Helios Finance: Bitcoin-Native Lending with Adaptive Risk Optimization

1. Abstract

Helios is a Bitcoin-native decentralized finance (DeFi) lending protocol designed to unlock Bitcoin's liquidity without relying on cross-chain bridges or wrapped assets. Built on the Bitcoin mainnet via the MIDL execution layer, Helios enables users to lend and borrow directly in BTC with self-custody and one-step transactions. The system architecture preserves Bitcoin's security and finality while introducing smart contract capabilities through an EVM-compatible layer. We propose a novel risk management model for Helios that adaptively adjusts loan parameters in real-time based on Bitcoin network conditions and market volatility, formulated as a convex optimization problem. This model dynamically tunes loan-to-value (LTV) ratios and liquidation penalties to maintain solvency under varying conditions, without resorting to black-box machine learning. Core features of Helios include BTC-collateralized lending, seamless deposit/withdrawal in a single on-chain step, partial collateral swaps for flexible position management, and institutional-grade compliance options. We demonstrate how these features combine to improve capital efficiency—enabling low-risk, delta-neutral strategies for yield—as well as alignment with emerging regulatory requirements (e.g. tax-efficient borrowing and KYC-ready architecture). The paper outlines Helios's design and engineering in a rigorous format: from motivation and architecture to the mathematical risk framework (detailed in the Appendix) and implications for institutional adoption. Our results indicate that a Bitcoin-native lending platform with adaptive risk controls can offer both high security and capital efficiency, differentiating Helios from existing Ethereum-based protocols like Aave.

2. Introduction

Decentralized lending platforms have become a cornerstone of DeFi, allowing users to borrow and lend crypto assets without intermediaries. Protocols such as Aave and Compound on Ethereum have attracted billions in liquidity by enabling over-collateralized loans. However, Bitcoin—the largest cryptocurrency by market capitalization—has remained underutilized in DeFi due to technical barriers [12]. Using Bitcoin in DeFi typically requires **wrapped** tokens (e.g. WBTC) or tokenized representations on other chains, which entails moving BTC onto a separate network via **cross-chain bridges**. This process adds complexity, incurs custody risk, and introduces significant security vulnerabilities. In 2022 alone, over $2 billion was stolen in cross-chain bridge hacks, accounting for 69% of all crypto stolen that year [3]. These incidents erode trust and highlight the need for a **BTC-native** solution that does not depend on external bridges.

Motivation: Helios is motivated by the enormous untapped liquidity of Bitcoin in DeFi. Despite Bitcoin's ~$2 Trillion market cap, most BTC sits idle or only enters DeFi through risky intermediaries [12]. A trustless platform that enables lending and borrowing directly on the Bitcoin network could unlock this liquidity while preserving Bitcoin's robust security. By eliminating the need for wrapping or bridging, Helios avoids the single points of failure and custodial risk associated with federated peg systems (for example, the RSK sidechain requires BTC to be locked and converted to RBTC via a federated multisig [16]). Instead, Helios operates natively on Bitcoin, meaning users can supply BTC as collateral and earn yield or take out loans without leaving the Bitcoin blockchain.

Helios vs. Aave (DeFi Context): Aave is a widely-used lending protocol on Ethereum (and other chains) that introduced innovations like flash loans and a wide variety of supported assets. However, Aave and similar platforms are **Ethereum-centric**; they support BTC liquidity only in indirect forms (e.g. WBTC or tokenized BTC on Ethereum). This means Bitcoin holders using Aave must trust a bridge or custodian to hold their BTC, which introduces **network/bridge risk** acknowledged by Aave governance [1]. In contrast, Helios is *Bitcoin-first* – it does not require any token wrapping. All loans are secured by real BTC on the mainnet, and interest payouts can occur in BTC or BTC-denominated assets. Helios also differentiates itself through an **adaptive risk management** approach. Whereas Aave's risk parameters (like collateral ratios) are set through governance and adjusted intermittently [1], Helios automates these adjustments continuously based on on-chain conditions (discussed in Section 5). This results in a more responsive system that can preemptively manage risk (e.g. tightening collateral requirements during high volatility or network congestion).

Another key differentiation is user custody and transaction flow. On Ethereum lending platforms, users typically approve and transfer tokens to a smart contract, receiving a tokenized deposit receipt. With Helios on Bitcoin, we leverage MIDL – an execution layer that enables one-step interactions with Bitcoin smart contracts [12]. A Helios user can initiate a lending or borrowing action with a single Bitcoin transaction, without any preliminary fund transfers or separate approval steps. This **one-step deposit/withdrawal** design, made possible by MIDL's architecture, stands in contrast to multi-step processes often required in Ethereum DeFi (for example, wrapping BTC then depositing). Helios thus aims to provide a seamless UX comparable to Ethereum DeFi, but on the Bitcoin base layer.

Paper Organization: The remainder of this paper is structured as follows. Section 3 describes the Helios system architecture, including deployment on Bitcoin mainnet via MIDL and plans for expansion to layer-2 networks. Section 4 details Helios's core features such as BTC-native lending, self-custody mechanisms, and partial collateral swaps. In Section 5, we present the risk management model with mempool-aware logic and adaptive parameters. Section 6 gives an overview of the mathematical framework underpinning this model (with a full formulation provided in the Appendix). Section 7 discusses how Helios enhances capital efficiency and enables low-risk, delta-neutral strategies for users. Section 8 addresses regulatory alignment and design considerations for institutional adoption. We conclude in Section 9. References and an Appendix containing the convex optimization model for risk management are included at the end.

3. System Architecture

Helios is deployed on the Bitcoin mainnet and utilizes **MIDL (Modular Integrated Decentralized Ledger)** as its execution layer. MIDL augments Bitcoin with an Ethereum-compatible smart contract environment, operating as an overlay network anchored to Bitcoin transactions [12]. This architecture allows Helios to achieve the expressiveness of Ethereum smart contracts (for lending logic, interest calculations, etc.) while directly transacting in BTC. The design consists of several layers and components working together:

- **Bitcoin Base Layer (Settlement):** All value transfer in Helios ultimately occurs on the Bitcoin blockchain. User collateral and loan repayments are settled in BTC L1 transactions. Bitcoin's proof-of-work chain provides final settlement assurance. Notably, Helios leverages Bitcoin's security and block space for finality: each Helios operation is confirmed with a single Bitcoin block (instead of waiting for multiple confirmations as traditionally done for finality) [12]. This means that loan deposits and withdrawals are considered finalized after one block confirmation, thanks to the deterministic design of MIDL that mitigates reorg risks. The base layer also records periodic state commitments from the execution layer (described below), anchoring Helios's contract state to Bitcoin in a verifiable way.

- **MIDL Execution Layer (Smart Contracts on BTC):** MIDL acts as an EVM-compatible layer on top of Bitcoin [12]. Technically, MIDL is not a separate blockchain but an integrated protocol: it uses the Bitcoin network for ordering transactions and embeds an overlay consensus (Delegated Proof-of-Stake, DPoS) among a set of **validators** [12]. Helios's smart contract logic (e.g. functions for depositing collateral, withdrawing loans, calculating interest, liquidations, etc.) runs within this MIDL layer. Users interact with Helios via standard Bitcoin transactions carrying data that invoke MIDL contracts. For example, a user sending BTC to a specific MIDL contract address (managed by the validators) with an embedded instruction can trigger a "deposit and borrow" action. The MIDL validators execute the corresponding EVM bytecode off-chain (enforcing Helios's lending rules) and then commit the results on-chain. Each batch of contract executions produces a cryptographic commitment (e.g. a Merkle root of the new state) that is written into a Bitcoin transaction [12]. This approach gives a **verifiable, tamper-resistant log** of Helios's contract state changes on Bitcoin. In essence, MIDL extends Bitcoin with a general computation layer for Helios, without altering Bitcoin's consensus rules.

- **Validators and TSS Vaults (Security and Self-Custody):** Helios relies on a decentralized set of validator nodes (the MIDL validators) to carry out contract execution and manage multisignature vaults. When a user supplies BTC to Helios, the funds are held in a **Threshold Signature Scheme (TSS) vault** controlled collectively by the validators [12]. A TSS vault is effectively a Bitcoin multisig wallet where a majority of validators must sign to move funds, preventing any single validator from misappropriating collateral. Importantly, user actions require an **intent signature** by the user's own Bitcoin key as well [12]. For example, to initiate a withdrawal of collateral, the user must sign an intent message authorizing the action, which validators verify. The actual BTC is then released from the multisig vault to the user's address once the conditions (like loan repayment) are met and a quorum of validators signs off. Because the user's private key is required to initiate critical actions (and never leaves the user's device) [12], the system preserves **self-custody**: users maintain ultimate control and authorization over their funds. The validators' role is automated and constrained by the Helios smart contracts and multi-party protocols – they cannot arbitrarily seize or redirect assets without the user's consent and proper protocol conditions. Misbehaving validators face slashing of their staked BTC and removal from the validator set [12], providing strong economic security. This design removes any central custodian; custody is distributed among independent participants, aligning with Bitcoin's trust model.

- **One-Step Transactions:** Thanks to MIDL's architecture, Helios supports one-step deposit and withdrawal interactions. Users do not need to perform separate bridging or token conversion transactions. For instance, to supply BTC as collateral and simultaneously borrow a stablecoin, a user simply sends a Bitcoin transaction to Helios with the BTC amount and a payload indicating the borrow request. This single transaction both locks the BTC into the TSS vault and triggers the lending contract, which mints or releases the stablecoin to the user in the same process (the stablecoin itself might be issued within MIDL as a Bitcoin-anchored token). All operations execute atomically within one Bitcoin block [12]. Similarly, loan repayment and collateral return can happen in one step. This represents a major UX improvement: there is no need to first wrap BTC, then move it, then deposit; Helios condenses the workflow into a single on-chain action, leveraging MIDL's capability to bundle up to 10 contract calls within one Bitcoin transaction [12].

- **Expansion to Bitcoin Layer-2s:** While Helios's initial deployment is on Bitcoin L1, its architecture is extensible to layer-2 networks and sidechains for scalability. Future versions of Helios may integrate with Bitcoin **Lightning Network** and sidechains such as Rootstock (RSK) or Liquid. The motivation is to increase transaction throughput and reduce fees for smaller transactions, while still using BTC as the unit of account. For example, one could imagine Helios channels on the Lightning Network that allow rapid, low-fee loan repayments or interest payments, settling back to mainnet periodically. Similarly, Helios contracts could be ported to a sidechain like RSK or a rollup, where smart contracts execute faster/cheaper, and BTC moves via a pegging mechanism. The Helios design, by keeping custody on Bitcoin and using MIDL as an adaptor, could allow a hybrid approach: large-value operations settle on mainnet, whereas microtransactions or high-frequency trades utilize an L2. An essential requirement for any L2 expansion is maintaining decentralization and security: e.g. RSK uses merge-mining and a federated peg, which introduces some trust assumptions; Lightning is trustless but limited in liquidity for large loans. Helios's roadmap includes exploring these trade-offs. Initially, by starting on Bitcoin L1 with batched contract execution, Helios achieves decent throughput (up to 10 contract actions per Bitcoin block) [12]. As Bitcoin L2 technology matures (e.g. potential validity rollups or drivechains), Helios can migrate or extend onto those to support a greater volume of loans and more complex products, all while keeping BTC as the native asset throughout. The modularity of the MIDL execution layer means Helios's core logic is portable – the same lending contracts could run on a Bitcoin sidechain environment with minimal changes, ensuring any expansion does not require re-architecting the protocol.

In summary, Helios's architecture marries Bitcoin's security and liquidity with Ethereum-like smart contract functionality. By using the MIDL execution layer and a validator-managed vault system, Helios achieves a **trust-minimized**, Bitcoin-native platform. Users interact with Helios through their Bitcoin wallets, and all critical transactions (deposits, withdrawals, liquidations) are settled on-chain in BTC. This architecture provides the foundation for the innovative features and risk controls discussed in subsequent sections.

4. Core Features of Helios

Helios introduces a suite of core features that distinguish it from traditional crypto lending platforms. These features prioritize security, user control, and seamless user experience, all while leveraging Bitcoin's unique strengths. Below we outline the primary features:

- **BTC-Native Lending and Borrowing:** All loans on Helios are denominated and collateralized in **native BTC** (or BTC-derived assets on Bitcoin) rather than wrapped tokens. Lenders deposit Bitcoin and earn interest in Bitcoin; borrowers post BTC as collateral to borrow either BTC or other Bitcoin-based assets (such as a Bitcoin-backed stablecoin). By operating natively on Bitcoin, Helios eliminates bridge and custody risks—users' BTC never leaves the Bitcoin blockchain at any point [12]. This is a stark contrast to Ethereum-based lenders where BTC must be represented by a token like WBTC (with attendant counterparty risk). Native support means that Bitcoin's liquidity can be directly utilized for lending. Additionally, interest rates and loan terms are structured in BTC terms, aligning with the preferences of many Bitcoin holders to accumulate more BTC. Helios effectively brings the lending functionality of Aave/Compound into the Bitcoin ecosystem without compromising on decentralization.

