Block Analitica

Lotus: Economic Review

An economic review of the Lotus Protocol, including its core Lending Market, Interest Rate and Liquidation Modules.

Contents

1 Background **3**

2 Lotus Market Risk **3**
- 2.1 Exogenous Market Risk . . . 3
- 2.2 Endogenous Market Risk (Idiosyncratic Risk) . . . 4

3 Risk Parameters **5**
- 3.1 Liquidation-Loan-To-Value (LLTV) . . . 5
- 3.2 Oracle . . . 5
- 3.3 Liquidation Incentive Factor (LIF) . . . 5
- 3.4 Health Discount Liquidation Module (Reverse Dutch Auction) . . . 6
- 3.5 Interest Rate Model (IRM) . . . 6

4 Slippage and Market Impact Model **7**

5 Agent Based Simulation Environment **7**

6 How Simulations Address Risk **8**

7 Simulation Results **9**
- 7.1 Choice of Tranche Parameters . . . 9
- 7.2 LLTV, LIF, and Bad Debt . . . 9
- 7.3 LLTV, LIF, and Participant Loss . . . 9

8 Access Control and Governance **10**
- 8.1 Core Protocol (`Lotus.sol`) . . . 11
- 8.2 Module Operations: IRM and Liquidation Module Operators . . . 11
- 8.3 Vault V2 Role Hierarchy . . . 12

9 Conclusion **12**

1 Background

Lotus is a decentralized lending protocol that distinguishes itself from similar money markets by granting both lenders and borrowers significantly more agency in determining their risk exposures. Lenders have greater discretion over the degree of market risk they choose to underwrite, as well as the risk profile of the borrowers to whom they supply capital. Conversely, borrowers are empowered with greater control over their specific risk exposure, minimizing the danger of contagion from borrowers positioned further out on the risk curve.

Lotus achieves this by segmenting a single lending market into distinct "Tranches," which can be conceptually likened to the liquidity "ticks" pioneered by Uniswap V3. However, rather than segmenting by ask price, Lotus Tranches segment the market based on liquidation thresholds. Furthermore, Tranches within a unified market are not structurally bound to a single collateral asset, oracle configuration, liquidation incentive, liquidation module, or interest rate policy. These critical parameters are configured on a per-Tranche basis, including the choice of liquidation module, allowing risk managers to select the mechanism best suited to the risk profile of each Tranche. The permutations of these parameters afford lenders and borrowers a high degree of agency, whilst arming risk managers with the flexibility necessary to maintain market solvency and remain highly competitive.

By introducing this architecture, Lotus identifies a strategic opportunity to attract borrowers desiring flexible loan terms and suppliers seeking granular control over the risks they underwrite. In the optimal equilibrium, conservative capital supplied by risk-averse lenders squarely meets the demand of conservative borrowers with minimal frictional costs. Crucially, the loan capacity within Lotus is not strictly constrained to the capital deposited in a given Tranche; it encompasses the supply from Junior Tranches as well. Consequently, Junior suppliers gain access to the yield generated by borrowers on the lower end of the risk curve, weighted according to Tranche utilization.

To rigorously evaluate the economic security of the Lotus protocol, we employ an agent-based simulation framework. At a high level, our models utilize real-world empirical data, specifically Kernel Density Estimation (KDE), alongside stochastic processes such as Geometric Brownian Motion (GBM) and GJR-GARCH(1,1) with jump diffusions (the latter chossen to model leverage effects during a market downturn). This allows us to systematically simulate borrower and liquidator behavior across various market environments to ensure the protocol does not suffer severe impairment.

The remainder of this report is structured as follows: Section 2 defines the precise market risks faced by Lotus participants, categorizing them into exogenous and endogenous factors. Section 3 details the critical risk parameters that govern the protocol. Section 4 introduces the slippage and market impact models used in our assessment. Section 5 describes the agent-based simulation environment. Sections 6 and 7 articulate how these simulations address risk and outline our empirical findings. Finally, Section 8 outlines access control roles and their economic implications with the core modules of the protocol.

2 Lotus Market Risk

2.1 Exogenous Market Risk

Like any decentralized lending facility, Lotus is exposed to volatility in the underlying asset-collateral pair, making it highly sensitive to the volatility in the secondary markets for each leg of the transaction. This risk is further exacerbated during black swan events or regime shifts, when macroeconomic shocks induce exceptionally high volatility and severe price dislocation. Such adverse market conditions increase the cost of borrowing, as the protocol's interest rate policies automatically adjust to prevent insolvency and manage utilization. However, if market

conditions continue to deteriorate and collateral values plummet, the protocol is forced to trigger a liquidation event to mitigate the severity of loss and prevent systemic insolvency. During this event, a portion or the entirety of a borrower's collateral is seized to cover the outstanding debt.

