ChatGPT

Interest Rate Models in Ethereum DeFi Lending and Derivatives (2015–2025)

Introduction

Decentralized Finance (DeFi) lending protocols like Aave and Compound have pioneered algorithmic interest rate models that differ fundamentally from traditional finance. In TradFi, central banks set base rates and banks adjust lending rates slowly based on policy and market factors. In DeFi, interest rates are *programmatically determined* by smart contracts based on real-time supply and demand in liquidity pools [1] [2]. Lenders deposit assets into pools and borrowers take loans against collateral; the protocols balance these forces by using **utilization-based interest rate curves** that update continuously with each block or transaction [3] [4]. The goal is to incentivize an equilibrium: low utilization of a pool will result in lower borrowing rates (to attract borrowers) and high utilization will drive up rates (to attract more lenders and encourage loan repayment) [5] [6]. This dynamic helps maintain liquidity for withdrawals while maximizing capital efficiency.

Over the past decade, a variety of interest rate models have been proposed and implemented in Ethereum-based lending and even certain derivative platforms. These range from simple linear models to "kinked" jump-rate curves and more complex adaptive or time-weighted schemes. Academic research has also flourished, analyzing the performance and optimality of these models. In this report, we survey the notable interest rate models from 2015 to 2025, including both theoretical frameworks and live protocol implementations. We compare their performance in terms of utilization balance, rate stability, capital efficiency, and risk management. Finally, we consider how these models might fit into the user's own DeFi lending system (as seen in the provided codebase) which supports pluggable interest rate strategies.

Theoretical Frameworks for DeFi Interest Rate Models

From the early days of DeFi, researchers recognized that interest rate mechanisms in decentralized money markets act as an **algorithmic market-clearing tool** – effectively the "price" of liquidity [4] [7]. Several academic works in the last 10 years have studied how to design these algorithms optimally:

- **Taxonomy of Models:** Gudgeon *et al.* (2020) introduced a taxonomy for DeFi lending rate models, classifying them as *linear, non-linear,* or *kinked* [7] [8]. Early protocols used simple linear models, while others introduced non-linear (convex) curves or piecewise-linear "kink" functions. A linear model increases interest proportionally with utilization; a non-linear model (e.g. quadratic) accelerates the rate increase as utilization grows; a kinked model is piecewise – gentle slope up to a chosen utilization threshold, then a steeper slope [9] [10]. This taxonomy helps frame the design choices made by protocols.

- **Equilibrating Supply and Demand:** A fundamental objective is to set rates that equilibrate the supply of lendable funds with borrow demand, achieving a target utilization that keeps the system

efficient but liquid [11] [12]. Cohen *et al.* (2023) modeled borrower-lender equilibrium and proposed setting rates that drive utilization toward a *target level* [13]. Such a feedback mechanism (essentially a control system) can adjust rates to maintain an "optimal utilization" (often 80–90%) that balances earning yield on assets vs. keeping some liquidity available [14] [10]. This idea underpins many practical models (as we'll see in Aave and Compound designs).

- **User Elasticity and Adaptive Models:** Recent research has explored making interest rates *adaptive* to market conditions beyond a static formula. Bastankhah *et al.* (2024) introduced the concept of **user elasticity** – how sensitively users change borrow or supply behavior in response to rate changes – and designed data-driven models that adjust rates based on observed elasticity [15]. Their findings suggest that **adaptive interest rate models can improve market efficiency** by responding to shifts in demand, but there is a catch: greater adaptivity can increase vulnerability to manipulation [16]. For example, if the rate algorithm reacts too quickly, a savvy user might oscillate utilization (e.g. withdrawing liquidity to spike rates, then supplying back) to earn higher yields or cheap borrowing in a short window. The researchers quantify a trade-off between adaptivity and adversarial robustness [16] – meaning a highly reactive model might be more efficient in theory but could be gamed under certain conditions. This insight informs protocol designers to introduce adaptivity cautiously or with safeguards.