- **Self-Custody via Threshold Signatures:** A cornerstone of Helios is that users retain **control of their keys and funds** to the greatest extent possible. When users deposit BTC into Helios, those funds go into a TSS vault that requires multiple parties to authorize movements [12]. The user's own signature (via BIP-322 attestations) is needed to initiate any withdrawal of their collateral [12], ensuring that funds cannot leave the system without user consent. Unlike centralized lenders or some DeFi platforms, there is no single entity in Helios that can freeze or seize collateral; even the Helios validators can only act according to protocol rules with majority consensus. This self-custodial design mitigates risks of theft and bankruptcy that have plagued centralized crypto lenders. It also means that **"not your keys, not your coins"** remains true – Helios users do not surrender ownership of BTC at any point, they merely lock it under smart contract-enforced conditions. The private keys controlling the BTC remain split between the user (who holds their wallet key) and the validator multisig quorum (which acts strictly per the contract logic). In practice, users interact with Helios through their existing Bitcoin wallets and signing devices, providing a familiar security model. Self-custody also appeals to institutions that have strict custody requirements; with Helios, institutions can use their own qualified custodian or custody solution to hold the keys that interface with Helios, satisfying internal security policies.

- **One-Step Deposit & Withdrawal:** Helios streamlines the user experience by bundling actions into a single on-chain step. A **single Bitcoin transaction** can both deposit collateral and execute a borrow or lend action. For example, if a user wants to supply 1 BTC to earn interest, they send 1 BTC directly into Helios's address with the appropriate data, and immediately Helios credits their account and starts accruing interest. Similarly, withdrawing collateral after loan repayment happens in one step. This is enabled by MIDL's ability to handle complex scripts that atomically move funds and update contract state [12]. The net effect is a **simplified UX** akin to a traditional bank: deposit and withdrawal feel like singular operations, not multi-step blockchain processes. One-step interactions also reduce fees and time. There is no need for separate allowance or approval transactions (common on ERC-20 based lending), which on Bitcoin translates to fewer transactions and UTXOs to manage. Users pay a single Bitcoin network fee for a combined operation, which is economical. Additionally, eliminating intermediate steps like token swaps or wraps means fewer potential points of failure or user error. The one-step design is particularly powerful on Bitcoin, where block times average ~10 minutes; by consolidating operations, Helios ensures that users wait for confirmation just once per action, rather than multiple 10-minute intervals. For the user, lending on Helios is as straightforward as sending a normal BTC transaction, making decentralized lending more accessible to the average Bitcoiner.

- **Partial Collateral Swap:** Helios allows borrowers to **swap their collateral asset** for another supported asset without fully closing out their loan. This innovative feature gives users flexibility to manage their exposure and potentially improve their position's stability or tax treatment. For instance, suppose a user borrowed against BTC collateral and later wishes to hold a more stable collateral (e.g. a BTC-denominated stablecoin or a different crypto asset) to reduce volatility risk. Traditionally, the user would need to repay the loan entirely, withdraw the BTC, trade it for the new asset, and then open a new loan — a cumbersome and taxable sequence. In Helios, the user can perform a **partial collateral swap**: a portion of the BTC collateral is atomically exchanged for an equivalent value of another asset (say a stablecoin or another token that Helios supports as collateral), which remains locked in the vault. The loan continues with the new collateral mix. This is done by using internal DEX functionality or auction mechanisms within the MIDL execution, ensuring the swap happens under the hood while maintaining the loan's continuity. Partial swaps can be used to **rebalance collateral** in response to market moves (e.g. moving into stable assets if BTC is expected to drop, or vice versa to capitalize on a rally). It can also be a tool for **tax optimization**: swapping collateral within the contract might not trigger a taxable event in some jurisdictions, since the original loan is still open and the user hasn't realized gains in fiat terms (though specific tax treatment can vary). The concept of collateral swap during an active loan has been explored in CeFi platforms (), and Helios brings it to DeFi in a trustless manner. Practically, Helios will support collateral swap between whitelisted asset types of sufficient liquidity. The swap is partial in that a user could choose to swap only a portion of collateral (e.g. 50% of BTC to a stablecoin) to retain some upside exposure. This feature enhances risk management for borrowers and adds a unique flexibility not commonly found in existing protocols.

- **Adaptive Interest Rates and Terms:** (Etc.) Beyond the headline features above, Helios incorporates additional capabilities to improve user experience and safety. For example, interest rates on Helios are determined algorithmically by supply and demand (similarly to Aave's rate curves), but with the twist that they can react to Bitcoin network conditions. If block space is scarce (high fees), Helios might adjust incentives for quicker loan repayments or higher liquidity to ensure smooth operations. Helios also plans to support **variable and fixed rate** loan options, with fixed-rate loans managed through separate pools or rate swaps, catering to different risk appetites. Another planned feature is **cross-collateral and cross-margin**: users can supply multiple types of collateral (when supported) and borrow multiple assets within a unified account, with Helios's risk engine monitoring the aggregate position. This resembles a prime brokerage model and increases capital efficiency by allowing diversification of collateral. All these features are built on the robust Bitcoin-native framework described earlier, ensuring that even advanced functionality remains trustless and transparent.

In sum, Helios's core features are aimed at delivering a secure, user-centric lending platform that capitalizes on Bitcoin's strengths. **No bridges, no custodians,** and **no unnecessary complexity** – Helios provides a **one-step**, self-custodial **native BTC lending** experience with flexibility like partial collateral swaps. These features set the stage for a highly efficient and resilient system, but they also introduce challenges in risk management, which we address with Helios's adaptive risk model in the next section.

5. Risk Management Model

Effective risk management is critical in a lending protocol, especially one dealing with volatile assets and variable network conditions. Helios employs a **dynamic risk management model** that continuously adjusts key parameters such as Loan-to-Value ratios and liquidation thresholds/penalties based on real-time indicators. The primary goal is to maintain the solvency of the lending pool and protect both lenders and borrowers from undue risk, while maximizing capital efficiency. Unlike traditional DeFi lending platforms that use static risk parameters set by governance [1], Helios's model is **adaptive and algorithmic**, responding to two main inputs: (a) Bitcoin mempool congestion and network conditions, and (b) asset price volatility and liquidity.

Mempool-Aware Adaptive Parameters: Bitcoin's network throughput and fee conditions can vary widely; during peak congestion, transactions can take extended time (hours) to confirm unless high fees are paid[18] [19]. This directly impacts the risk in a lending system: if a borrower's collateral value is plummeting, the protocol needs to execute a liquidation transaction quickly to repay lenders. If the mempool is backed up, that liquidation might be delayed, potentially resulting in **undercollateralized loans**. Helios addresses this by making the **maximum Loan-to-Value (LTV)** ratio allowed for loans a function of mempool congestion. When the mempool is low (fast confirmations), the system can safely allow higher LTVs because liquidations can occur promptly. When the mempool is highly congested, Helios automatically **lowers the permissible LTV** for new loans and may even trigger incremental collateral top-up requirements or partial liquidations for existing loans if they become high-risk. Similarly, the **liquidation penalty** (or bonus paid to liquidators) is adaptive ("LP" here refers to Liquidation Penalty/Bonus). Under normal conditions, the liquidation bonus might be a standard, small percentage. But if the network is congested (meaning a liquidator will have to pay high fees or wait longer, increasing their risk), Helios will **increase the liquidation bonus** to incentivize liquidation bots to prioritize these positions. This adaptive LP ensures that even in adverse network conditions, there is sufficient incentive for market actors to secure the protocol's solvency. In effect, Helios's smart contracts (or off-chain controllers) continuously monitor the mempool level (e.g. the median fee needed for next-block inclusion, or number of transactions queued) and adjust LTV and LP parameters within predefined bounds. These adjustments can occur per block or at a fixed time interval. By being *mempool-aware*, Helios preempts scenarios where borrowers could exploit network slowness to avoid liquidation or where lenders might suffer losses due to execution lag. This design is novel in the Bitcoin context, as it ties on-chain technical conditions to financial risk parameters in real time.

Volatility-Responsive LTV and Thresholds: Price volatility of the collateral (BTC or other assets) is the other key input. Helios uses price oracles (secure data feeds) to track the real-time price of BTC and any other collateral or loan asset. When market volatility is low and prices are relatively stable, Helios can afford to offer more generous borrowing terms (higher LTV, lower collateral requirements) to improve capital efficiency. But when volatility spikes, the probability of a sharp price drop that could liquidate collateral increases. Helios's model therefore tightens risk parameters during high volatility regimes. Concretely, the protocol computes an **adaptive LTV cap** based on recent volatility metrics (for instance, a short-term historical volatility or an implied volatility index if available). If BTC's 1-hour price volatility exceeds a certain threshold, Helios might reduce the max LTV for new loans by a certain percentage automatically. Likewise, the **liquidation threshold** (the point at which a loan is eligible for liquidation) can be dynamically lowered (meaning loans will get liquidated sooner if volatility is high, requiring more collateral buffer). This dynamic resembles a risk-based margin system: in traditional finance, brokers require higher margins in volatile markets; Helios brings a similar concept on-chain. The adjustments are done smoothly to avoid whipsawing users – for example, the LTV might change by at most 1% per hour to gradually adapt if volatility remains elevated. This gives borrowers time to respond (add collateral or repay loans) if needed, rather than sudden calls. Helios's volatility-responsive logic is underpinned by models from financial risk management, ensuring that the probability of loan default (failure to cover debt upon liquidation) stays below a target threshold (e.g. <0.1% probability given current volatility). Essentially, Helios attempts to keep the system's effective **Value-at-Risk** within acceptable bounds by adjusting collateral requirements on the fly.

Convex Optimization and Formal Risk Modeling: To coordinate the effects of multiple inputs (mempool, volatility, liquidity) on multiple parameters (LTV, liquidation penalty, reserve factors, etc.), Helios utilizes a **convex optimization framework** (described in Section 6 and Appendix). At a high level, Helios's risk engine periodically solves an optimization problem: it seeks to maximize capital efficiency (allowing as much useful borrowing as possible) subject to constraints on system safety (such as ensuring expected loss is minimized and no insolvencies occur under certain stress scenarios). This can be formulated, for example, as minimizing a risk cost function that includes terms for **probability of liquidation shortfall** and for **deviation from target utilization**, subject to linear constraints linking LTV and liquidation bonus to observed volatility and confirmation times. The choice of a convex (specifically, piecewise-linear convex) formulation is deliberate – it allows the protocol to reliably compute new parameters quickly and explain the decision rationale, as convex problems have unique, globally optimal solutions and are transparent to analyze. By contrast, approaches using machine learning or heuristic rules might be harder to audit or could behave unpredictably in novel conditions. Helios avoids opaque **neural network models** in risk parameter adjustment, instead favoring an approach grounded in well-understood financial engineering techniques (e.g. linear programming for worst-case optimization). For instance, Helios might model the worst-case price drop over the next block as a linear function of current volatility, and the worst-case liquidation delay as a linear function of mempool size. These yield linear constraints that relate how low collateral value could go vs. debt. The optimization then solves for the highest LTV that satisfies all these linear inequality constraints (ensuring even in worst-case delay and price drop, collateral > debt). The output might say, for example, "Given current conditions, max LTV = 68%, liquidation bonus = 8%" to maintain safety. In the next block, if conditions change, a new solution is found (say LTV 70% if things improve, or 65% if they worsen). This continuous recalibration keeps Helios resilient. The mathematical details are provided in the Appendix, including the objective function and constraints used.

Liquidation Mechanism and Partial Liquidations: Helios's liquidation process is similarly refined to mitigate risk. If a borrower's collateral value falls below the required level, Helios triggers a liquidation: a portion of the collateral is sold to repay the debt plus a liquidation penalty fee. Helios can perform **partial liquidations**, meaning only the amount of collateral necessary to bring the loan back to a safe LTV is sold, rather than the entire collateral at once. This prevents over-liquidation and can reduce the borrower's losses and market impact. The adaptive model determines the **repay factor** – what fraction of the debt must be repaid in a liquidation – which might be less than 100% if the situation isn't too severe [4]. By allowing partial liquidation, Helios keeps loans alive (the borrower keeps the remaining collateral) and gives them a chance to recover or repay later, which can be more equitable and also avoid flooding the market with collateral sales. The liquidation penalty collected (which goes to the liquidator or to an insurance fund) is, as noted, adjusted based on conditions to ensure liquidators are incentivized. Helios integrates with decentralized auction mechanisms or DEXes on the Bitcoin layer to execute liquidations transparently, and can even incorporate **mempool**

fee bumping (e.g. using Child-Pays-for-Parent or RBF fee increases) to ensure the liquidation transaction confirms quickly during congestion. All these measures collectively maintain a robust risk management stance: even during a sudden 20% drop in BTC price combined with a congested network, Helios will have dynamically tightened collateral requirements beforehand, and its liquidators will be motivated to respond due to higher rewards, thereby protecting lenders from loss.

In summary, Helios's risk management model is **proactive and rules-based**. By constantly observing network and market health metrics, Helios adjusts the system's guardrails in real time. This adaptivity significantly reduces the likelihood of cascade liquidations or systemic failures. Importantly, it achieves this without manual intervention – whereas other protocols rely on governance to tweak parameters after a crisis has started, Helios's algorithmic controller acts continuously to prevent crises before they happen. The next section provides an overview of the mathematical framework that enables this adaptive risk optimization, and the Appendix formalizes it with equations.

6. Mathematical Framework Overview

Helios's adaptive risk management is grounded in a mathematical framework that translates risk considerations into a solvable optimization problem. In this section, we outline the framework and its key components, deferring the full technical formulation to the Appendix. The objective is to provide an intuition for how Helios mathematically guarantees safety while optimizing for efficiency.