Low or depleted liquidity in the secondary market for the loan and collateral assets also heavily impacts the solvency of the lending market. Rational liquidators, acting as opportunistic arbitrageurs, rely on the depth of the secondary market for the debt and/or collateral asset. The liquidation process frequently involves the liquidator selling a numéraire asset for the loan asset in the secondary market to repay the unhealthy borrower's debt, receiving the borrower's collateral at a discount. Subsequently, the liquidator must sell this collateral in the secondary market to realize a profit. If the liquidator forecasts a loss due to prohibitive slippage or lack of liquidity, they will rationally decline to liquidate the position. Consequently, unrealized Bad Debt will persist within the protocol. Bad Debt is defined as the shortfall when the value of a borrower's outstanding debt exceeds the value of their deposited collateral, representing a direct impairment to the lenders' principal.

2.2 Endogenous Market Risk (Idiosyncratic Risk)

In addition to external shocks, Lotus faces endogenous risks tied to its specific mechanics.

Aggressive liquidations within Junior Tranches can precipitate excessive realized Bad Debt. Realized Bad Debt represents the portion of a defaulted loan that is not covered during the liquidation process, which is subsequently settled pro-rata amongst lenders in the Junior Tranches, proportional to their respective Tranche utilization. To protect suppliers from severe impairment, risk managers must calibrate interest rates to encourage timely repayment (in many cases, routinely), ensuring borrowers have sufficient grace to avoid liquidation, while simultaneously guaranteeing that liquidators are fairly compensated proportional to the risk level of the collateral/loan pair and the specific LLTV.

Furthermore, Lotus' continuous market structure inherently allows Junior suppliers to earn yield generated by Senior borrowers. While beneficial for capital efficiency, this structure introduces liquidity risk in the event of high utilization across current and more Senior Tranches. A "bank run" scenario localized in Senior Tranches can cascade and deplete the available liquidity across all Junior Tranches, driven by the continuous nature of the market's capital pooling.

To illustrate this, we define the *Free Supply* for any given tranche i as:

$$\text{Free Supply}_i = \min_{j \le i}(\text{Net Junior Supply}_j)$$

where $\text{Net Junior Supply}_j = \max(0, \sum_{k \ge j} \text{Supply}_k - \sum_{k \ge j} \text{Borrow}_k)$, representing the aggregate underutilized liquidity from tranche j downwards (more junior tranches).

Now, consider a market state defined by the following distribution:

Index (i)	**LLTV**	**Supply**	**Borrow**	**Net Junior Supply$_i$**
0	75%	2500	2000	$\max(0, 10000 - 9000) = 1000$
1	80%	2500	2200	$\max(0, 7500 - 7000) = 500$
2	85%	1000	2000	$\max(0, 5000 - 4800) = 200$
3	90%	2000	1600	$\max(0, 4000 - 2800) = 1200$
4	95%	2000	1200	$\max(0, 2000 - 1200) = 800$

Table 1: Example Tranche Supply/Debt Distribution and Net Junior Supply

Based on this state, the free supply for the 85% LLTV tranche ($i = 2$) consists of the minimum Net Junior Supply among all senior tranches (including itself):

$$\text{Free Supply}_2 = \min(1000, 500, 200) = 200$$

If a supplier within the 80% LLTV tranche ($i = 1$) conducts a withdrawal of 350, they are able to do so because their localized free supply is $\min(1000, 500) = 500$, and $350 \leq 500$.
However, this withdrawal alters the market state by reducing the cumulative supply for all tranches $j \leq 1$. The new Net Junior Supply for $i = 1$ becomes $\max(0, 7150 - 7000) = 150$, and for $i = 0$ it becomes $\max(0, 9650 - 9000) = 650$.
Therefore, if a supplier in the 85% LLTV tranche ($i = 2$) now attempts to withdraw 200, their withdrawal will fail. Their updated free supply would be:

$$\text{Free Supply}_2 = \min(650, 150, 200) = 150$$

Because $200 > 150$, the withdrawal is blocked. This demonstrates how liquidity constraints in Senior tranches instantly drain the available exit liquidity for Junior participants. To mitigate this dynamic constraint, Interest Rate Models (IRMs) need to be calibrated to sufficiently penalize high utilization, compelling borrowers to promptly repay their debts and free up liquidity.

3 Risk Parameters

This section details how the various parameters within Lotus mitigate the aforementioned market risks. Lotus supports multiple liquidation modules that can be selected independently for each market at creation time. The choice of liquidation module determines the mechanism by which collateral is priced and awarded to liquidators; consequently, certain parameters described below are specific to a particular module and do not apply universally. Each subsection below clearly identifies the scope of the parameter being discussed.