- **Optimal Control Approaches:** Pushing the envelope, some works treat interest rate setting as an *optimal control* or reinforcement learning problem. Bertucci *et al.* (2024) and Baude *et al.* (2025) formulated the interest rate adjustment as a continuous control system that maximizes a utility (such as total lender profit) under constraints (like keeping utilization within safe bounds) [17] [18]. By modeling block-by-block liquidity dynamics and using techniques like stochastic control and deep reinforcement learning, they solved for an "optimal" interest rate policy. These studies often benchmark standard industry models (linear or kinked curves) against a theoretically optimal model. Interestingly, results indicate that the *simple utilization-based models used in practice are quite effective*: for instance, one study found that common parametric models (linear or bilinear/kinked) achieve nearly the same risk-adjusted performance as the optimal control solution in many scenarios, with a simple linear model being the notable underperformer [19]. In other words, the widely used piecewise linear ("kink") model is already close to optimal in balancing pool wealth vs. liquidity risk in their analyses [19]. The remaining performance gap might be closed by slight parameter tuning or mild adaptivity rather than completely different functional forms.

- **Interest Rate Rules vs. External Factors:** Another thread of research examines how DeFi rates relate (or don't) to traditional interest benchmarks. Studies like Heimbach and Huang (2024, BIS) found that transmission of central bank policy rates into DeFi lending markets has been limited, especially for USD stablecoin markets. This is unsurprising because DeFi rates are driven predominantly by on-chain supply-demand and crypto-market conditions, not by fiat lending markets. Governance can adjust base rates or curves to react to macro conditions, but it's a manual and lagging process [20]. For example, MakerDAO's governance has adjusted its stability fee (the borrowing rate for DAI) between 0% and 8% over time to influence DAI supply [21], and protocols like Compound have had governance votes to raise rates during periods of excessive borrowing. However, these adjustments are infrequent compared to the continuous algorithmic updates of utilization-based models.

In summary, theoretical research supports the core principle that **utilization-driven interest models** are effective and largely optimal for decentralized lending pools [22] [23]. More advanced schemes (e.g. adaptive or optimal control-based) can marginally improve efficiency or stability, but they introduce complexity and potential new risks [16]. The prevalent wisdom is to use a model that keeps the utilization in a healthy range (not too low to waste capital, not too high to risk liquidity lock-up) and adjust parameters through governance or mild algorithmic tweaks as needed.

Implemented Interest Rate Models in Major Protocols

Linear and Non-Linear Utilization Curves

Early simple models: The simplest model used in DeFi lending is a *linear interest rate*:

$$R_{\text{borrow}}(U) = R_{\text{base}} + R_{\text{slope}} \times U,$$

where U is the current utilization (fraction of available liquidity that's borrowed) [24]. Compound's original "Standard" rate model for assets like WBTC was exactly this: e.g. Base 2% and Slope 30% meant borrow APR = 2% + 30% * U [25]. At 10% utilization, borrow APR would be 5% in that example, scaling linearly up to 32% if utilization hit 100% [26]. The supply APY is then derived as $R_{\text{supply}} = R_{\text{borrow}} \times U \times (1 - f)$, where f is the reserve factor (protocol's cut) [27] [28]. This ensures the system is solvent (interest paid out $\leq$ interest paid in) [29]. A linear model is easy to implement and predictable, but it has a drawback: it doesn't strongly penalize very high utilization. For instance, at 90% utilization, borrow APR would be only moderately high (e.g. 2% + 30%*0.9 = 29% in the WBTC case)* [30]*. If the pool gets* close* to 100% utilization, a linear model might not raise rates fast enough to deter further borrowing or to attract enough new deposits, potentially risking liquidity exhaustion.

Quadratic/non-linear models: To address the above issue without an abrupt kink, some protocols opted for a smoothly **accelerating curve**. Notably, dYdX (an early DeFi margin trading protocol with lending pools) implemented a quadratic interest rate formula [31]. In essence, dYdX set $R_b(U) = a \cdot U + b \cdot U^2$ (specific coefficients varied per market) [31]. This *non-linear model* means as utilization increases, the marginal rate increase gets larger (since the U^2 term grows). In effect, interest rates start rising slowly at low utilization but become increasingly steep as U approaches 100%, **without a sudden kink**. Gudgeon *et al.* observe that such convex models create a "non-linearly increasing incentive" for lenders and borrowers to respond as utilization grows [9]. For example, at moderate utilization the rate might be only slightly above linear, but at very high utilization the quadratic term dominates, pushing the rate up sharply to signal urgency. This achieves a similar end goal as a kinked model (discouraging >90% utilization) but with a smooth curve. However, quadratic models are a bit more complex to calibrate and reason about. In practice, dYdX's specific parameters yielded interest rate growth that was comparable to kinked models in effect [9], and later iterations of dYdX shifted focus toward perpetual markets (with a different "funding rate" mechanism) rather than pool-based lending.