Optimization Objectives

At its core, Helios formulates the tuning of risk parameters (such as Loan-to-Value ratio (LTV) and liquidation penalty) as an optimization problem. One way to frame this is:

Maximize capital efficiency subject to risk constraints.

Capital efficiency can be approximated by aggregate borrow usage. A higher LTV allows more borrowing against a given amount of collateral, making more efficient use of funds. However, constraints are imposed to ensure that, even under worst-case conditions, the system remains solvent and liquidations can be executed without loss to lenders.

Alternatively, we can frame the problem as a dual objective:

Minimize risk (likelihood of shortfall) subject to a target level of efficiency.

In practice, Helios solves for a set of parameters that balances these goals—often by minimizing a weighted sum of a **risk metric** and an **inefficiency penalty**. Because both risk and efficiency can be expressed as convex functions of the decision variables, the resulting optimization is convex, which guarantees a stable, reliable optimum.

Convex, Piecewise-Linear Modeling

Helios avoids complex non-convexities in its design. While many real-world risk measures—like probability of ruin or Value-at-Risk—are non-linear, Helios approximates them using piecewise-linear functions to maintain convexity.

For example, suppose the worst-case 10-minute price drop for BTC, given a volatility measure sigma (σ), is approximated as:

Delta_max = a + b * σ

Here, a and b are calibrated constants representing baseline risk and sensitivity to volatility. This is a linear and therefore convex expression.

A constraint is then imposed such that:

(1 - Delta_max) * Collateral Value ≥ (1 + P_liq) * Debt

Where `P_liq` is the liquidation penalty (expressed as a fraction). This inequality is linear in terms of key variables like LTV (which determines Debt relative to Collateral) and `P_liq`.

Helios sets up similar constraints for a range of stress scenarios—different time horizons, volatility levels, and mempool delays. All of these constraints define a **convex feasible region** (essentially an intersection of half-spaces defined by linear inequalities). The optimization process finds the point in this region that either:

- Maximizes LTV, or
- Minimizes a cost function that penalizes risk.

Helios also uses piecewise-linear functions to model discrete effects like mempool congestion. For instance:
"If mempool size > X, assume Y blocks of delay," which can be represented as linear constraints.

By **avoiding multiplication of decision variables** or other non-convex operations, the optimization remains solvable as a **Linear Program (LP)** or **Quadratic Program (QP)**. These are solvable in polynomial time, making them suitable for frequent re-optimization.

Key Variables and Constraints

The main decision variables in Helios's optimization are:

- Maximum Loan-to-Value (L_max)
- Liquidation Penalty (P_liq)
- Possibly a Liquidation Threshold (typically just above L_max)
- Interest rate model parameters (as a secondary risk lever)

While interest rates are primarily driven by market conditions, Helios can adjust a base rate to incentivize or disincentivize borrowing under specific risk conditions.

Key Constraints Enforced:

1. **Collateral Adequacy (Stress Scenarios):**
   For each stress scenario s (with price drop Δ_s and transaction delay), ensure:
   (1 - Δ_s) * C ≥ (1 + P_liq) * D
   where C is collateral value and D is debt (at L_max).

2. **Liquidity Constraint:**
   Ensure that the collateral can be liquidated within available market depth, typically represented as:
   The fraction of daily volume to be liquidated ≤ a threshold.
   This yields a linear constraint on LTV.

3. **User Experience Constraint:**
   To avoid sudden changes, constrain how much L_max can change over time.
   Example: L_max_new - L_max_prev ≤ δ (per hour)

4. **Regime Constraint:**
   Enforce minimum and maximum bounds for parameters such as:
   50% ≤ L_max ≤ 80%
   These bounds are set by governance or initial calibration.

Objective Function

The general form of the objective function can be:

Maximize: L_max - λ * RiskCost

Where:

- L_max is the maximum safe loan-to-value ratio
- λ (lambda) is a trade-off weight
- RiskCost is a linear combination of slack variables or explicit risk metrics like expected shortfall

Alternatively, a simpler formulation might directly maximize L_max, pushing it to the point where one or more constraints are just binding (i.e., active). Or, Helios might target a safety margin and include a term in the objective that **minimizes the amount by which constraints are over-satisfied**, avoiding overly conservative behavior.

This remains a linear or convex objective.

Solution and Implementation

The optimization is solved by Helios's controller, which may run either:

- Off-chain (by validators or an oracle), or
- On-chain (if small enough for Solidity-based environments)

Because the optimization is convex and linear (or quadratic), it can be solved efficiently. Off-chain solutions may be preferable for performance reasons. The model's transparency allows any participant to verify that the published parameters satisfy all constraints using input data (e.g., price and mempool state). This provides **trustless assurance** that parameter changes are justified and risk-aware.

Example Scenario

Assume:

- Bitcoin price = $97,000
- Recent volatility implies a 1-block drop of 5%
- Mempool indicates a 2-block delay → model a 10% price drop

- Liquidation penalty = 5%

Then impose:

$(1 - 0.10) * C \geq 1.05 * D$
$\rightarrow 0.90 * C \geq 1.05 * D$

Since $D = L\_max * C$:

$0.90 * C \geq 1.05 * L\_max * C$
$\rightarrow 0.90 \geq 1.05 * L\_max$
$\rightarrow L\_max \leq 0.857$ (i.e., 85.7%)

Now add a market liquidity constraint:
If L_max must be ≤ 85% based on daily volume limits, then L_max = 0.85 is chosen.

Perhaps the optimizer also returns P_liq = 5%. These parameters are then enforced until inputs (e.g., volatility or mempool state) change.

Helios may also add a **safety buffer** (e.g., multiply Δ_max by 1.1 to account for model uncertainty), which is still linear.

Final Safety Guarantee

The outcome of this modeling approach is a **provable safety guarantee**:
If reality conforms to the modeled worst-case scenarios (which are conservatively defined), the system remains solvent. Even if the actual market behavior is more severe, the frequency of such extreme conditions is modeled as extremely low (comparable to a "once-in-a-century" event in TradFi).

Helios can also use an **insurance fund**, funded by interest revenue(and protocol token staking mechanism), to cover rare shortfalls—adding another layer of protection.

Helios's mathematical risk framework provides a solid foundation for **adaptive, automated lending operations**. By continuously solving a **convex optimization problem**, Helios dynamically adjusts risk parameters in response to changing market and network conditions.

This allows Helios to operate at an optimal point—neither overly conservative nor dangerously aggressive—based on real-time data, providing a **transparent, verifiable, and economically sound system**.

The full mathematical formulation, including proofs of convexity, is available in the Appendix.

7. Capital Efficiency and Strategy Enhancements

A major advantage of Helios's design is improved **capital efficiency** for users. Capital efficiency refers to how effectively deposited assets can be utilized to generate returns. Through its adaptive risk model and innovative features, Helios enables users to unlock more value from their Bitcoin holdings compared to traditional setups, all while controlling risk. In this section, we discuss how Helios achieves high capital efficiency and how users (especially sophisticated ones like quant funds or arbitrageurs) can employ low-risk, delta-neutral strategies on Helios to earn stable yields.

Higher Utilization via Adaptive LTV: Traditional lending platforms often set conservative, one-size-fits-all collateral requirements to account for worst-case scenarios. This means most of the time, the system is under-utilized (borrowers could safely borrow more, but are not allowed to). Helios's dynamic LTV adjustment ensures that during tranquil market periods, borrowers can access higher leverage. For example, if BTC price is stable and network is clear, Helios might allow an LTV of, say, 80%, whereas Aave might fixedly allow only ~70% for WBTC. This extra 10% translates to more borrowing power per BTC of collateral. For lenders, that means more of their deposited BTC is actually being borrowed by someone (higher utilization rate), which typically leads to higher interest earnings. Helios can safely push utilization to higher levels without added risk because the moment conditions change, the LTV is scaled down accordingly. In effect, Helios's users get **the best of both worlds**: high leverage when it's safe, and automatic deleveraging (or tightening) when it's not. This contrasts with other platforms where either leverage is limited at all times or manual intervention is needed during crises. The net result is that Helios can run at a higher average utilization. In DeFi lending, interest rates paid to suppliers (lenders) increase with utilization of the pool (FAQ | Aave). Therefore, lenders on Helios can earn more yield on average, and borrowers can borrow more against the same collateral, which is a win-win on efficiency.

Delta-Neutral Yield Strategies: Helios opens the door for **delta-neutral strategies** that generate yield with minimal price exposure. Delta-neutral strategies are common in DeFi and involve balancing long and short positions such that one's net exposure to price movements is zero [9]. On Helios, a user can implement such strategies entirely within the Bitcoin ecosystem, which was previously hard to do. For instance, consider an institution that holds BTC but wants to earn a USD-denominated yield without taking on BTC price risk (perhaps they believe BTC price might fall, but they still want to utilize their holdings). With Helios, they could do the following: deposit BTC into Helios as collateral and borrow Taproot Assets USDT or BTC-backed stablecoin (let's call it XUSD for example). Now they have XUSD which is pegged to USD, and an ongoing debt denominated in BTC. If they simply hold the XUSD, their position is short BTC (because if BTC's price rises, the BTC they owe is worth more in USD, effectively a short exposure). To neutralize that, they could **short the stablecoin against BTC** or equivalently, use the XUSD to buy more BTC. One straightforward approach: borrow BTC from Helios instead (if another user provided BTC liquidity to lend) and use their originally held BTC as collateral – effectively, they are borrowing BTC against BTC, which sounds redundant, but if structured properly, they can create a loop to adjust exposure. A clearer approach is: deposit

BTC, borrow XUSD, then *immediately convert XUSD back to BTC on a decentralized exchange (like a BTC-AMM on MIDL)*. Now the user has additional BTC in hand, but also a debt of XUSD. The extra BTC they bought can be used as further collateral or kept aside. The combined position is that they owe XUSD (a USD stablecoin) and hold an equivalent amount of BTC they bought – this is effectively a short USD, long BTC position which is not delta-neutral (it's actually leveraged long BTC). To truly go delta-neutral, the user should **short BTC while holding BTC collateral**. This can be done by *borrowing BTC* on Helios and using some of their BTC collateral to back that, then selling the borrowed BTC for XUSD or another stable asset. Now they have a short BTC position (owed BTC) and the proceeds in stable form. The user can then lend those stablecoins out or put them in a yield-bearing instrument on Bitcoin. The net effect: the BTC price risk cancels out – if BTC price moves, the value of their debt vs collateral offset – and they earn yields from lending the stablecoins or arbitraging interest rates.

One concrete delta-neutral example: **BTC Basis Trade** – A user deposits BTC to Helios and borrows, say, 70% of its value in BTC (essentially taking a leveraged short BTC position). They immediately sell the borrowed BTC for fiat or a fiat-pegged stablecoin. Now they have cash earning (for example) 5% in a money market, and a BTC loan on Helios paying (for example) 3% interest. Because they are short BTC vs cash, any interest they pay is like a cost of carry. If the interest on the BTC loan is lower than what they earn on the cash (or if BTC's forward price implies a lower yield), they net a profit without caring about BTC's price direction. This is similar to the classic cash-and-carry arbitrage or yield farming, but done using Helios. In traditional markets this is known as arbitraging the futures basis; in DeFi, it's an implementation of **delta-neutral lending** [9]. Helios facilitates this by providing a reliable way to borrow BTC against BTC (or against other assets) at algorithmic interest rates. Arbitragers can move in and out quickly (especially once Helios expands to faster L2s). The adaptive rates and risk parameters ensure that the system remains stable even as such strategies are executed at scale.

Low-Risk Yield for Bitcoin Holders: For an average Bitcoin holder who wants to earn yield but is afraid of losing their BTC to market swings, Helios offers a safer environment. One could deposit BTC in Helios to earn interest (yield coming from borrowers who pay interest). Since loans are over-collateralized and risk-managed dynamically, the likelihood of default is very low, meaning lenders can be confident in the safety of their principal plus interest. This is analogous to leaving money in a savings account. Additionally, Helios could integrate **insurance funds or coverage** whereby a portion of interest is allocated to an insurance pool that backstops any unexpected losses (if, say, a black swan event exceeded the model's assumptions). With such safeguards, the effective risk of loss for a lender approaches that of holding BTC outright, yet they receive an extra return. In traditional finance terms, Helios is providing a way to earn the "risk-free rate" on Bitcoin (noting that nothing is truly risk-free, but it's minimized). This concept did not exist on Bitcoin before – Bitcoin holders either kept coins idle (0% yield) or took on significant platform risk with centralized lenders. Helios changes that by offering a decentralized, smart contract-governed yield that is more sustainable and transparent.

Efficient Tax Management: Another aspect of capital efficiency is how much of the returns one keeps after taxes and fees. As discussed, Helios's loan structure enables a common strategy:

borrow against appreciated BTC instead of selling it, to avoid realizing a taxable gain [5]. Many crypto investors use this strategy to defer taxes indefinitely – “Buy, Borrow, Die” as it’s sometimes called. Helios is an ideal platform for this: a user can lock up their BTC, get a loan in a stablecoin or fiat proxy, and use that for liquidity needs without selling their BTC. Loans are generally not treated as taxable income, and using crypto as collateral for a loan is tax-free in many jurisdictions[8]. By automating this and making it safer (through adaptive risk management preventing surprise liquidations), Helios helps users hold their long-term BTC positions while still accessing cash or stable assets. Furthermore, the **partial collateral swap** feature can be used for tax-loss harvesting or rebasing one’s cost basis without a full taxable event. For example, if a user’s BTC collateral has significantly appreciated, swapping a portion of it to a stablecoin within the loan context might not trigger capital gains tax immediately, yet it effectively locks in that value – the user could later swap back in a lower price environment to increase BTC holdings, in a manner similar to taking profits but via the loan structure. These nuanced strategies mean users can enhance their **after-tax returns**, which is a vital component of true capital efficiency.