3.1 Liquidation-Loan-To-Value (LLTV)

LLTV is configured at the Tranche level. It acts as a bulwark against insolvency by imposing a strict ceiling on borrower risk. If a borrower's Loan-to-Value (LTV) ratio breaches the LLTV threshold, the position is immediately deemed liquidatable. The LLTV also inherently dictates the maximum leverage a borrower can assume when opening a borrow position.

3.2 Oracle

The Oracle mechanism is a Tranche-level configuration responsible for the collateral-debt pricing mechanism. Lotus is deliberately agnostic to the specific oracle architecture. The oracle could rely on spot secondary market prices, algorithmic pricing mechanisms (such as ERC-4626 Vault pricing), or even a fixed-price oracle. For instance, risk managers can deploy fixed pricing for pegged asset/collateral pairs within Senior Tranches, or implement smoothed pricing methodologies, such as Moving Averages, to dampen the impact of volatility spikes.

3.3 Liquidation Incentive Factor (LIF)

Note: The LIF parameter is exclusively applicable when the `OracleIncentiveLiquidationModule` *is selected during market creation. Markets using alternative liquidation modules, such as the* `HealthDiscountLiquidationModule` *described in Section 3.4, do not use the LIF parameter.*

The LIF is a critical Tranche parameter dictating the discount, or bonus collateral, awarded to liquidator agents for executing a liquidation on an unhealthy position (LTV > LLTV). Within Lotus, establishing the LIF requires navigating a delicate trade-off between incentivizing liquidators and tolerating Bad Debt for Junior Tranches. If the liquidated collateral is insufficient to cover the debt—typically due to abrupt market stress, deleveraging spirals, or oracle mispricing—the protocol suffers a loss. The LIF must be calibrated to guarantee that liquidators are sufficiently motivated to clear distressed debt from the protocol's balance sheet before exogenous market liquidity erodes further.

3.4 Health Discount Liquidation Module (Reverse Dutch Auction)

The `HealthDiscountLiquidationModule` implements a *Reverse Dutch Auction* (RDA) mechanism as an alternative to the fixed-incentive approach of the `OracleIncentiveLiquidationModule`. Rather than awarding a static, pre-configured discount to liquidators, the RDA derives the liquidation discount dynamically from the current health score of the account being liquidated.

The discount d offered to a liquidator is computed as:

$$d = \min(d_{\max}, \max(0, 1 - H))$$

where $H \geq 0$ is the account's health score (with $H \geq 1$ indicating a healthy, non-liquidatable position and $H = 0$ representing full insolvency), and $d_{\max}$ is the `maxLiquidationDiscount` parameter—a hard ceiling on the discount that prevents liquidators from seizing an unfairly large share of collateral even for severely distressed accounts.

This design creates a self-regulating incentive structure. As an account's health deteriorates (i.e., H decreases), the discount increases continuously, attracting progressively more competitive liquidators. Conversely, accounts that are only marginally unhealthy receive a small discount, limiting unnecessary borrower loss for positions that have only narrowly breached the LLTV.

The `HealthDiscountLiquidationModule` essentially incorporates a *forgiveness* mechanism for partial repayments. The liquidator may repay only a portion of a borrower's debt, thereby allowing the account to remain open at its improved (but potentially still sub-threshold) health, preventing a cascade of forced full-liquidations and affording the borrower an opportunity to improve the health of their position.

3.5 Interest Rate Model (IRM)

Each market within Lotus utilizes a single interest rate module. While IRMs are permanently installed for a given market, a designated "Operator" agent (representing a Risk Manager) can dynamically adjust the parameters. The available parameter set is contingent upon the chosen module. At launch, Lotus features five primary IRM designs:

- **Adaptive Linear Kink IRM:** Inspired by Morpho's battle-tested Adaptive Curve IRM, this model determines rates algorithmically at a predictable cadence, driven by the market's utilization dynamics. The foundational architecture resembles the classic Linear Kink IRM utilized by established protocols such as Aave and Compound, where interest rates grow proportionally to the utilization ratio. Governance establishes a target utilization threshold; if breached, rates increase sharply. The critical innovation of the Adaptive Linear Kink IRM is its responsiveness: rates across the curve can dynamically shift upward or downward based on the distance to the target rate and the velocity of borrowing and supplying activity, allowing the model to adapt to prevailing market sentiment. Furthermore, Lotus grants Risk Managers the authority to tune the target utilization, the base borrow rate at the target (including upper and lower bounds), the steepness of the curve post-target, and the "grace period" between updates.

- **Fixed Rate IRM:** This simplistic model shifts the onus entirely to the Risk Manager to establish and maintain rates for each Tranche. Rates remain fixed irrespective of endogenous utilization or exogenous market volatility. This framework is highly suitable for permissioned markets where loan terms are contractually codified prior to debt issuance.

