bonded_cape_list[simulation_step - self.lock_period - 1]
        else:
            return 0

# USC Stablecoin Simulation by Alexander Kartavchenko & Mikhail Kartavchenko
# Research: "Overcoming the Stablecoin Trilemma with a New Peg Maintenance System"
# Part of the Capstone Research Initiative
#July 2024

class Merchant():
    def __init__(self):
        #params
        self.reserve_percent = 0.3
        self.daily_interest_rate = 0.035 / 365
        self.borrow_rate = 0.3

        #state
        self.reserved_eth = 0
        self.staked_eth = 0
        self.earned_eth = 0
        self.already_borrowed_eth = 0

    def add_eth(self, eth_amount):
        """
        Add ETH to reserves and staking
        """
        # 30% of ETH goes to reserves, 70% to staking
        self.reserved_eth += self.reserve_percent * eth_amount
        self.staked_eth += (1 - self.reserve_percent) * eth_amount

    def earn_interest(self):
        """
        Earn interest from staked ETH
        Return earned interest in ETH
        """
        daily_interest = self.staked_eth * self.daily_interest_rate
        self.earned_eth += daily_interest
        self.reserved_eth += self.reserve_percent * daily_interest
        self.staked_eth += (1 - self.reserve_percent) * daily_interest
```

```

return daily_interest

def deposit_to_treasury(self, treasury, required_loan_eth, cape_price, eth_price):
    """
    Process loan accoring to the guideliness
    """
    # if we can fully take ETH loan from Merchant
    if required_loan_eth + self.already_borrowed_eth <= self.borrow_rate * self.staked_eth:
        treasury.total_eth_amount += required_loan_eth
        self.already_borrowed_eth += required_loan_eth
        self.reserved_eth += self.reserve_percent * required_loan_eth
        self.staked_eth += (1 - self.reserve_percent) * required_loan_eth

    else:
        possible_loan_eth = self.borrow_rate * self.staked_eth - self.already_borrowed_eth
        # if we can partially take ETH loan from Merchant
        if possible_loan_eth > 0:
            treasury.total_eth_amount += possible_loan_eth
            self.already_borrowed_eth += possible_loan_eth
            self.reserved_eth += self.reserve_percent * possible_loan_eth
            self.staked_eth += (1 - self.reserve_percent) * possible_loan_eth

        else:
            required_loan_cape = required_loan_eth * eth_price / cape_price
            treasury.total_cape_amount += required_loan_cape

# USC Stablecoin_Simulation by Alexander Kartavchenko & Mikhail Kartavchenko
# Research: "Overcoming the Stablecoin Trilemma with a New Peg Maintenance System"
# Part of the Capstone Research Initiative
#July 2024

class TreasuryCreditModule():
    def __init__(self, lock_period=90, max_interest_rate=0.10):
        # params
        self.lock_period = lock_period
        self.max_interest_rate = max_interest_rate

        # state
        self.deposited_usc_list = []
        self.total_usc_balance = 0
        self.rewards_pool_usc = 0
        self.total_dividends = 0
        self.total_potentially_removable_usc = 0

    def deposit(self, usc_amount):
        """
        Deposit USC amount to the balance
        """
        self.deposited_usc_list.append(usc_amount)
        self.total_usc_balance += usc_amount

    def allocate_rewards_pool(self, usc_amount):
        """
        Allocate rewards pool
        """
        self.rewards_pool_usc += usc_amount

    def pay_dividends(self):
        """
        Pay daily dividends using dynamic interest rate
        Return dividends and calculated interest rate (yearly)
        """
        # dynamic interest rate
        calculated_interest_rate = min(self.rewards_pool_usc / self.total_usc_balance, self.max_interest_rate)

        # calculate devidends for 1 day
        dividends = self.total_usc_balance * calculated_interest_rate / 365

        # pay dividends from reward pool
        self.total_dividends += dividends
        self.rewards_pool_usc -= dividends

        return dividends, calculated_interest_rate

    def potentially_removable_usc_cum(self):
        """
        Calculate cumulative potentially removable USC at the correspoding simulation step if lock period has expired
        Return cumulative amount of potentially removable USC

```