Institutional and Cross-Market Arbitrage: Professional traders can utilize Helios to arbitrage between Bitcoin markets and Ethereum DeFi markets. Since Helios allows borrowing in BTC on Bitcoin, and Aave (for instance) allows borrowing in ETH or stablecoins on Ethereum, a fund could take a position where they borrow BTC cheap on Helios, convert to WBTC and lend on Ethereum if rates differ, or vice versa, until rates equilibrate. This cross-market efficiency will drive liquidity to Helios if it offers better terms. Thanks to the absence of bridge risk on Helios, institutions might prefer to park liquidity in Helios and arbitrage outwards using small representative hedges rather than move large sums through bridges. Over time, this should lead to **rate convergence** where Helios’s rates reflect the broader crypto credit market with an added premium for being Bitcoin-native. That premium is likely reduced risk, which can attract large-scale capital (potentially even from traditional finance looking for yield on BTC without exchange risk).

In summary, Helios not only preserves capital through strong risk management, but it also **puts capital to work more effectively**. Users can achieve high utilization, implement sophisticated yield strategies, and optimize their financial outcomes (including tax considerations) in ways not previously possible on Bitcoin. By enabling delta-neutral and arbitrage strategies [9], Helios also likely improves market efficiency: any mispricing between Bitcoin’s money markets and others will be arbitraged away, leading to more stable and predictable rates for all participants. All these factors contribute to Helios’s vision of a Bitcoin-centric financial system where holding BTC is not just speculative but also productive, generating yield with minimal additional risk.

8. Regulatory Alignment and Institutional Design

As decentralized lending matures, regulatory considerations become increasingly important. Helios is designed with a proactive approach to compliance and institutional requirements, seeking to align with emerging regulations without sacrificing decentralization. In this section, we discuss how Helios addresses regulatory and institutional concerns, including KYC/AML, taxation, reporting, and governance, and how its design can incorporate or complement compliance measures.

Permissionless Core with Optional Permissioned Pools: Helios's base protocol is permissionless – anyone with BTC can participate as a lender or borrower without undergoing identity verification. This open access is fundamental to DeFi's ethos. However, recognizing the needs of institutional participants, Helios can support **permissioned pool instances** or overlays that require KYC (Know Your Customer) verification, similar to Aave Arc's approach [10]. In a permissioned deployment of Helios, only whitelisted addresses (belonging to verified institutions) could supply or borrow funds. This would be a separate pool isolated from the main pool, or implemented via smart contract checks on the main pool if separation is not needed. By structuring this way, Helios ensures that it can cater to regulated entities (like banks, funds) who are mandated to only interact with KYC'd counterparts, **without compromising** the permissionless nature of the public pool. All the benefits of Helios (Bitcoin-native, adaptive risk) remain in the permissioned version, but with an additional identity layer managed by accredited whitelisters. The architecture (with validators and smart contracts) allows such constraints to be added as a module – for instance, validators could be required (by governance decision) to enforce that addresses have a signed credential from a KYC oracle before processing a transaction in that pool. This aligns with global regulatory trends: jurisdictions are increasingly looking at DeFi and discussing whether certain activities should only be accessible to verified users. Helios's flexibility means it can operate a compliant walled garden for those who need it, while still providing an open garden for the broader community.

AML and Transaction Monitoring: Even in the permissionless pool, Helios can integrate tools for AML (Anti-Money Laundering) risk mitigation. For example, Helios could utilize blockchain analytics services to monitor deposits for known illicit sources (such as coins associated with sanctioned addresses or hacks). If a deposit is flagged (say a large amount coming from a blacklisted address), Helios's governance or automated circuit breakers could **pause interactions** with that address or quarantine those funds. While Helios's smart contracts themselves are automated and cannot outright freeze specific users (consistent with decentralization), front-end interfaces and alert systems could inform validators to not service certain requests if they violate sanctions or laws. The protocol could also be configured such that any rewards or interest destined for a blacklisted address are diverted to an escrow until cleared. These measures would help Helios align with regulations like the FATF Travel Rule indirectly, by making it difficult for bad actors to use the platform, thereby preserving the platform's reputation among regulators.

Regulatory Classification and Compliance: Helios is aware of the regulatory frameworks like MiCA (Markets in Crypto-Assets Regulation in the EU) and guidance from bodies like the BIS and IOSCO. MiCA, for instance, will impose rules on crypto-asset service providers and possibly on significant DeFi protocols. Helios's fully collateralized model might avoid classification as a bank-like entity (since it doesn't issue fractional reserves or its own redeemable token), but if Helios were deemed a provider of crypto credit, it might need to register or comply with certain prudential requirements. Fortunately, Helios's robust risk framework could facilitate compliance: the **convex optimization model** can be seen as an algorithmic internal control system that a regulator might actually favor, since it systematically limits risk. If required, Helios could publish regular reports of its risk metrics, similar to a bank's Basel accords reporting. The transparency of on-chain data means metrics like total value locked, collateralization ratios, and liquidity can be reported in real-time. Institutions using Helios will need these data for their own regulators and auditors, and Helios can provide dashboards or APIs for that purpose.

Moreover, **stress testing** can be done openly given the model in Appendix – regulators or independent auditors could simulate extreme scenarios on Helios's parameters to verify the protocol would remain solvent. This kind of provable safety is a strong argument in favor of letting institutions engage with Helios. Traditional banks might even use Helios if it's demonstrably safer and more transparent than off-chain lending.

Institutional Onboarding Considerations: For institutional adoption, certain design considerations have been included in Helios:

- *Accounting and Reporting:* Institutional users require clear records for every transaction for accounting. Helios can provide an **audit trail** for each loan – essentially the Bitcoin transactions themselves serve as an immutable record, and Helios can tag each loan with an ID for reference. Appendices in each loan smart contract can store metadata (like a hash of off-chain documents or an account reference) to help institutions map on-chain activity to their books. Because everything is on Bitcoin's ledger, it's actually easier to verify and trace than on multi-hop cross-chain setups.

- *Tax Compliance Tools:* Helios could offer an optional feature where it generates reports of interest paid/earned in a tax-friendly format. For instance, an institutional borrower could get a monthly statement: "X BTC interest paid on loan #123, equivalent to Y USD on that day's rate," which simplifies their tax and interest expense accounting. Likewise, lenders get statements of interest earned. These kinds of features, while not on-chain, can be provided by the Helios ecosystem (perhaps by third-party analytic providers) to improve the institutional user experience.

- *Governance and Legal Structure:* While Helios is a decentralized protocol, there may be a DAO governing it, possibly with a governance token (though not strictly required for functionality). If a DAO is involved, ensuring that governance does not inadvertently create a centralized "issuer" of the financial product is key. Helios's parameter adjustments are mostly automated, limiting the DAO's role to broader decisions or upgrades. This can help argue that Helios is more of a software service or tool than a financial intermediary, possibly placing it under more lenient regulatory categories. Some jurisdictions might still consider the governance participants as providing a financial service. To address this, the DAO could be structured legally (some DAOs incorporate or establish foundations). Such steps, while external to the protocol itself, indicate a readiness to engage with regulators proactively.

Institutional Partnerships: Helios could partner with custodial firms for institutional access. For example, an institutional user could custody their BTC with a provider that integrates with Helios (possibly by running a Helios validator or providing a gateway). This is similar to Aave Arc's model where Fireblocks was the whitelister and custodian for institutions [10]. Through these partnerships, institutions get a familiar interface and security while Helios operates in the background. This dual approach (decentralized protocol + institutional interface) means that Helios can cater to large players (who might bring in hundreds of millions in BTC liquidity) in a compliant way. The impact of institutional involvement could be huge: unlocking even a fraction of institutional BTC holdings (currently mostly held in cold storage by funds) into DeFi could create a *trillion-dollar opportunity* over the next decade [10].

Regulatory Future-Proofing: Helios is built to be flexible. If laws change, the protocol's governance can adjust certain features (for instance, if a certain jurisdiction bans non-KYC lending, Helios could restrict frontend access or encourage those users to migrate to a permissioned instance). Because the core is decentralized and lives on Bitcoin, it cannot be shut down easily, but it can adapt to the environment to ensure long-term viability. One important consideration is **consumer protection**. Regulators will scrutinize how liquidations occur, whether users are treated fairly, etc. Helios's partial liquidation and adaptive measures actually align with protecting users from sudden failure. No user is ever margin-called overnight without warning; the dynamic adjustments act as early warnings. Helios could also implement a feature where users can opt in to automatic deleveraging (e.g. convert some collateral to stablecoin if risk gets too high) to avoid liquidation – effectively a stop-loss. These are user-protection features that show Helios is trying to be prudent. We can document these mechanisms and perhaps even seek certification or code audits that attest to the safety measures in place.

Decentralized Governance and Audits: Finally, Helios will maintain rigorous **security audits and formal verification** of its smart contracts, given the high stakes of managing potentially large BTC pools. From a regulatory standpoint, demonstrating that the protocol has been audited by reputable firms and has no backdoors or vulnerabilities is essential (any major exploit would not only harm users but draw regulatory ire). The **open-source nature** of Helios allows continuous peer review by the community. Institutional validators may also run their own risk models in parallel to cross-check Helios's algorithm outputs, adding redundancy. All changes to the protocol can be subjected to a time lock and community review to ensure nothing malicious is introduced (which helps address regulators' concerns about governance attacks or sudden changes harming users).

Helios is architected not just as a technological solution, but as a system that can **coexist with regulatory frameworks**. By providing both permissionless and permissioned options, integrating KYC/AML where needed, and aligning its risk practices with prudential standards, Helios aims to be *"institution-ready."* The protocol seeks to demonstrate that decentralization is not at odds with compliance – in fact, Helios can offer a level of transparency and risk management rigor that traditional lenders (and regulators) can appreciate. As regulation evolves, Helios's governance and development community will continue to engage and adjust, ensuring that the protocol can thrive in the long run, bringing trustless Bitcoin lending to mainstream finance in a compliant manner.

9. Future Work

Helios Finance introduces adaptive risk optimization for Bitcoin-native lending, but several limitations still remain that open promising avenues for future research and development. The following directions will be explored in future to address current shortfalls and advance the field:

Improved Stochastic Modeling for Bitcoin Price and Network Risk

- Develop and integrate heavy-tailed and regime-switching models for Bitcoin price movements, moving beyond the current geometric Brownian motion assumption. This should include extreme value theory to better capture the fat-tailed risk profile of BTC.

- Model the dynamic and potentially endogenous relationship between mempool congestion and price volatility, especially during periods of market stress. Techniques such as state-space models or Hawkes processes could help capture feedback loops and joint risk amplification.

Formal Analysis and Verification of Risk Model Properties

- Provide rigorous mathematical proofs of the convexity of expected bad debt (EBD) and other key risk metrics under more realistic, non-normal price distributions.

- Extend the theoretical framework to analyze the impact of piecewise-linear approximations used for on-chain implementation, quantifying any potential loss in accuracy or safety.

Comprehensive Empirical Validation

- Conduct extensive backtesting of the risk model using historical Bitcoin price and mempool data, including stress periods (e.g., 2020 market crash), to evaluate real-world performance.

- Benchmark Helios against existing protocols such as Aave and Compound using quantitative metrics like risk-adjusted returns, liquidation rates, and capital efficiency.

- Implement systematic stress testing, including Monte Carlo simulations across a range of parameter configurations, to assess protocol resilience under extreme scenarios.

Technical Architecture and Security

- Publish detailed technical documentation and security analysis of the Helios Finance's operations on MIDL execution layer, including formal threat modeling and validation of the validator and threshold signature scheme.

- Explore the use of advanced cryptographic techniques (e.g., zero-knowledge proofs) for validator integrity and oracle data reliability, especially for price and mempool feeds.

Refinement of Liquidation and Oracle Mechanisms

- Undertake a formal game-theoretic analysis of liquidation incentives and strategic behavior among liquidators, particularly under high congestion or volatile markets.

- Analyze and mitigate oracle manipulation risks, delays, and failures, possibly through decentralized oracle designs and robust incentive mechanisms.

Institutional and Regulatory Considerations

- Investigate the design of privacy-preserving KYC/AML solutions compatible with Bitcoin's architecture, using cryptographic attestation or zero-knowledge credentials.

- Model and simulate capital adequacy requirements and reporting standards for decentralized lending pools under evolving regulatory frameworks.

Pursuing these research directions will address current limitations and will establish Helios as both a robust academic reference and a practical, secure protocol for Bitcoin-native decentralized lending.

10. Conclusion

This paper presented Helios, a novel **Bitcoin-native lending protocol** that blends the security of the Bitcoin network with the sophistication of modern DeFi lending. We began by highlighting the gap in the DeFi ecosystem for directly utilizing Bitcoin’s liquidity without relying on insecure wrapped tokens or centralized bridges. Helios addresses this gap by operating directly on Bitcoin’s mainnet via the MIDL execution layer, thereby enabling fully trustless BTC-denominated lending and borrowing. This design fundamentally differentiates Helios from Ethereum-based counterparts, eliminating the systemic risks associated with bridging BTC and expanding financial access to Bitcoin holders in a decentralized manner.