- **Managed Linear Kink IRM:** This model closely mirrors the traditional Linear Kink IRM popularized by Aave. Risk Managers configure parameters including the target utilization and the base borrow rate. A defined "gentle slope" dictates the rate's trajectory as utilization approaches the target. Beyond the target utilization, a "steep slope"—designed

to be significantly more punitive than the gentle slope—is implemented to strongly incentivize repayment and mitigate the risk of insolvency and cascading liquidations.

- **Monotonic Adaptive Linear Kink IRM:** A variant of the Adaptive Linear Kink IRM that replaces the default utilization formula with an enforced monotonic curve alternative. Rather than computing each tranche's utilization solely from cumulative supply and net junior supply, this model uses running maximum per-level local utilization ratios: $\max_{k \le i}\big(1 - \text{Net Junior Supply}_k / \text{Cumulative Supply}_k\big)$ (where Net Junior Supply is the cumulative underutilized liquidity defined in Section 2.2). This guarantees that the utilization input to the rate curve is non-decreasing across tranches—from most senior to most junior—preventing rate inversions where a senior tranche would otherwise price borrowing at a higher rate than a junior tranche. The rate curve and adaptive state mechanics, and operator role are identical to the standard Adaptive Linear Kink IRM.

- **Monotonic Managed Linear Kink IRM:** A variant of the Managed Linear Kink IRM with the same monotonic utilization formula. A designated operator applies rate curve parameters (`TrancheRateConfig`: target utilization, base rate, gentle slope, target rate, steep slope), but the utilization passed into the piecewise kink formula is guaranteed to be non-decreasing across the tranche stack at every accrual. This variant is suitable for permissioned markets that require both manual rate control and structural protection against cross-tranche rate inversions.

4 Slippage and Market Impact Model

In evaluating liquidation dynamics, we must distinguish between temporary and permanent market impact. Slippage refers to the temporary market impact, or the execution cost incurred when a liquidator sells a large quantity of repossessed collateral into the secondary market, temporarily depressing the quoted price. Permanent market impact reflects the long-term adjustment of the asset's equilibrium price following a massive supply shock.

In our agent-based models, slippage is strictly modeled to capture the costs incurred by greedy liquidators looking for immediate arbitrage. To accurately reflect thin liquidity and high-stress environments, we employ an Anchored Gompertz state-space model. This framework evaluates the latency and capacity of the secondary market, bounding the maximum allowable slippage by the underlying collateral total supply. When liquidators evaluate the profitability of a liquidation, they incorporate this slippage model to compute their expected net profit in USD.

5 Agent Based Simulation Environment

To rigorously evaluate the economic limits of the protocol, we employ an Agent-Based Simulation (ABS) environment based on our modeled Lotus market, IRM, and liquidation module. Our models accurately instantiate the Lotus market mechanics, employing Python's `Decimal` type to eliminate fixed-point math precision errors and mirroring the exact calculations of the Solidity smart contracts.

For the initial markets designated for deployment—CBBTC/LotusUSD and wstETH/LotusUSD (with LotusUSD functioning as a USDC wrapper)—we have exclusively modeled the Adaptive Linear Kink IRM. While both the Adaptive and Managed Linear Kink IRMs are structurally appropriate for these markets, the parameters of a Managed Linear Kink can be effectively distilled from the behavior of the Adaptive variant. Consequently, the Adaptive Linear Kink IRM was selected to drive our primary stress tests. **We also base this economic audit on the CBBTC/LotusUSD market**.

Within the environment, agents interact dynamically based on probabilistic models and market state. The simulation is initialized by injecting a predefined budget of liquidity via Supplier agents, distributed evenly across all active tranches.

Borrower agents are then iteratively spawned into the market. Their behavior—specifically their collateral sizes and desired Loan-To-Value (LTV) ratios—is sampled from a Kernel Density Estimation (KDE) distribution trained on empirical onchain lending data. Figure 1 visualizes the LTV distributions across distinct borrower tiers derived from the KDE model.