```

"""
# length of deposited_usc_list equals the number of simulation steps
simulation_step = len(self.deposited_usc_list)
if simulation_step > self.lock_period:
    # -1 since we order list elements from 0
    # add usc that can be removed to total cumulative usc sum
    self.total_potentially_removable_usc += self.deposited_usc_list[simulation_step - self.lock_period - 1]

return self.total_potentially_removable_usc

# USC Stablecoin Simulation by Alexander Kartavchenko & Mikhail Kartavchenko
# Research: "Overcoming the Stablecoin Trilemma with a New Peg Maintenance System"
# Part of the Capstone Research Initiative
#July 2024

cape_pool = CapePool(eth_amount=10**6, cape_amount=10**6)
cape_bonder = CapeBonder()
merchant = Merchant()
treasury = Treasury()
bank = TreasuryCreditModule()
merchant.add_eth(cape_pool.eth_amount)
records = []

usc_just_held_by_people = 0

eth_price = 1566.86

for t in range(298):

    # FIRST INCOMING FLOW; change by Misha: no 80% CAPE 20% ETH & wasted cape anymore, ratio is the same as in Treasury

    #ETH is increasing in price from t = 1 to t = 151, that is from October 12th 2023 to March 11th 2024

    if 1 <= t <= 151:
        eth_price *= (1 + 0.63 / 100)

    #ETH is decreasing in price from t = 152 to t = 184, that is from Mach 12th 2024 to April 13th 2024

    if 152 <= t <= 184:
        eth_price *= (1 - 0.94 / 100)

    #ETH is increasing in price from t = 185 to t = 228, that is from April 13th 2024 to May 27th 2024

    if 185 <= t <= 228:
        eth_price *= (1 + 0.60 / 100)

    #ETH is decreasing in price from t = 229 to t = 298, that is from May 28th 2024 to August 5th 2024

    if 229 <= t <= 298:
        eth_price *= (1 - 0.69 / 100)

    eth_to_bonder = 20

    # register ETH at Merchant
    merchant.add_eth(eth_to_bonder)

    # exchange 60% of ETH to CAPE in CapePool
    bought_cape_amount = cape_pool.sell_eth(0.6 * eth_to_bonder)
    cape_price = cape_pool.get_cape_price(eth_price)

    # buy bonded CAPE using 40% of ETH and 60% of CAPE
    cape_bonded = cape_bonder.buy_bond(0.4 * eth_to_bonder, eth_price, bought_cape_amount, cape_price)

    # exchange from bonder to treasury in the same proportion (ETH 40% to CAPE 60%)
    eth_to_exchange = cape_bonder.collected_eth
    cape_to_exchange = cape_bonder.collected_cape

    # buy USC in treasury
    usc_bought_by_bank = treasury.buy_usc(eth_to_exchange, cape_to_exchange, eth_price, cape_price, fee=False)

    # allocate USC to Bank's reward pool

```

```

bank.allocate_rewards_pool(usc_bought_by_bank)

# reset Bonder's balance
cape_bonder.collected_eth = 0
cape_bonder.collected_cape = 0

# check usc_support condition
usc_support = treasury.calc_usc_support(eth_price, cape_price)

# Third Incoming Flow
eth_for_third_flow = 100 # You can specify the amount of ETH to be used

# Calculate the amount of CAPE needed to meet the treasury's ratio
cape_needed = eth_for_third_flow * treasury.cape_fraction / treasury.eth_fraction

# Register ETH at Merchant (if needed)
merchant.add_eth(eth_for_third_flow)

# Buy the necessary CAPE from CapePool
cape_acquired = cape_pool.sell_eth(eth_for_third_flow * treasury.cape_fraction / (treasury.cape_fraction + treasury.eth_fraction))