We detailed Helios’s **system architecture**, showing how it leverages a delegated Proof-of-Stake validator network and threshold signature wallets to ensure that user funds remain secure and under user control at all times. The architecture demonstrates that advanced smart contracts can be executed with Bitcoin as the settlement layer, achieving one-block finality and one-step user interactions. Core features such as self-custody, one-step deposit/withdrawal, and partial collateral swaps offer users both security and flexibility. These features, combined with Bitcoin’s ubiquity, position Helios as a user-friendly yet powerful protocol: Bitcoin holders can earn yield or obtain liquidity in a single transaction without leaving their preferred blockchain, and without surrendering control of their private keys.

A key contribution of Helios is its **adaptive risk management model**. By introducing mempool-aware and volatility-aware adjustments to loan parameters, Helios dynamically maintains stability in the face of changing conditions. We described how this model uses convex optimization techniques to formalize risk constraints and automatically tune parameters like LTV and liquidation bonuses. This represents an advancement in DeFi risk management – an algorithmic, provably optimal controller that can react within minutes to market stress, something that manual governance processes cannot achieve. The mathematical framework, summarized in the main text and elaborated in the Appendix, demonstrates that Helios can mathematically guarantee certain safety margins while maximizing capital efficiency. The avoidance of machine learning in favor of an explainable, audit-able optimization framework will engender greater trust and ease of verification for users, auditors, and regulators alike.

Helios also introduces improvements in **capital efficiency** and strategy availability. We discussed how higher allowable LTVs (during calm periods) and the ability to deploy delta-neutral strategies make Helios attractive for both retail and institutional users. Bitcoin holders can effectively use Helios to monetize their holdings (for example, via tax-efficient loans) or to earn passive income with mitigated risk. Meanwhile, traders and liquidity providers can arbitrage between Helios and other markets, ensuring that capital flows to where it’s most productive. By enhancing Bitcoin’s role in DeFi, Helios could increase the overall efficiency of crypto markets – Bitcoin will no longer sit idle or only serve as collateral on distant chains; it becomes an active asset fueling a native financial market.

On the **regulatory and institutional front**, Helios is forward-thinking. The design allows for permissioned environments and compliance integrations without altering the protocol’s core operations. This dual approach can accelerate institutional adoption, as evidenced by analogous efforts like Aave Arc which have shown significant interest from regulated entities [10]. We believe that by providing a robust risk framework and transparency, Helios can serve as a blueprint for a DeFi protocol that regulators might be more inclined to approve or at least tolerate, as it addresses many of their concerns (such as custody, AML, systemic risk). In time, Helios could become part of the financial infrastructure that bridges traditional finance and crypto in a compliant manner, for example by enabling banks to offer Bitcoin-backed loans via Helios or by having Helios's rates serve as a benchmark for Bitcoin's time value.

In conclusion, Helios represents a significant step in the evolution of decentralized finance on the Bitcoin network. It contributes a reference architecture for trustless Bitcoin smart contracts operating in harmony with Bitcoin's base layer, introduces an academically rigorous risk management approach in DeFi, and brings new levels of efficiency and compliance readiness to crypto lending. The protocol's formal foundations and practical innovations make it well-suited for peer review by both the research community and industry practitioners. By unlocking Bitcoin's potential in DeFi, Helios aims to foster a more inclusive, secure, and efficient financial system. Future work will include further testing of the convex optimization model under various scenarios, security audits of the smart contracts, and deployment on test networks (and eventually mainnet) to validate performance. Additionally, research into incorporating more complex financial instruments (such as options for hedging or native stablecoin designs) into the Helios ecosystem could be explored. Helios opens the door for Bitcoin to not only be "digital gold" but also a backbone of decentralized credit markets. We invite the community to engage with Helios's development, provide feedback on our approach, and collaborate in refining this protocol to safely bring Bitcoin into the forefront of decentralized finance.

Appendix: Convex Risk Optimization Framework for Helios

1. Definitions and Assumptions

To formalize the trade-off between capital efficiency and systemic risk on a BTC-native lending platform, we first define the key variables and assumptions:

Loan-to-Value (LTV)

The ratio of loan amount to the current value of collateral.

If V is the collateral value and L is the loan principal, then:

LTV = L / V

It ranges from 0 to 1, with higher LTV indicating more leverage (less collateral buffer). A higher LTV increases usage of funds but also elevates default risk. We assume each asset has a maximum allowed LTV set by the protocol.

Liquidation Penalty (LP)

A fee (or bonus to liquidators) applied during liquidation, expressed as a fraction of collateral seized.

For example, if **LP = 10%**, a liquidator may purchase collateral at 90% of market price (a 10% discount). A higher LP incentivizes faster liquidations by increasing liquidator profit, but also means the protocol may recover slightly less debt per unit of collateral (since a portion goes to the liquidator as a penalty to the borrower). LP must be non-negative and is typically bounded by an upper limit (e.g., 30%) to avoid excessive borrower loss. We denote LP as a decimal (e.g., 0.10 for 10%).

Mempool Congestion (M)

A real-time metric of Bitcoin network congestion, reflecting how slow or costly transactions are to confirm.

High M implies longer delays for liquidation transactions to be mined, increasing the window in which collateral value can drop further before liquidation executes. We treat M as an exogenous parameter (measured via an oracle) that increases the expected liquidation delay:

T = T(M)

For instance, in high congestion (large M), liquidation might be delayed by several blocks, heightening the risk of collateral value decline. In the model, M enters as a risk factor that amplifies default probability.

Price Volatility (σ)

The asset's price volatility, measured over a relevant horizon (e.g., annualized or short-term realized volatility). A higher σ means the asset's value is more prone to large swings in short timeframes, increasing the likelihood of sudden drops.

We assume σ can be observed or estimated in real-time (via oracles) and treat it as an input. Volatility directly influences the probability that collateral value falls below a critical threshold before liquidation can occur.

Default Risk (D)

The probability that a loan becomes undercollateralized beyond recovery—that is, the collateral value after a delay falls short of covering the loan plus penalties.

We define the default event for a single loan as:

Default Event = { V_final < (1 + LP) * L }

Where:

- `V_final` is the collateral value at the time of delayed liquidation
- L is the loan principal
- LP is the liquidation penalty

In this case, even full liquidation yields insufficient funds to repay the debt.

D (as a function of LTV, LP, M, and σ) denotes the probability of this event:

$D = D(LTV, LP, M, \sigma)$

Increasing `LTV` raises `D` (less cushion against price drops), as do increases in `M` (slower liquidation) and σ (higher volatility). A higher `LP` generally reduces `D` by encouraging quicker liquidations, though it may increase the loss severity if default occurs.

Expected Bad Debt (EBD)

The expected value of unrecouped loan amount in the event of default, which is minimized as the primary risk metric.

It represents the protocol's expected loss due to defaults, accounting for both:

- Probability of default
- Severity of loss given default

For a single loan with normalized initial collateral value (V = 1), EBD is:

EBD(LTV, LP; M, σ) = Expected value of max{ 0, L - V_final / (1 + LP) }

Where:

- `L = LTV` (loan as a fraction of initial collateral)
- `V_final` is the random collateral value at liquidation time

The term inside the max is the shortfall:
Shortfall = L - V_final / (1 + LP)

Since a liquidator paying off L can claim collateral worth `L * (1 + LP)`, only `1 / (1 + LP)` of collateral value per unit debt goes to debt repayment.

- If `V_final >= (1 + LP) * L`, the shortfall is zero (no bad debt)
- If `V_final < (1 + LP) * L`, the shortfall is positive

EBD integrates over the distribution of `V_final` (which depends on σ and `M`).

In simpler terms:

EBD ≈ D × Loss Given Default

Reducing EBD makes the protocol more robust to insolvency.

Capital Efficiency (CE)

A measure of how effectively the protocol's collateral is utilized for lending. We aim to maximize CE secondarily.

Higher CE means borrowers can draw more loans from the same collateral, improving protocol utility. While CE can be measured in several ways (e.g., total loan volume divided by total collateral), we use **LTV** as a proxy for CE, because a higher LTV allows users to borrow a larger share of their collateral value.

In effect, maximizing CE means increasing LTV—as long as the system remains safe.

Note: Extremely high LTVs may reduce actual borrowing, as users avoid risky leverage. So practical usage may peak before theoretical maximum LTV.

Assumptions

- LTV is a continuous variable in the range: **$0 \le \text{LTV} < 1$** (i.e., strictly less than 1 to ensure overcollateralization)
- LP is bounded: **$0 \le \text{LP} \le \text{LP\_max}$**, where LP_max is typically 0.3 to 0.5
- We consider a **single collateral asset (BTC)** in base modeling. Extension to multiple assets is addressed later.
- Collateral price follows a **stochastic process** (e.g., geometric Brownian motion or a heavy-tailed model)
- The distribution of `V_final` is influenced by σ and `M`:
    - Higher σ increases the downside tail
    - Higher `M` increases expected liquidation delay `T(M)`
    - Together, they worsen expected price decline over the liquidation window

- Therefore, higher σ or `M` → higher EBD, for fixed LTV and LP

These dynamics are encoded in the optimization model.

Finally, we assume the protocol can **dynamically adjust LTV and LP** (via governance or algorithmic policy) in response to changes in `M` and σ.

Goal: Choose LTV and LP values that optimally balance risk (EBD) and capital efficiency (LTV), under prevailing market and network conditions.

2. Convex Optimization Objective Function

We formulate a convex optimization problem to capture the trade-off between minimizing Expected Bad Debt (EBD) and maximizing Capital Efficiency (CE). In line with the protocol's priorities, minimizing risk (EBD) is the **primary objective**, and capital efficiency is **secondary**—meaning we do not significantly increase risk for marginal gains in efficiency. This aligns with risk management practices in DeFi protocols like Aave, which tune parameters to maintain insolvency risk at acceptable levels while enabling high utilization.

Objective Function Formulation

We define a single objective `F` that combines EBD and CE with appropriate weighting to ensure convexity and to prioritize risk reduction. A convenient form is a weighted sum:

Minimize over LTV and LP:
$F(LTV, LP; M, \sigma) = EBD(LTV, LP; M, \sigma) + \lambda \cdot \Phi(LTV)$

Where:

- `Φ(LTV)` is a penalty function that increases when capital efficiency is lower
- `λ > 0` is a small weighting parameter balancing the two objectives

By minimizing `F`, the solver will reduce EBD while also avoiding inefficient LTV values. Since EBD is the priority, λ is chosen to be small—representing a lexicographic preference:
first minimize risk, then prefer better capital efficiency among safe solutions.

Choice of Penalty Function $\Phi(LTV)$

To maintain convexity, `Φ(LTV)` must be a convex, **decreasing** function—so that lower LTV (i.e., lower CE) incurs a higher cost.

A simple choice is a **linear penalty** on unused collateral capacity. For instance, if `LTV_max` is an upper bound (say 1.0), we could define:

$\Phi(LTV) = -LTV$

Then the objective becomes:
$F = EBD(LTV, LP; M, \sigma) - \lambda \cdot LTV$

This formulation rewards higher LTV values (i.e., greater efficiency). However, we must note:

- `EBD(LTV)` is an increasing **convex** function (since risk rises with LTV)
- $-\lambda \cdot LTV$ is a **concave** function (negative linear)

And the **sum of a convex and concave function is not guaranteed to be convex**. To preserve overall convexity, we instead define `Φ(LTV)` as:

- $\Phi(LTV) = (LTV_{max} - LTV)$ → linear (both convex and concave), or
- $\Phi(LTV) = (LTV_{max} - LTV)^2$ → quadratic (strictly convex)

A linear penalty effectively shifts the function by a constant (irrelevant to optimization) and still behaves like $-LTV$ up to a constant. So:

We abuse notation and interpret $\Phi(LTV) \approx -LTV$ for intuition, while using a convex surrogate (e.g., small quadratic or piecewise-linear approximation) in implementation.

In summary, the working form of the objective becomes:

Minimize over LTV and LP:
$F = EBD(LTV, LP; M, \sigma) - \lambda \cdot LTV$

With λ chosen small enough that insolvency risk dominates.

This approach parallels Gauntlet's tri-objective framework (minimizing insolvency, minimizing liquidations, and maximizing borrow usage). In our simplified two-term formulation, EBD carries **higher weight** than CE.

Convexity

As we will show in Section 4, **EBD(LTV, LP)** is a convex function in the decision variables under standard modeling assumptions. Conceptually, EBD behaves like a **convex risk measure**, similar to **Expected Shortfall (CVaR)**, which is known to be coherent and convex.

Thus:

- `F = EBD − λ · LTV`
  is the sum of a **convex** function (EBD) and an **affine** function (–λ · LTV),
  and therefore is **convex overall**

If strict convexity is required (e.g., to avoid flat regions), we can add a tiny **quadratic regularizer**.

This ensures:

- No local minima
- Reliable convergence using standard optimization solvers
- Suitability for both on-chain and off-chain computation

Constraints

The optimization is subject to several constraints reflecting protocol safety and usability limits:

1. **LTV Bounds:**
   0 ≤ LTV ≤ LTV_max
   Where LTV_max is an absolute upper bound (e.g., 0.9).
   In practice, we may set LTV_max = 0.8 or 0.85 for BTC to provide a risk buffer.
   This also bounds CE from above.

2. **LP Bounds:**
   0 ≤ LP ≤ LP_max
   LP_max could be 0.2 (i.e., 20%) or higher in extreme scenarios.
   High LP values can deter borrowers or create cascade liquidations.