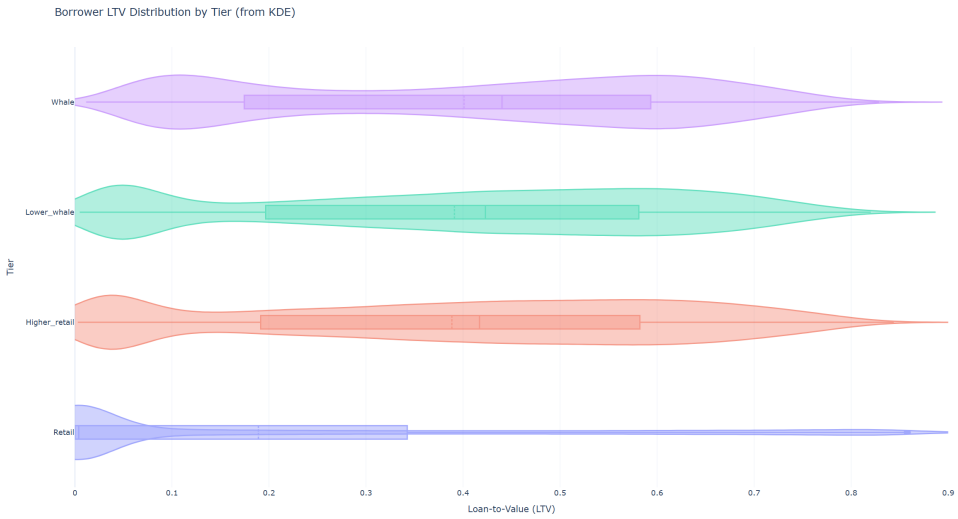


Figure 1: Borrower LTV Distribution by Tier sampled from KDE.

As illustrated in Figure 1, different tiers exhibit distinct risk profiles. When spawned, a borrower evaluates the available tranches and is optimally assigned to the most Senior tranche whose LLTV threshold safely accommodates their desired LTV. However, to model realistic market friction, borrowers will abort their entry if the target tranche lacks sufficient unutilized liquidity or if its localized annualized borrow rate has temporarily spiked above a prohibitive threshold (e.g., 5%).

Finally, Liquidator agents continuously monitor the health of all borrower positions. If a position's LTV breaches the tranche's LLTV, liquidators attempt to seize the collateral. Their execution logic incorporates the Anchored Gompertz slippage model; they dynamically calculate the market impact of offloading the seized collateral and will only execute the liquidation (or partial liquidations at 80% or 64% of the debt) if the net USD profit remains strictly positive, thereby leaving unprofitable debt to persist as unrealized Bad Debt.

6 How Simulations Address Risk

Our agent-based models are subjected to both moderate and severe/exaggerated market stress scenarios to evaluate protocol resilience during macroeconomic regime shifts and black swan events. Asset pricing paths are generated using stochastic models to simulate abrupt jump shocks and continuous volatility. Simulating under extreme volatility is crucial for parameter optimization; it identifies the precise thresholds at which liquidator incentives fail and the protocol accrues terminal Bad Debt.

To evaluate and optimize parameter combinations, we rely on two primary quantitative metrics:

1. **Bad Debt Ratio:** The ratio of the sum of unrealized and realized Bad Debt to the total supply of a given Tranche.

2. **Agent Returns:** The profitability of the liquidator agents, ensuring the system remains economically attractive to external keepers during periods of duress.

7 Simulation Results

Note: The simulation results and parameter recommendations presented in this section—including Sections 7.2 and 7.3—assume that markets are deployed using the `OracleIncentiveLiquidationModule`*. The LIF-based incentive structure is central to the analysis below. Markets utilizing the* `HealthDiscountLiquid` *(Reverse Dutch Auction) operate under a fundamentally different incentive mechanism and would require a separate parameterization study.*

7.1 Choice of Tranche Parameters

To reduce complexity in the simulations and focus on meaningful economic interactions, we tested a discrete set of LLTV boundaries rather than a continuous spectrum. In production, Lotus' deployments—particularly on Ethereum Mainnet—must carefully choose which tranches to make available to users due to gas constraints. Managing liquidity across dozens of granular tranches is expensive; Lotus is aware of this limitation, which is why the protocol offers suppliers the option to opt out of interest accrual upon withdrawal to save gas.

This environment makes it highly impractical to deploy markets with spacing as tight as 0.01 between LLTVs. Consequently, our analysis determines that only tranches which demonstrate significant potential for meeting localized demand for credit, as identified by borrower profiles in our KDE models, should be considered for deployment.

7.2 LLTV, LIF, and Bad Debt

The analysis in this subsection applies to markets using the `OracleIncentiveLiquidationModule`*.*

LLTV thresholds are primarily set to protect suppliers from principal impairment by triggering liquidations before a borrower's debt spirals to a level where repayment becomes impossible. Simultaneously, setting sufficiently high LLTVs allows risk-seeking borrowers to apply significant leverage to their positions, stimulating market utilization. Borrowers operating within more Junior tranches (higher LLTVs) are therefore expected to manage their positions actively and repay their debt in a timely manner, or else liquidators must be incentivized enough to clear the debt for them.