Exponential and other curves: Other functional forms have been tested. Some projects (including the user's codebase) experimented with **exponential interest models**, where $R_{\text{borrow}} = R_{\text{base}} + C \times (\exp(kU) - 1)$. Here, k controls the curvature intensity. This model is highly punitive as U → 1: interest accelerates exponentially, which virtually guarantees no one will borrow the last drops of liquidity because the rate becomes prohibitive. The downside is that even at mid-range utilization it might yield higher rates

than linear/kinked models, potentially under-utilizing capital if k isn't tuned well. In the uploaded `mini-defi` code, for example, an `ExponentialInterestRateModel` computes APR as `baseAPR + coefficientAPR * (e^(factor * U) - 1)`. If `factor` is small, the curve is closer to linear; if large, it mimics a hard kink, skyrocketing at high U. Real-world use of pure exponential curves has been limited – most major protocols stick to linear or piecewise-linear for simplicity – but it's conceptually similar to quadratic models in creating a *smoothly* increasing slope.

Overall, purely linear models have largely fallen out of favor for multi-asset pools because they don't provide a "safety valve" at high utilization. Non-linear continuous curves (quadratic, exponential) address that by steepening the rate as utilization grows, but they are somewhat harder to intuitively govern. The dominant design that emerged is the **piecewise linear kink model**, which combines simplicity and a strong high-utilization response, as discussed next.

Kinked "Jump" Rate Models (Compound & Aave)

The most famous model in DeFi lending is the **jump-rate model** popularized by Compound, and similarly used by Aave and many others. This is a *piecewise linear* curve with a defined **kink (optimal utilization)** point. For utilization U up to the kink U^*, the borrow interest rate increases at a moderate slope; once U exceeds U^*, the slope switches to a much higher value, causing a sharp upturn in rates [14] [32]. In formula terms (as described by Compound's documentation):

$$R_b(U) = \begin{cases} R_{\text{base}} + S_1 \cdot U & , \; U \le U^*; \\ R_{\text{base}} + S_1 \cdot U^* + S_2 \cdot (U - U^*) & , \; U > U^*, \end{cases}$$

where S_1 is the slope (per-utilization rate increase) below the kink, and S_2 (often called *jump multiplier*) is the much steeper slope after the kink [32]. The kink U^* (optimal utilization ratio) is typically set around 80%–90%. For example, Compound's **USDC** market uses $U^* = 80\%$ and parameters like Base = 0%, $S_1 = 5\%$ / year, $S_2 = 109\%$ /year [33]. Under 80% utilization, borrow APR grows linearly from 0% up to 5% at 80%. Beyond 80%, the rate *jumps* – at 90% utilization, the borrow APR would be $5\% * 80\% + 109\% * (90\% - 80\%) \approx 14.9\%$ [34], and if it hit 100%, APR would approach ~25% in this case [35]. Such high rates strongly incentivize corrective actions: new lenders are attracted by higher returns, and some borrowers will repay or get liquidated due to the cost, pushing utilization back down. As a result, the system tends to self-correct toward the kink point.

Aave's interest rate strategy is very similar. Aave V2/V3 reserves define an **optimal utilization (U_optimal)** and two slopes `slope1` and `slope2` for interest before and after that point [36] [37]. For instance, Aave's DAI reserve (cited in 2023) had U_optimal = 80%, base rate = 0%, slope1 = 4% and slope2 = 75% [36]. That means up to 80% utilization, DAI borrow rates climb gradually from 0 to 4%; beyond 80%, the rate curve steepens dramatically (if utilization hit 100%, the variable borrow rate would exceed 75%). This two-segment model ensures **"a gentle slope before optimal utilization (to promote efficient usage) and a sharp slope after (to protect liquidity)"** [10] [38]. The rationale, as Aave documentation notes, is to keep the pool attractive and liquid: under normal conditions the moderate rates encourage borrowing and lending, but if liquidity gets scarce (U → 100%), the protocol reacts with punitive rates to restore balance [39] [40]. In effect, the kink is a target that the system will hover around.