3. **Minimum LP:**
   We may impose **$LP \geq LP\_min > 0$**, e.g., $LP\_min = 0.05$
   This ensures liquidators have sufficient incentive.
   Aave typically uses a "liquidation bonus" around 5%–10%.

4. **(Optional) Liquidation Threshold LT:**
   If LT is separate from LTV, then:
   $LT \geq LTV$, possibly defined as:
   $LT = LTV + buffer$
   But for simplicity, we assume **LTV acts as the threshold**, as in MakerDAO-like systems.
   Our EBD already incorporates LP, which is equivalent to the penalty buffer in LT.

   In fact, the condition:
   $V\_final < (1 + LP) * L$
   is equivalent to:
   $V\_final / L < 1 + LP$,
   i.e., a collateral ratio less than the required coverage. So:
   $1 / (1 + LP)$ plays a similar role to LTV_max.

5. **Non-Negativity of EBD:**
   EBD is always ≥ 0 by construction (expectation of a non-negative shortfall)

6. **Optional Risk Cap:**
   Add:
   $EBD \leq \varepsilon$
   to enforce a max acceptable risk level (e.g., set by governance).
   If active, this fixes the maximum allowable risk, and the optimizer simply maximizes LTV until hitting that limit.

Outcome

Solving this optimization yields the **optimal values of LTV and LP**—denoted LTV* and LP*—as functions of the current mempool congestion `M` and volatility σ.

The solution:

- Minimizes expected protocol loss (EBD)
- Enables the **maximum borrowing** that is **still safe** under given market/network conditions

In the next sections, we present **two practical formulations** of this optimization problem:

- **(A)** A simplified static model using piecewise-linear approximations — suitable for on-chain smart contract implementation
- **(B)** A dynamic real-time model — suitable for off-chain solvers with live oracle inputs for volatility and congestion

3. Formulation A: Simplified On-Chain Model (Piecewise Linear)

On-chain environments—especially those on Bitcoin or EVM-based systems—have limited computational resources and favor deterministic, simple logic. To implement risk optimization within a smart contract, we propose a **simplified convex model** using **piecewise-linear functions**. This approach approximates key relationships in a tractable, on-chain friendly manner (e.g., via Solidity or parameter lookup tables).

3.1 Convex Piecewise-Linear Approximation

We approximate the EBD function, which may be nonlinear in LTV and LP, using a **piecewise-linear** function. Convexity is preserved by taking the **upper envelope** of linear segments.

Suppose we pre-compute or estimate EBD for a range of LTV values at typical levels of mempool congestion `M` and volatility σ. Let the points on the EBD-vs-LTV curve be:

(x_j, y_j) for j in some index set.

Then we construct linear segments between these points. The piecewise-linear approximation of EBD becomes:

EBD_hat(LTV) = max over j of { a_j * LTV + b_j }

Where `a_j` and `b_j` are the slope and intercept of each linear segment `j`.

This formulation ensures convexity (as the maximum of linear functions), and conservatively **over-approximates true EBD**—which is acceptable since our goal is safety. In practice, only a few segments are needed because EBD remains flat at low LTV and increases rapidly near unsafe levels. Likewise, the nonlinear impact of LP can be approximated by:

- Piecewise-linear terms
- Or a small set of fixed discrete values for LP (e.g., 5%, 10%, 15%)

3.2 Linear Constraints for Risk

This optimization can now be written as a **Linear Program (LP)** suitable for on-chain execution.

Introduce an auxiliary variable Z to represent EBD. Impose linear constraints:

Z ≥ a_j * LTV + b_j for all segment indices `j`.

Then minimize the linear objective:

Z – λ * LTV

Under these constraints, the optimizer drives Z to the smallest feasible value—equal to the piecewise-linear EBD approximation at the optimal LTV.

Because linear programming is convex, this model is **safe, efficient**, and **analytically solvable** for small instances. Often, the optimal point occurs at a **"kink"** (where two linear segments meet), so the protocol can pre-compute optimal candidates—checking only:

- Segment breakpoints
- Parameter boundaries

—rather than performing iterative optimization on-chain.

3.3 Discretized Solution Space

To further simplify on-chain logic, we can **discretize the decision variables**:

- Allow only certain LTV values: e.g., {50%, 60%, 70%, 80%}
- Allow only certain LP values: e.g., {5%, 10%, 15%}

Then, for each `(LTV, LP)` combination, we:

1. Pre-compute approximate EBD or full objective
2. Choose the one that minimizes **F = EBD – λ * LTV**

This becomes a simple lookup or series of conditional branches in the contract. It effectively divides the risk model into **regions**, and each region has a predefined optimal.

This also allows small rule trees such as:

> *"If volatility is low and mempool is normal, use LTV = 0.80 and LP = 5%. If volatility or congestion is high, drop to LTV = 0.60 and LP = 10%."*

We formalize this logic below using model insights.

3.4 Incorporating M and σ On-Chain

Real-time optimization on-chain is impractical, but we can **parameterize the strategy** using mempool congestion `M` and volatility σ via **piecewise-linear logic**.

Define thresholds:

- `M_low` and `M_high` for congestion levels
- `σ_low` and `σ_high` for volatility levels

The policy becomes:

- **If M < M_low and σ < σ_low:**
  Use **maximum LTV** and **low LP** for efficiency and user-friendliness
- **If M or σ is moderate:**
  Use **medium LTV and LP**
- **If both M and σ are high:**
  Use **low LTV and high LP** to prioritize safety

This logic can be implemented using simple `if-else` comparisons and assignments. The smart contract is essentially applying a **pre-optimized decision tree** derived from more complex off-chain analysis (Formulation B).

3.5 Example (On-Chain Rule)

Suppose off-chain analysis reveals:

- **Low congestion and volatility:**
  Optimal → LTV = 80%, LP = 5%
- **High congestion or volatility:**
  Optimal → LTV = 60%, LP = 10%
- **Extreme stress (both M and σ high):**
  Optimal → LTV = 40%, LP = 15%

Then the smart contract could implement:

```
if (M < M_low && sigma < sigma_low) {

   LTV_allowed = 0.80;  LP = 0.05;

} else if (M < M_high && sigma < sigma_high) {

   LTV_allowed = 0.60;  LP = 0.10;

} else {

   LTV_allowed = 0.40;  LP = 0.15;

}
```

Each branch corresponds to a **linear region** in the risk model. This is a **piecewise-constant policy**, which is a special case of a piecewise-linear formulation.

The thresholds `M_low`, `M_high`, etc. are determined by identifying inflection points where risk significantly increases.

3.6 Ensuring Convexity and Feasibility

Even though the decision space is discrete, all choices come from the **convex hull of safe configurations**. The piecewise-linear structure guarantees that:

- Interpolating between defined points cannot yield a better (i.e., lower) objective
- All constraints (e.g., bounds on LTV and LP) are inherently respected

Since all logic is **linear** and **branch-free** within each decision region, it's easy to formally verify the smart contract's behavior—critical for protocol security and audits.

This on-chain formulation provides a **simple, convex, deterministic** strategy for setting LTV and LP based on observable conditions like mempool congestion and volatility. Key advantages:

- Ensures **predictability** and **safety** (never exceeds risk constraints)
- Enables **fast, gas-efficient** parameter updates
- Encodes off-chain insights into an on-chain ruleset that remains robust and auditable

Next, we describe the more sophisticated off-chain optimization model (Formulation B) that underpins this strategy and allows fine-grained calibration across real-time market states.

4. Formulation B: Full Dynamic Off-Chain Model (Continuous, Time-Varying Inputs)

For precision and adaptability, Helios employs a comprehensive off-chain optimization model that continuously ingests real-time data—such as mempool congestion and market volatility—and recalculates optimal risk parameters. This **dynamic model** can be run on a server or by an off-chain oracle, and the resulting optimal values (LTV and LP) are then fed into the protocol, either automatically or through expedited governance.

We formalize this approach as a **convex dynamic optimization problem**.

4.1 Time-Varying Inputs

We treat mempool congestion and volatility as **time-dependent functions**:

- **M(t)**: mempool congestion at time `t`
- **σ(t)**: volatility at time `t`

At each time `t`, we solve the convex program:

Minimize over LTV(t), LP(t):
F(LTV, LP; M(t), σ(t)) = EBD(LTV, LP; M(t), σ(t)) – λ · LTV

The constraints are the same as in previous formulations. Since `M` and σ vary with market conditions, the optimizer's output—denoted LTV*(t), LP*(t)—is a function of the current state.

Here, `M` and σ act as **exogenous variables**, while `LTV` and `LP` are **control variables**, re-optimized whenever external conditions shift significantly.

4.2 Stochastic Dynamic Formulation

The framework can be extended into a **stochastic control problem** if future values of `M` and σ are forecasted. However, a simpler, more tractable method is **myopic (greedy) optimization**:

- At each step, assume current `M` and σ persist for the near term
- Solve accordingly and repeat in the next time step

This creates a **receding-horizon control system** that adjusts parameters iteratively. Since protocol parameters update faster than users can change their loans (or markets can shift drastically), this quasi-static approach is effective.

4.3 Modeling EBD More Precisely

Off-chain computation allows a **more accurate model** of EBD.

Assume the collateral price `P_t` follows a stochastic process—e.g., **geometric Brownian motion** with volatility σ.

When liquidation is triggered, the time to liquidation is:

$T = T_0 + f(M)$

Where:

- T_0 = base latency (e.g., one block)
- `f(M)` = additional delay due to mempool congestion

We might model `f(M)` as a linear function of backlog, or use historical confirmation time percentiles.

Let:

- `Q` = quantity of collateral
- `L` = loan amount
- `P_t` = initial price of collateral
- `P_t * LTV = L / Q` (loan per unit collateral)

Then default occurs if:

$P_{t+T} < (1 / (1 + LP)) * (L / Q)$

Rearranged, define the required drop fraction h as:

$h > 1 – (1 / (1 + LP)) * LTV$

Define this threshold as:

*$h = 1 – (1 / (1 + LP)) * LTV$**

Assume h follows a normal distribution with mean 0 and variance σ^2T, then:

*$D(LTV, LP; M, \sigma) \approx \Phi(–h / (\sigma \sqrt{T(M)})) = 1 – \Phi(h / (\sigma \sqrt{T(M)}))$***

Where Φ is the standard normal cumulative distribution function (CDF). This provides a **closed-form approximation** of the default probability.

We can compute **expected shortfall (EBD)** similarly, using tail expectations:

Let `X = (1 + LP) * L − V_final`
Then EBD is:

E[EBD | default] = E[X | X > 0]

This is equivalent to **Conditional Value-at-Risk (CVaR)**, a known convex risk measure with standard formulas under normality. Using this, off-chain solvers (or closed-form expressions) can precisely compute EBD.

4.4 Solving the Continuous Problem

The objective function:

F(LTV, LP) = EBD(LTV, LP; M, σ) – λ · LTV

is convex under the assumptions given (verified in Section 5). It can be solved with standard convex optimization methods:

- **Interior-point methods**
- **Gradient descent**
- **Fine-grid brute force (2D)**

Because the problem only has two variables (LTV and LP), even brute-force scanning on a fine grid is computationally feasible.

The **first-order condition** (from ∂F/∂LTV = 0) provides an analytic trade-off:

Marginal increase in EBD = λ (marginal value of CE)

This can guide closed-form derivations.

Often, in extreme scenarios, the optimal values lie at the boundaries:

- Low risk → LTV = LTV_max
- High risk → LP = LP_max or LTV = LTV_min

In more moderate environments, an interior solution exists that balances both objectives.

4.5 Adaptive Output

Once the off-chain solver returns the optimal LTV* and LP* for the current values of `M` and σ, these parameters are published to the protocol.

The **update frequency** should match the rate of change in `M` and σ:

- Mempool congestion (`M`) can spike in minutes
- Market volatility (σ) can change hourly or faster

A good cadence is:

- Run the optimization every block or every few minutes
- Update parameters only if significant deviation is observed

To avoid **oscillation and user confusion**, the system may implement:

- **Hysteresis logic**
- **Grace periods** before parameter drops
- **Gradual transitions** (e.g., tapering down LTV instead of abrupt drops)

4.6 Example (Dynamic Adjustment)

Suppose initially:

- `M` is low
- σ is moderate
- Solver outputs: **LTV = 0.75**, **LP = 0.06**

Later, σ spikes from 60% to 100% annualized and `M` exceeds 100,000 transactions.

- Risk increases significantly

- New solution: **LTV = 0.50**, **LP = 0.12**

The protocol updates accordingly. When conditions normalize:

- The solver gradually raises LTV and lowers LP again
- For example:
    - First to LTV = 0.65, LP = 0.08
    - Then back to LTV = 0.75, LP = 0.06

This is akin to a **risk throttle** that tightens during turbulence and relaxes during calm. This dynamic modulation is increasingly adopted in modern DeFi systems for adaptive risk control.

4.7 Computational Feasibility

All **heavy computation** (e.g., integrals, optimization, simulations) is conducted **off-chain**.

This allows:

- **Use of complex models**, including:
    - Heavy-tailed distributions
    - Monte Carlo simulations
    - ML-based volatility forecasting
- **No on-chain gas burden**
- **Clear, interpretable decision logic**:
  Even if machine learning aids parameter forecasting, final decisions must be explainable—grounded in volatility and congestion metrics

This transparency is key to building user trust and defending against "black-box" risk logic.