In our analysis, we find that ideally, the Liquidation Incentive Factor (LIF) should be set with inverse proportionality to the LLTV. Senior tranches with low LLTVs should carry the highest incentive for liquidators. This design ensures that in the event borrowers are not reactive enough to repay their debt during a severe black swan event—where even positions in a conservative 50% LLTV market rapidly become unhealthy—liquidators remain highly motivated to step in despite severe secondary market slippage. Figure 2 illustrates the relationship between these parameters and the resulting Bad Debt Ratio.

7.3 LLTV, LIF, and Participant Loss

The analysis in this subsection applies to markets using the `OracleIncentiveLiquidationModule`*.*

Beyond systemic Bad Debt, we evaluated the direct financial impact on market participants, measuring both Supplier and Borrower losses under varying configurations.

Supplier loss correlates strongly with LLTV. Our analysis indicates, unsurprisingly, that suppliers providing liquidity to more Junior tranches capture significantly more yield due to higher utilization and borrowing rates. It is important to note that in Figure 3, negative values for the supplier loss metric denote a net positive return (profit) for the supplier.

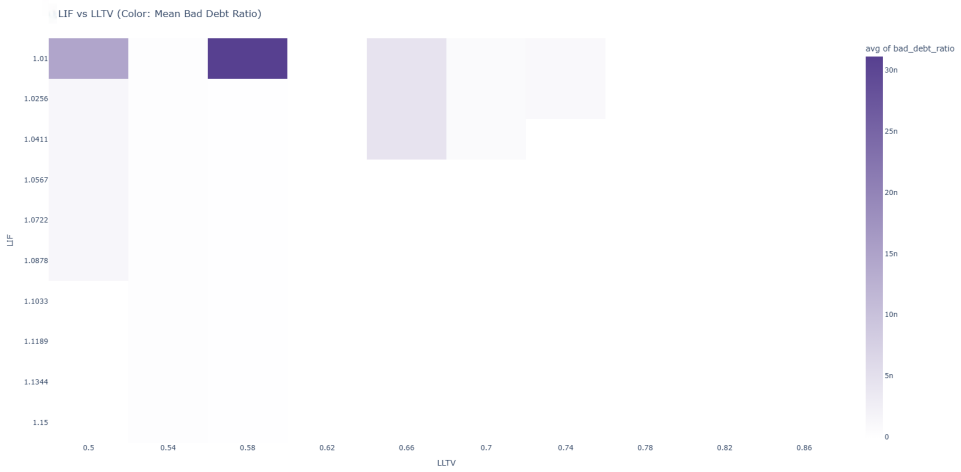


Figure 2: Mean Bad Debt Ratio relative to LLTV and LIF.

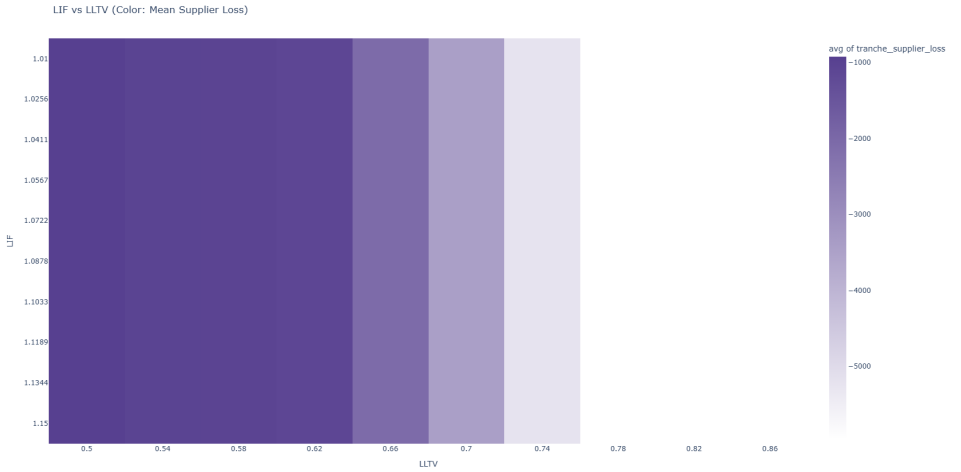


Figure 3: Supplier Loss (negative denotes profit) across LLTV and LIF.

Conversely, Borrower loss demonstrates a stronger correlation with the LIF. A higher LIF dictates a greater loss for the borrower, which is an expected outcome given that liquidators are seizing their underlying collateral at a steeper discount. Since borrower loss inherently accounts for the total value of their forfeited collateral, the most Senior tranches bear the highest cost for borrowers during stress events; liquidations at this Senior level are typically triggered by sharp, catastrophic downturns in collateral price rather than gradual interest accrual. Meanwhile, the 78% LLTV tranche experienced high demand based on the dense concentration of LTV preferences within our KDE borrower distribution, concentrating liquidation activity in that zone during localized volatility. Figure 4 visualizes these outcomes.