# Buy USC using specified ETH and acquired CAPE
usc_bought_by_people_at_this_step = treasury.buy_usc(
    eth_for_third_flow * treasury.eth_fraction / (treasury.cape_fraction + treasury.eth_fraction),
    cape_acquired,
    eth_price,
    cape_pool.get_cape_price(eth_price)
)

# Update the amount of USC held by people
usc_just_held_by_people += usc_bought_by_people_at_this_step

# SECOND INCOMING FLOW - only execute if usc_support < 2
if usc_support < 1.5:
    eth_to_bank = 100

    # register ETH at Merchant
    merchant.add_eth(eth_to_bank)

    # exchange 60% of ETH to CAPE
    cape_bought = cape_pool.sell_eth(0.6 * eth_to_bank)

    # buy USC in treasury (40% ETH and 60% CAPE)
    usc_to_bank = treasury.buy_usc(0.4 * eth_to_bank, cape_bought, eth_price, cape_pool.get_cape_price(eth_price))

    # deposit USC to Bank
    bank.deposit(usc_to_bank)

    # reduce Treasury emitted USC balance
    treasury.total_emitted_usc -= usc_to_bank

else:
    eth_to_bank = 0
    usc_to_bank = 0
    bank.deposit(usc_to_bank)

# OTHER ROUTINE
daily_interest = merchant.earn_interest() #remove from condition
bank.allocate_rewards_pool(daily_interest) #remove from condition

# control treasury balance
if treasury.calc_usc_support(eth_price, cape_price) < 1:
    current_balance_usd = treasury.get_total_balance(eth_price, cape_price)
    usc_balance = treasury.total_emitted_usc

    # we want to have support >= 1
    required_loan_usd = usc_balance - current_balance_usd
    required_loan_eth = required_loan_usd / eth_price

    # merchant deposits to treasury
    merchant.deposit_to_treasury(treasury, required_loan_eth, eth_price, cape_price)

potentially_removable_usc_cum = bank.potentially_removable_usc_cum()
daily_dividends, daily_interest_rate = bank.pay_dividends()

cape_emitted = cape_bonder.emit_cape()

```

```

cape_pool.sell_cape(cape_emitted)
cape_fraction = treasury.calc_cape_fraction(eth_price, cape_price)
usc_support = treasury.calc_usc_support(eth_price, cape_price)

records.append(
    eth_to_bonder, eth_to_bank, eth_price, cape_bonded, cape_price, cape_emitted,
    merchant.earned_eth, merchant.reserved_eth, usc_to_bank, potentially_removable_usc_cum,
    daily_dividends, bank.total_dividends, daily_interest_rate,
    cape_fraction, usc_support, treasury.total_emitted_usc,
    treasury.total_eth_amount, treasury.total_cape_amount,
    merchant.already_borrowed_eth, usc_just_held_by_people, usc_bought_by_people_at_this_step
))

results = pd.DataFrame(records, columns=[
    'eth_to_bonder', 'eth_to_bank', 'eth_price', 'cape_bonded', 'cape_price', 'cape_emitted',
    'merchant.earned_eth', 'merchant.eth_amount', 'usc_to_bank', 'potentially_removable_usc',
    'daily_dividends', 'total_dividends', 'daily_interest_rate',
    'cape_fraction', 'usc_support', 'emitted_usc',
    'treasury.total_eth_amount', 'treasury.total_cape_amount',
    'already_borrowed_eth', 'usc_just_held_by_people', 'usc_bought_by_people_at_this_step'
])
```
results.head(298).loc[:, [
 'eth_to_bonder', 'eth_to_bank', 'eth_price', 'cape_bonded', 'cape_price', 'cape_emitted',
 'merchant.earned_eth', 'merchant.eth_amount', 'usc_to_bank', 'potentially_removable_usc',
 'daily_dividends', 'total_dividends', 'daily_interest_rate',
 'cape_fraction', 'usc_support', 'emitted_usc',
 'treasury.total_eth_amount', 'treasury.total_cape_amount',
 'already_borrowed_eth', 'usc_just_held_by_people', 'usc_bought_by_people_at_this_step'
]]
```

```