The off-chain model enables:

- High-fidelity risk optimization
- Real-time response to volatility and congestion
- Safe but flexible risk-based LTV/LP modulation
- Separation of **computation-heavy logic** from the **on-chain protocol**

This ensures **solvency and capital efficiency** are continuously maintained, adapting to real-world conditions with a mathematically sound foundation.

5. Convexity Analysis and Modeling Choices

We now explain why the previous formulations are convex and why we have chosen convex (and piecewise-linear) modeling techniques over alternatives like machine learning or heuristic-based algorithms.

5.1 Convexity of EBD

Expected Bad Debt (EBD), as defined in our model, is effectively an **expected shortfall**—a concept well-established in financial risk management. Expected shortfall (also known as **CVaR**, or Conditional Value-at-Risk) is a **coherent risk measure**, known to satisfy properties including **subadditivity** and **convexity**.

Intuitively:
If we consider two parameter sets for loans, then the **weighted average** of those two sets will not yield a higher EBD than the average of their individual EBDs. This reflects the idea that partial allocation or diversification cannot worsen tail risk beyond the linear combination.

More concretely:

- As **LTV increases**, EBD typically rises **nonlinearly and faster**.
    - Near LTV = 0, EBD is close to zero
    - Near LTV = 1, EBD rises steeply
      → This creates a **convex curve**
- As **LP increases**, EBD tends to decrease—but with **diminishing returns**.

- The EBD vs LP relationship also exhibits **convex behavior** (albeit downward-sloping)

Thus, the function:
EBD(LTV, LP) is **convex over the domain of interest**

In multidimensional analysis, the **Hessian matrix** of EBD is **positive semi-definite** (under standard assumptions). For example, under a normal approximation, EBD depends on the CDF $\Phi(h^*)$—which is a smooth, convex sigmoid-like function of LTV.

5.2 Convex Combination Objective

Our objective function includes a convex EBD term and a linear penalty on low CE (e.g., **$-\lambda \cdot$ LTV**). The **weighted sum of convex functions is convex**, so:

F(LTV, LP) = EBD(LTV, LP) – $\lambda \cdot$ LTV

is convex.

Why is this important? If we were to include **non-convex terms**—such as step functions or complex priority heuristics—we would introduce the risk of **local minima**, which could cause an automated system to select **suboptimal or unsafe parameters**.

To avoid this, we do one of the following:

- Use a **small linear penalty weight λ** to prioritize risk minimization
- Or, recast the problem as a constraint-based optimization:

Minimize EBD, subject to Capital Efficiency $\geq c$

This is known as a **convex feasibility problem** for any fixed value of c. Varying c traces out a **Pareto-efficient frontier** between EBD and CE, which is **concave** due to diminishing returns.

Our optimizer selects the point on this frontier that aligns with protocol preferences—balancing safety with efficiency.

5.3 Convexity of Constraints

All of the constraints we impose are **linear**, and therefore **convex**:

- Bounds on LTV:
  $0 \leq LTV \leq LTV\_max$

- Bounds on LP:
  $0 \leq LP \leq LP\_max$

In more complex versions, we might add **probabilistic constraints**, e.g.:

> Ensure solvency probability ≥ 99%
>
> This can be written as:
> **Pr(shortfall > 0) ≤ p_max**,
> or equivalently:
> **D(LTV, LP) ≤ p_max**

If D is increasing in LTV, this becomes:
$LTV \leq f^{-1}(p\_max)$ — again, a **convex constraint** (possibly mildly nonlinear but univariate)

We **avoid non-convex constraints** (e.g., discrete choices or step thresholds) in the core model to preserve **solvability and tractability**.

5.4 Why Convex Instead of Neural Networks or Heuristics

Convex models provide:

- **Transparency**
- **Reliability**
- **Global optimality guarantees**

In a financial engineering context, and especially for scientific or regulatory scrutiny, **interpretability matters**.

In contrast:

- A **neural net** or **heuristic** model might output some LTV/LP values, but there's no guarantee they are **safe**, **optimal**, or even **sensible**
- They are **hard to audit** and **difficult to debug** under edge conditions

- Their performance in unseen conditions can be unpredictable (e.g., non-monotonic behavior)

In a **convex model**, we can:

- Prove that **solutions are optimal**
- Run **sensitivity analyses** (e.g., “What happens to LTV* if σ increases 10%?”)
- Understand **qualitative structure** (e.g., in high risk, LTV drops to minimum; LP rises to maximum)

Additionally:

- **On-chain**, neural nets are **infeasible** (too much computation, storage, and complexity)
- Even **off-chain**, machine learning introduces **approximation error**, and may **violate constraints** unless post-processed carefully

In contrast, convex optimization **never proposes invalid or unsafe values**, and its outputs are **explainable** and **deterministic**.

5.5 Rationale for Piecewise Linearity (in On-Chain Model)

Piecewise-linear functions are chosen for on-chain implementation because they are:

- **Deterministic**
- **Efficient**
- **Gas-friendly**
- **Auditable**

Any **convex curve** (like EBD vs LTV) can be approximated arbitrarily well by **piecewise-linear segments**.

In practice, just a few segments are sufficient:

- The EBD curve is flat at low LTV
- It rises sharply near unsafe levels
  → This can be captured by 2–4 linear pieces

Using **piecewise-linear approximations**:

- Makes the optimization problem a **Linear Program (LP)**
- Allows for **constant-time computation** via `if-else` branches
- Avoids expensive operations like exponentiation or square roots
- Keeps **auditability simple** (e.g., easily check: "Is EBD at LTV = 0.70 below safe threshold?")

In contrast:

- **Neural nets** on-chain would require matrix multiplications and stored weight matrices—completely impractical
- Even off-chain, neural nets do not guarantee monotonicity or robustness in edge cases

Convex piecewise-linear models, on the other hand, **automatically fall back to conservative boundaries** when conditions deteriorate. For example:

> If volatility spikes, the model might set:
> **LTV = minimum allowed, LP = maximum allowed**

This is safe and predictable.

5.6 Robustness and Overfitting

Convex models—especially **piecewise-linear ones**—are **inherently robust**:

- They are not overfit to historical scenarios
- They yield **smooth, generalizable policies**

For example, the rule:

“Higher volatility → lower LTV”

is a **universal principle** that emerges naturally from convex risk minimization.

This is critical for handling **unknown future risks** or **black swan events**.

Example:
On Black Thursday 2020, MakerDAO suffered major losses due to:

- Extreme volatility
- Ethereum mempool congestion

A robust convex model would likely have:

- Lowered LTV preemptively
- Halted new borrows if risk exceeded feasible thresholds
- Prevented undercollateralized loans by proactively raising LP

Indeed, the Helios model could implement a **“circuit breaker”**:

- If risk metrics breach critical thresholds, pause borrowing
- Or dynamically reduce LTV to protective levels

Because convex solvers **detect infeasibility**, they naturally act conservatively when risk exceeds tolerable bounds.

Convexity in our risk model is not merely a mathematical preference—it is a **core design decision** that ensures:

- **Clarity**
- **Safety**
- **Tractability**
- **Scientific credibility**

The result is a model that:

- Can pass **peer review**
- Can be audited and verified
- Can inspire trust that Helios parameters are always set with **optimal safety and efficiency** in mind

6. Asset Quality and Collateral-Specific Adjustments

Not all collateral assets are equal. One optional extension of our framework is to adjust the optimization for **asset quality**—e.g., recognizing **Bitcoin (BTC)** as a “pristine” form of collateral with unique advantages. The model can incorporate asset-specific risk factors so that higher-quality assets are granted more favorable terms (such as higher LTV) without compromising system safety.

6.1 Quality Factor

We introduce a parameter **Q** for each asset, representing its **quality or risk adjustment factor**.

- A high-quality asset like BTC might have **$Q = 1$** (baseline)
- Riskier or less liquid assets would have **$Q < 1$**

We can modify the optimization constraints or objective using this **Q factor**. For instance, scale the **effective LTV** seen by the EBD function by **$1 / Q$**. This means that:

- If **$Q < 1$**, the risk engine treats the asset as **riskier**, effectively increasing LTV in the risk calculation and penalizing it
- If **$Q > 1$**, the system may treat the asset as **safer** than volatility alone would indicate

A practical implementation is to **cap LTV** for each asset as:

$LTV \le Q \times LTV\_max\_base$

Examples:

- For BTC:
  Q = 1, so LTV ≤ LTV_max_base (e.g., 0.80)
- For a more volatile token:
  Q = 0.5, so LTV ≤ 0.5 × 0.80 = **0.40**

This linear scaling is analogous to **risk-weighted assets** in traditional finance.

6.2 Volatility and Liquidity Considerations

The **Q value** can be a function of:

- Historical volatility (σ)
- Market capitalization
- Liquidity (depth of order books)

BTC maintains **Q = 1** because:

- It has deep global markets
- It trades 24/7
- It is relatively less volatile than small-cap tokens

While volatility is directly captured by σ, **liquidity is not explicitly modeled** in the base framework. To account for this, we adjust Q:

- Assets with thin order books → lower Q
- BTC with deep liquidity → Q = 1 or potentially higher

We can also consider **correlation** between assets. For example, Aave V3 introduced “Isolation Mode” and “E-Mode” for correlated collateral.

In Helios:

- If only BTC is used → correlation isn't a concern
- If multiple collaterals are allowed → joint risk increases, especially if they fall together in crises

Future extensions may include:

- **Multivariate EBD**, incorporating cross-asset correlations
- Or remain conservative by applying **per-asset risk buffers**

6.3 Example Adjustment

Suppose Helios supports a tokenized BTC mining stock, which is:

- More volatile than BTC
- Less liquid

We might assign **Q = 0.7**.

Even if its volatility (σ) is similar to BTC's, the optimizer would limit LTV to:

*LTV = 0.8 × 0.7 = 0.56**

Implementation options:

1. Scale volatility:
   σ_eff = σ / Q
   → This inflates perceived risk
2. Scale final result:
   Apply **Q × LTV_result** after solving

We can also adjust LP upward for riskier assets:

- Riskier assets → higher LP
- Liquidators may demand more incentive

6.4 Pristine BTC Implications

BTC’s reputation as the **most pristine collateral** (see *The Investor’s Guide to Bitcoin-Backed Lending*) supports the idea of **slightly higher LTV limits** than volatility might justify.

Why?

- 24/7 trading
- Global liquidity
- High trust and recognition among lenders

Some lending desks allow **70–80% LTV on BTC**, while limiting small-cap assets to **~50%**.

In Helios:

- A slight **Q > 1** could be justified for BTC
- Conservatively, we default to **Q = 1** to ensure safety

This adjustment reflects **real-world risk behavior** while maintaining robustness in volatile conditions.

6.5 Implementation

On-Chain (Piecewise Policy)

In the on-chain model, we can implement asset quality adjustments as **discrete settings**:

Example:

> "If collateral is WBTC, use base LTV from policy.
> If it is a low-cap token, apply a **0.5× multiplier** to LTV output."

This logic is:

- **Simple**

- **Transparent**
- **Deterministic**

Off-Chain (Optimizer)

Run the convex optimizer **separately per asset**, using:

- That asset's σ
- A common `M` (since mempool is shared across assets)
- That asset's **Q**

Formally, for each asset `i`, solve:

Minimize: EBD_i(LTV_i, LP_i)
Subject to: asset-specific constraints

If borrowers use **multiple assets** in one vault:

- Optimization becomes multivariate
- Requires handling **cross-collateral constraints** and **covariances**

To simplify, Helios may enforce **isolated vaults per asset**:

- Aave uses a "health factor" combining assets
- That requires complex optimization and correlation modeling
- Isolated vaults avoid this complexity, at the cost of cross-asset capital efficiency

Asset quality adjustments extend Helios's convex optimization framework to support **multiple types of collateral** with **differentiated risk caps**:

- BTC, being a high-quality asset, can safely support **higher LTV**
- Riskier or less liquid assets receive **conservatively lower caps**

This ensures the protocol remains **capital-efficient for top-tier assets**, while **preserving solvency** across varying market conditions.

7. Adaptive Parameter Logic: Mempool and Volatility Triggers

A crucial feature of Helios's risk framework is the **adaptive adjustment of LTV and LP** in response to mempool congestion and price volatility. This section details how the optimization model reacts dynamically as these external conditions change.

Impact of Mempool Congestion (M)

As **M** (mempool congestion) increases, the **expected time to liquidation** T also increases. This directly:

- Raises the chance of a significant price drop before liquidation
- Increases both **default probability (D)** and **Expected Bad Debt (EBD)**

To mitigate this, the optimizer typically responds by:

- **Lowering LTV** → Reduces loan size relative to collateral, increasing buffer
- **Raising LP** → Boosts liquidator incentives to act despite high gas costs or confirmation delays

This combination keeps expected loss within safe bounds.

Example:
During an extreme Bitcoin mempool backlog (e.g., scarce block space or mempool flooding as seen in March 2020), Helios may enter a **"risk-off" mode**:

- **Drastically reduced LTV limits**
- **Tighter caps on borrowing**

This mirrors protective measures in other protocols during crises (e.g., Aave's risk minimization changes during volatile events).

Conversely:
When **M drops** (i.e., fast confirmations, low fees), the urgency subsides. The model may then:

- **Increase LTV**
- **Reduce LP**

— since liquidations can now execute more efficiently and reliably.

Impact of Volatility (σ)

When market **volatility (σ)** spikes:

- Price swings become larger, even over short timeframes
- The model lowers LTV to reduce exposure during volatile periods

It may also **increase LP** to ensure liquidators are properly compensated for heightened risk.