8 Access Control and Governance

This section reviews the trust assumptions embedded in the Lotus protocol's role model, identifying the parties that hold privileged authority over live markets, the scope of their powers, and the safeguards—or absence thereof—surrounding those powers.

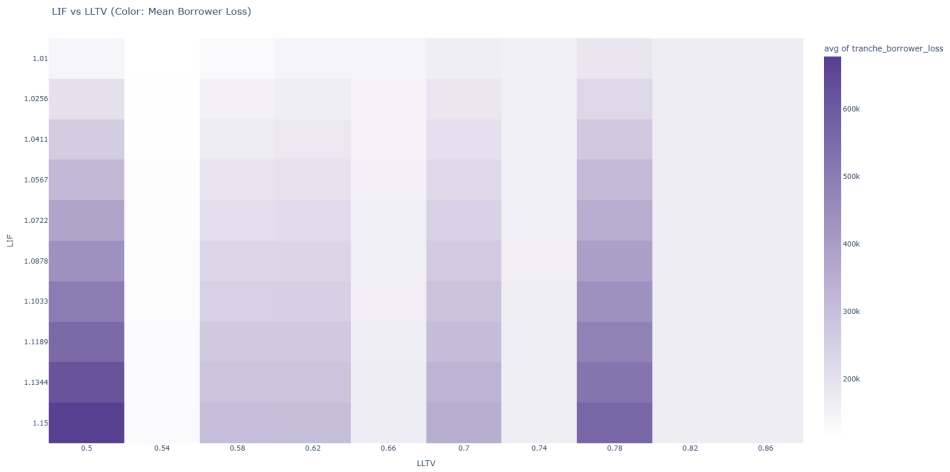


Figure 4: Borrower Loss relative to LLTV and LIF.

8.1 Core Protocol (`Lotus.sol`)

The core protocol exposes a single `owner` address that holds broad, instantaneous authority over the most security-critical parameters. Specifically, the `owner` can: set per-tranche protocol interest fees (capped at 25%), designate or change the fee recipient address, set per-token flash-loan fees (capped at 25%), whitelist new LLTV values (a one-way, irreversible operation), whitelist IRMs and Liquidation Modules (shared with the `moduleManager` role; also one-way), grant or revoke the `moduleManager` role, and attach per-market hook contracts. Crucially, the `owner` transfer itself (`setOwner`) is a single-step operation with no acceptance step (i.e. push only, not the often preferred push-pull pattern): a misconfigured address would permanently and irreversibly lock all of these administrative functions. There is no timelock and no multi-step confirmation on any core protocol action.

The `isModuleManager` mapping grants a secondary tier of addresses the ability to whitelist Interest Rate Models (`enableIrm`) and Liquidation Modules (`enableLiquidationModule`) on the protocol. Both of these whitelisting operations are also one-way and irreversible: once an IRM or Liquidation Module is enabled, it cannot be disabled. Module managers can be granted or revoked by the `owner` at any time without a timelock.

8.2 Module Operations: IRM and Liquidation Module Operators

Each deployed IRM contract (including the `FixedRateIrm`, `ManagedLinearKinkIrm`, `AdaptiveLinearKinkIrm`, and their monotonic variants) and each Liquidation Module (`HealthDiscountLiquidationModule` and `OracleIncentiveLiquidationModule`) maintains a dedicated `operator` address. This role is entirely separate from the Lotus core `owner` and grants direct, instant authority over the parameters of live markets.

For IRM contracts, the `operator` can modify per-tranche rate curve parameters—including fixed rates, kink slopes, target utilization, and adaptive configuration bounds—without any delay, directly affecting the cost of borrowing for all active positions in the affected market. The `AdaptiveLinearKinkIrm` operator can additionally freeze the adaptive rate-adjustment mechanism entirely via `setAdaptationPaused`. For Liquidation Modules, the `operator` controls the `maxLiquidationDiscount` (for the `HealthDiscountLiquidationModule`) and the `liquidationIncentiveFact` (for the `OracleIncentiveLiquidationModule`), both of which directly govern the incentive structure applied to liquidations.

It is worth noting that for the monotonic IRM variants (`MonotonicAdaptiveLinearKinkIrm` and `MonotonicManagedLinearKinkIrm`), the monotonic utilization property—which guarantees

non-decreasing utilization inputs across the tranche stack—is an immutable implementation invariant, not a configurable parameter. It is enforced by the contract's override of the internal `_computeUtilizations` and `accrueBorrowInterest` hooks and cannot be toggled or overridden by the `operator`. The operator's authority is therefore identical to that of the corresponding base IRM, with the structural rate-inversion protection always active regardless of parameter configuration.