If **σ remains high over time**, Helios will:

- **Maintain conservative LTV/LP settings**, even if capital efficiency drops
- Prioritize **system solvency over utilization**

This preemptive adjustment reflects lessons from past events. In 2021's crypto crash, prolonged high volatility triggered cascades of liquidations and bad debt across DeFi. Helios aims to **prevent such outcomes** via real-time de-risking.

Joint Effect of M and σ

The **worst-case scenario** is when both:

- **M (mempool congestion)** is high

- **σ (volatility)** is high

This occurred during **Black Thursday 2020**, when MakerDAO accrued millions in bad debt.

In such a situation, Helios responds by:

- **Pushing LTV to its minimum bound** (possibly halting all new loans)
- **Setting LP to its maximum allowed value** (maximizing liquidator incentive)

Effectively, the protocol enters **full protection mode**—analogous to a circuit breaker in TradFi.

Given the off-chain optimization model, Helios can:

- **Iterate frequently** during stress (e.g., every few blocks)
- **Progressively lower LTV** until risk returns to acceptable levels
- React much faster than traditional governance

This **adaptive feedback loop** is core to the design:
Helios doesn't rely on static rules—it **responds fluidly** to changing market risk.

When Conditions Improve

Just as the model tightens during stress, it also **loosens during calm**.

When:

- **Volatility subsides**
- **Mempool clears**

→ Helios may **scale LTV back up**, potentially above typical values.

For example, during **ultra-low volatility**, the system might permit:

- **LTV > normal thresholds**

- **Lower LP values**

This is akin to Aave’s **E-Mode**, except it’s triggered by **macro conditions** rather than asset correlation.

The benefit is a **competitive edge**:

- **High capital efficiency** when it’s safe
- **Automatic defense** when risks emerge

Liquidation Penalty Adjustments (LP)

While LTV is the primary lever, **LP is also dynamically adjusted**.

In general:

- **Risk-off:**
  LTV ↓, LP ↑
- **Risk-on:**
  LTV ↑, LP ↓

However, there's a **balance** to strike:

- **Too high an LP** deters borrowers (fear of heavy losses if liquidated)
- Optimizer increases LP **only when needed**, and **only by as much as needed**

Since LP appears in the default condition:

$(1 + LP) \times L \geq V\_final$

→ Increasing LP helps ensure liquidators get enough collateral to act, even under stress.

But due to **diminishing returns**:

- Raising LP from **5% to 10%** may significantly improve risk

- Raising from **15% to 20%** may offer little additional benefit

The optimizer captures this and avoids **over-penalizing borrowers**.

Thus, LP typically stays in the **5–15%** range, rising above that **only in extreme conditions**.

Communication to Users

Adjusting LTV (especially downward) affects borrowers directly. Therefore, Helios should include:

- **Notifications** via off-chain alerts or on-chain events
- Clear messages like:

> "Global LTV limit reduced due to market conditions."

This helps users:

- Avoid going underwater
- Add collateral or repay before becoming at risk of liquidation

Smooth transitions are key:

- The model won't drop LTV from 80% to 50% in a single block unless under severe stress
- Instead:
  80 → 70 → 60% over a few hours as risk increases

This gives users **time to respond**.

In flash crashes, auto-liquidations will still protect the protocol **if parameter updates occur quickly enough**.

The adaptive logic of Helios ensures:

- **Aggressive growth** during low-risk periods
- **Defensive safety** during market turbulence

It codifies what human risk managers would do into a **fully algorithmic policy**:

- Adjusts parameters continuously
- Responds to real-time inputs
- Optimizes for the prevailing environment

This dynamic mechanism places Helios a step beyond protocols with static risk settings—offering **responsiveness, transparency, and resilience** in one elegant system.

8. Illustrative Scenarios and Stress Testing

To demonstrate how the Helios model behaves in different market conditions, we present four scenarios with indicative inputs. Each shows how the optimizer selects LTV and LP values and what outcomes result for expected bad debt (EBD), capital efficiency (CE), and user experience.

Scenario 1: Calm Market, Low Congestion

Inputs:

- Volatility σ = 50% (annualized) — relatively low for BTC
- Mempool M = low (e.g., < 1,000 transactions)

Optimization Outcome:

- *LTV = 0.80**, *LP = 5%**
- CE ≈ 0.80 (80% utilization of collateral)
- EBD ≈ 0.1% of total loans per year

- Default probability D ≈ 0.5% annually

Despite occasional tail events, losses are negligible and easily covered by fees. This corresponds to a **risk-on, efficiency-maximizing mode**, similar to Aave's "E-mode" but triggered here by low volatility rather than asset correlation.

User Impact:

A borrower with 1 BTC (~$30,000) can borrow up to $24,000. With market calm and low LP, there's a large buffer before liquidation—BTC price would need to fall over 20% to reach liquidation risk.

Scenario 2: Moderately High Volatility, Moderate Congestion

Inputs:

- Volatility σ = 80%
- Mempool M = medium (e.g., 20,000+ backlog)

Optimization Outcome:

- *LTV = 0.65**, *LP = 8%**
- CE ≈ 0.65
- EBD ≈ 0.5% of loans
- D ≈ 1–2% annually

The optimizer tightens parameters in response to rising risk, decreasing borrowing capacity and increasing LP to ensure liquidation incentives.

User Impact:

With 1 BTC collateral, the borrowing cap falls to $19,500. If a user had borrowed $24,000 earlier, they may need to add collateral or repay. At a 15% price drop (BTC value to ~$23,000), their LTV reaches ~85%, nearing the liquidation threshold. If liquidated, the borrower would lose an additional 8% of collateral as the LP.

Scenario 3: Extreme Volatility, High Congestion (Stress Test)

Inputs:

- Volatility σ = 150%
- Mempool M = high (100,000+ transactions, full blocks)

Optimization Outcome:

- *LTV = 0.40**, *LP = 15%**
- CE = 40%
- D ≈ 5% over short horizon
- Expected loss in default ≈ 10% of loan → EBD ≈ 0.5%

In a severe market crash, these settings ensure the system survives even if some defaults occur. By keeping loans small and LP high, the model cushions against execution delays.

User Impact:

1 BTC now supports only $12,000 in borrowing. Users with larger loans may need to deleverage. Whether existing loans are "grandfathered" or subjected to new rules depends on policy.

Liquidators are highly motivated with 15% bonuses, ensuring participation despite congestion. Borrowers incur steep penalties if liquidated, but this protects the system from systemic loss.

Scenario 4: Asset Quality Differentiation (Multiple Collaterals)

Inputs:

- BTC: σ = 60%, Q = 1.0
- XYZ (altcoin): σ = 100%, Q = 0.5
- Mempool M = shared (same blockchain)

Optimization Outcome:

- **BTC:** LTV ≈ 70%, LP = 7%

- **XYZ:** LTV ≈ 35%, LP = 10%

Even under the same macro conditions, XYZ receives a more conservative treatment due to its lower asset quality and higher volatility. If volatility spikes, BTC's LTV might drop to 40%, while XYZ's might fall to 20%.

Interpretation:

This mirrors traditional finance approaches where riskier assets receive greater haircuts. Our framework handles this through the Q parameter, σ scaling, or direct constraint imposition.

Model Behavior Summary

These scenarios show that the Helios model behaves intuitively:

- **Permissive in low-risk environments**
- **Tightens defensively as risks rise**
- **Responds smoothly** to real-time data
- **Balances capital efficiency with solvency**

Helios can be tuned through:

- The weight λ in the objective function
- A hard cap on EBD
- Governance preferences (more aggressive or conservative stance)

Stress Testing

To validate model robustness, we propose historical backtesting. Example:

- Replay BTC price and mempool data from early 2020, including **Black Thursday**
- Simulate the LTV and LP that the model would have assigned over time
- Compare actual vs. simulated EBD and collateral losses

Preliminary analysis suggests:

- Helios would have preemptively reduced LTV as BTC fell and mempool clogged
- This could have **prevented major insolvency events** via early liquidation and tighter limits
- Unlike protocols relying on governance votes (which are slow), Helios adapts automatically

Comparison to Aave V3

- **Aave V3** uses periodic human-reviewed parameter updates (e.g., via Gauntlet or Chaos Labs)
- These updates are reactive, not real-time, and occur weekly or less frequently
- **Helios**, by contrast, integrates optimization directly into protocol behavior—**fully automated and continuous**

This represents a **next-generation approach** to on-chain risk management:

- Scientific, measurable, and explainable
- Built to respond instantly to risk without waiting for off-chain governance
- Transparent and adjustable to any desired risk-return profile

In conclusion, this appendix has defined a rigorous convex optimization framework that underpins Helios's risk parameter selection. By balancing expected bad debt minimization with capital efficiency maximization, and by dynamically adjusting to network and market conditions, the model ensures a **sustainable, resilient lending platform**. The use of convex programming and piecewise-linear approximations marries scientific rigor with on-chain implementability, offering a solution that is both **theoretically sound** and **pragmatically feasible** for a BTC-native DeFi protocol. Each component – from definitions, objective function, to dynamic adjustments – has been justified and exemplified, laying a strong foundation for Helios's risk management strategy to pass both peer review and real-world tests.

References

[1] Aave Documentation (2023). *FAQ: Network / Bridge Risk*. Describes LTV, liquidation thresholds, and governance-adjusted risk parameters in Aave's multi-chain deployments.
[Source: Aave Docs]

[2] Blockworks (n.d.). *The Investor’s Guide to Bitcoin-Backed Lending*. Describes Bitcoin as “pristine collateral” with deep global liquidity, supporting higher LTV treatment.
[Source: Blockworks]

[3] Chainalysis Team (2022). *Cross-Chain Bridge Hacks Emerge as Top Security Risk*. Details how $2B was stolen in cross-chain bridge exploits, accounting for ~69% of all crypto thefts that year.
[Source: Chainalysis Blog]

[4] ChainRisk (2023). *DeFi Lending & Borrowing Risk Framework*. Explores key lending risk parameters (LTV, liquidation bonus, etc.) and their interrelations—provides foundational context for Helios’s risk model.
[Source: ChainRisk Blog]

[5] Chandrasekera, S. (2025). *What are Bitcoin Loans, and How Do They Work?* Explains tax deferral through crypto-backed loans without triggering capital gains events.
[Source: CoinTracker Blog]

[6] Chaos Labs (2020). *Black Thursday for MakerDAO: $8.32 Million Was Liquidated for 0 DAI*. Postmortem analysis of liquidation failure due to volatility and mempool congestion, resulting in ~$5.67M DAI bad debt.
[Source: Medium]

[7] CoinDesk (2020). *Mempool Manipulation Enabled Theft of $8M in MakerDAO Collateral on Black Thursday*. Details how network congestion led to liquidation failure and millions in protocol losses.
[Source: CoinDesk Report]

[8] CoinLedger (2025). *11 Ways to Avoid Cryptocurrency Taxes*. Notes that using crypto as loan collateral is typically non-taxable in most jurisdictions—supports Helios's tax optimization use case.

[Source: CoinLedger Tax Guide]

[9] Factor.fi (2025). *The Power of Delta Neutral Strategies in DeFi*. Introduces delta-neutral yield strategies that eliminate market exposure while retaining interest or yield farming benefits—aligns with Helios's strategy framework.
[Source: Medium Blog]

[10] Fireblocks (2022). *Permissioned DeFi Goes Live with Aave Arc + Fireblocks*. Discusses institutional KYC whitelisting via Aave Arc and the trillion-dollar potential of institutional DeFi—relevant for Helios's permissioned pool roadmap.
[Source: Fireblocks Blog]

[11] Gauntlet (2023). *ARC: Risk Parameter Updates for Aave V3 Optimism (March 8, 2023)*. Defines Aave V3's risk tuning methodology, balancing insolvency, liquidation, and usage metrics. Basis for Helios's own risk-off parameter logic.
[Source: Aave Governance Forum]

[12] High Tower (2025). *MIDL: Infrastructure for Native dApp Execution on Bitcoin*. Introduces the MIDL execution layer, enabling smart contract deployment directly on Bitcoin.
[Source: Medium Article]

[13] Inlock (2018). *Collateral Management*. Explores mid-loan collateral swapping (e.g., BTC to ETH), which Helios implements trustlessly.
[Source: Inlock Whitepaper]

[14] Morpho Docs (n.d.). *Efficiency Mode (E-Mode)*. Discusses Aave's mechanism for boosting LTV on correlated assets, similar in spirit to Helios's volatility-based efficiency mode.

[15] Quantitative Finance Stack Exchange (n.d.). *Is CVaR a Concave or Convex Risk Measure?* Confirms that CVaR (expected shortfall) is a coherent and convex risk measure.
[Source: StackExchange]

[16] Sovryn Docs (2023). *What is Rootstock (RSK)?* Explains RSK as a federated peg Bitcoin sidechain using 1:1 RBTC and merge-mining, as a precedent for BTC-native DeFi.
[Source: Sovryn Platform FAQ]

[17] Aave GitHub (2023). *Aave V3 Risk Parameters* (risk-v3/asset-risk/risk-parameters.md). Lists asset-specific LTV, liquidation thresholds, and isolation modes as part of Aave's risk engine.
[Source: GitHub Repository]

[18] OSL (2025). *Bitcoin Mempool: What Happens to Unconfirmed Transactions?*

[19] OSL (2025). *What is the Bitcoin mempool, and how does it work?*