A notable operational risk exists in the operator succession model. The `setOperator` function on all IRM and Liquidation Module contracts is restricted to the *current operator* exclusively—there is no owner override path. If the operator key is lost or compromised, the module's upgrade owner (held via `Ownable2StepUpgradeable`) cannot recover the operator role through governance alone; a UUPS implementation upgrade would be required to restore access, introducing significant operational complexity and potential market disruption.

8.3 Vault V2 Role Hierarchy

Vault V2 (`VaultV2`) implements a four-role hierarchy with differentiated privilege levels:

- `owner`: Holds the highest authority. Can transfer vault ownership (`setOwner`), designate the curator (`setCurator`), grant or revoke the sentinel role (`setIsSentinel`), and update vault metadata. All `owner` actions are instant with no timelock.
- `curator`: Responsible for structural vault configuration, including adding adapters (`addAdapter`), managing allocator permissions (`setIsAllocator`), updating the adapter registry, and adjusting allocation caps. Curator actions are subject to a *submit-then-execute* timelock pattern, providing a safety delay before structural changes take effect. However, timelocks are initialized at zero for vaults deployed via `LotusVaultFactory`; the effective delay for post-deployment governance depends entirely on the vault's own configuration after handoff. The curator can also call `abdicate` to permanently and irrevocably forfeit its authority over specific functions.
- `allocator`: Manages capital deployment. Can supply and withdraw assets across adapters (`allocate`, `deallocate`), configure the "auto deposit" liquidity adapter (automatically allocates based on predefined allocation ratios), and set the maximum interest rate. Allocator actions carry no timelock.
- `sentinel`: Holds emergency withdrawal authority (`forceDeallocate`) to pull assets from adapters with a penalty. No timelock.

An important depositor protection is enforced at vault creation: `LotusVaultFactory` calls `abdicate(setAdapterRegistry.selector)` during deployment, permanently preventing any future curator from substituting the vault's adapter registry. While the registry owner can still revoke adapter trust globally, the per-vault registry binding is immutable from inception.

9 Conclusion

Through our rigorous agent-based Monte Carlo simulations, we evaluated the economic security of the Lotus Protocol under both moderate and severe market stress. The multi-tranche architecture of Lotus successfully empowers users to tailor their risk exposures, but it necessitates careful parameterization to mitigate systemic insolvency and limit excessive participant losses.

Our findings indicate that Tranche LLTVs should ideally be constrained within the range of $[0.64, 0.92]$ for both markets. To effectively protect suppliers while maintaining a fair environment for borrowers, the Liquidation Incentive Factor (LIF) should be calibrated inversely

to the LLTV for markets utilizing the `OracleIncentiveLiquidationModule`. We recommend establishing LIFs using a methodology akin to Morpho's battle-tested formula:

$$\text{LIF} = \min\left(1.15, \frac{1}{c \cdot \text{LLTV} + (1 - c)}\right)$$

where c is a weighting parameter that should not fall below 0.3. A maximum LIF of 1.15 is widely recognized and battle-tested across decentralized lending markets as an appropriate ceiling to incentivize liquidators during severe drawdowns without overly penalizing borrowers. Markets deployed with the `HealthDiscountLiquidationModule` instead parameterize the `maxLiquidationDiscount` ($d_{\max}$). We recommend a $d_{\max}$ of 15% (`1500e-4`) per attempt, as this provides an analogous incentive profile to a 1.15 LIF to return the borrower to good health.

Furthermore, our simulations were designed to identify optimal parameter configurations for the Adaptive Linear Kink Interest Rate Model (IRM). While the Adaptive IRM responds dynamically to market conditions, we recognize that deploying the Managed Linear Kink IRM may present a simpler implementation path and deliver more predictable rate environments. However, protocols must acknowledge that utilizing the Managed IRM inherently shifts a significant onus onto the Risk Manager, who must actively monitor and manually adjust rates for each deployed tranche to respond to evolving liquidity demands and market volatility.

Finally, the access control review in Section 8 reveals that the Lotus protocol relies heavily on trusted, privileged roles of operating authority with limited restrictions. The core protocol `owner` can modify fees, whitelist modules, and transfer protocol ownership in a single transaction with no timelock or multi step confirmation (push-pull pattern). This allows owners to act quickly in case of some catastrophic disaster, but puts tremendous trust assumptions on this agent and the infrastructure in which it uses to manage its keys. IRM and Liquidation Module `operator` addresses can alter rate curves and liquidation discount caps on live markets also without a timelock, which could potentially affect active borrower positions. Another potential risk vector for the IRM and Liquidation Module is the operator succession model: since `setOperator` is restricted to the current operator with no owner override, a lost or compromised operator key cannot be recovered through conventional governance—only a UUPS implementation upgrade can restore control. Therefore standard multi-signature wallets should be opted for these roles rather than externally owned accounts before critical modifications take effect.