This kinked model has proven very successful in practice. Compound's and Aave's experiences show it maintains higher **capital utilization (~80% on average)** than a linear model would (since rates don't become high until needed), yet provides **strong stability** by averting 100% utilization scenarios. For example, during a spike in DAI borrowing on Aave in March 2023 when utilization neared 100%, the interest rate spiked automatically, which quickly incentivized repayments and new deposits, preventing a liquidity crunch [41]. The piecewise model's parameters are adjustable via governance (and indeed have been tweaked over time for different assets and market conditions). Governance might raise the base rate in volatile periods (ensuring a minimum yield for lenders) or adjust slopes if an asset consistently sits at too high/low utilization [42]. However, these adjustments are infrequent; day-to-day, the model runs autonomously.

Reserve factor and supply rate: Both Compound and Aave also apply a reserve factor (typically a small percent, e.g. 10% or 20%) which takes a cut of borrower interest to accrue to the protocol's reserves or insurance fund [43] [44]. The result is that the lender APY is slightly less than the borrow APR times utilization. For instance, in Compound if borrow APR = 15% and utilization is 90% with a 10% reserve factor, the supply APY = 15% * 90% * (1-0.1) = 12.15%. This ensures the protocol builds reserves and that lenders are only paid interest actually earned from borrowers [45] [46]. The user's codebase similarly accounts for this via a `reserveFactor` in the supply rate formula. It's worth noting that Compound V3 (an isolated market version) simplified the supply rate calculation – it derives supply rate directly as a function of utilization, rather than using the borrow rate product formula [47] – but the principle of supply being < borrow rate * util still holds to avoid insolvency [47].

Stable (fixed) rates: Aave historically offered a **Stable Rate** borrowing option alongside variable rates [48]. A stable rate in Aave is essentially a semi-fixed APR for borrowers that does not change with every block, providing predictability. However, it's not immutable – if the market's variable rate diverges too much, the protocol can *rebalance* stable loans (either by migrating them to variable or adjusting the stable rate). For example, if stable borrowers are paying far less interest than new variable borrowers (say stable APR is < 90% of current variable APR), Aave can trigger rebalancing to prevent abuse [49] [50]. This feature was intended for users wanting long-term loans with known costs. In practice, stable rates were set based on a formula using the current variable rate plus a premium, and would update only upon certain thresholds or user actions. It provided a derivative-like product (a fixed-rate loan) within the protocol. However, stable rate mode had complexity and even a bug; Aave governance decided in 2024 to deprecate stable borrowing entirely [51]. The move underscores that managing fixed rates in a volatile environment is challenging – essentially the protocol was taking on interest rate risk, which proved non-trivial. In the broader DeFi space, separate protocols like **Notional Finance** emerged to specialize in fixed-rate lending via on-chain order books or AMMs for loan tokens (offering fixed rates by matching borrowers and lenders). These platforms treat fixed-rate as a true tradeable derivative (e.g. zero-coupon bond model) [52] [53] rather than an internally subsidized rate. Still, for most general lending pools (Compound, Aave, etc.), **variable-rate with kinked curves remains the standard**, as it's easier to manage risk when rates float with utilization.

Dynamic and Algorithmic Rate Adjustments

A newer development in DeFi interest models (circa 2022–2025) is the introduction of **algorithmic dynamic adjustments** that go beyond a static function of instantaneous utilization. The prime example is **Fraxlend**

(by Frax Finance), which implemented innovative rate calculators that change over time based on utilization trends [54] [55]:

- **Time-Weighted Variable Rate:** Fraxlend's *TWV* model doesn't set APR solely by the current utilization; instead, it defines a target utilization range (e.g. 75–85%), and an adjustment speed (half-life parameter) [56] [57]. If utilization stays above the target band, the interest rate *ramps up gradually over time*; if utilization stays below, the rate *decays over time*. In other words, the rate will keep increasing until borrowing slows enough to bring utilization back into the desired band. The half-life (e.g. 12 hours) determines how fast the rate can move – after each half-life period out of equilibrium, the difference between current rate and some max/min is halved [58]. This creates a self-correcting system that **lets the market find the appropriate rate** without governance intervention [57] [59]. The design acknowledges that no static curve fits all conditions; instead, the rate dynamically *seeks* an equilibrium where utilization is healthy. For example, if demand surges and utilization goes high, the TWV model will continuously raise rates (with exponential time-based growth) until demand cools and utilization falls back into range [57]. Conversely, in a period of low demand, rates will slowly drift down to encourage more borrowing. This reduces oscillation compared to an instantaneous jump model; changes happen smoothly over hours unless utilization is at extremes.

- **Hybrid Linear + Adaptive (Variable Rate V2):** Fraxlend also offers a combination model where the instantaneous rate comes from a linear or kink function, *but the curve's parameters themselves adjust over time* based on utilization [60] [61]. Specifically, the "Variable Rate V2" uses a linear formula with a vertex (kink) as in a standard model, but it dynamically raises or lowers that curve's slopes (vertex rate and max rate) using the TWV formula if utilization remains too high or low for extended periods [61]. For instance, if a market persistently sees utilization near 100%, not only does the momentary rate go up, but the entire curve (future rates) will shift upward over time – meaning even if utilization drops back to 50%, the new base rate might be higher than before, because the system learned that demand can be high. This approach tries to capture long-term market regime shifts (similar in spirit to how a central bank might gradually raise interest rate levels in a tightening cycle). It's like having an **algorithmic monetary policy** encoded in the contract. Over the long run, this could adjust a market's interest rate sensitivity to match observed demand patterns, theoretically improving capital efficiency while still safeguarding liquidity [62] [63].

- **Other Dynamic Models:** A few other protocols have toyed with dynamic adjustments. Some versions of **MakerDAO** considered adaptive Peg Stability Module fees or dynamic DSR (though currently DAI Savings Rate is manually set by governance). **Algorithmic stablecoin platforms** sometimes use dynamic interest for lending markets to maintain peg stability (e.g. lowering borrowing costs when the stablecoin is overpeg and raising them when underpeg). These are usually simple feedback controllers targeting an external metric (the peg) rather than utilization per se. Additionally, **Gauntlet Network** and other risk managers have provided recommendations to Compound/Aave to occasionally update interest model parameters based on market conditions (a form of off-chain algorithmic tuning).

Real-world usage of fully autonomous adaptive rates (like Fraxlend's) is still young, but preliminary results are promising. Fraxlend demonstrated that you can run months without governance tweaks, as the contract finds rates that keep utilization near the target most of the time [57] [59]. This yields very high **capital utilization** without liquidity crises and lets the "market" (borrowers and lenders reacting to rates) determine the appropriate level. The trade-off, as academic research noted, is potential complexity and new

attack surfaces: if someone can predict or influence the rate changes, they might exploit that (for example, timing a large deposit/withdrawal before a rate update to game the system). So far, no major exploits of this nature have occurred, but it's an area to monitor.

Interest Rate Mechanisms in Derivatives Platforms

Beyond lending pools, the concept of an interest rate appears in some DeFi derivatives contexts too:

- **Perpetual Futures Funding Rates:** Many decentralized perpetual swap exchanges (like dYdX v3, Perpetual Protocol, etc.) use a *funding rate* mechanism to keep the derivative's price aligned with the index. Funding rates are effectively periodic interest payments between long and short traders. On dYdX's perpetual markets, for example, the funding rate each hour had an "interest rate component" plus a premium based on price deviation [64] [65]. In practice, the interest rate component was often a fixed small number (representing the baseline difference in borrowing costs of the two assets, e.g. USD stablecoin vs. crypto). This is different from lending pool rates – it's more about balancing derivative positions than utilization of a pool. However, it underscores that *some* DeFi platforms incorporate interest-like fees to maintain stability. The models here are usually trivial (constant or externally determined interest, plus a market-driven premium component).

- **Fixed-Rate and Rate Derivatives Protocols:** As mentioned, protocols like Notional, Yield, and Voltz have created markets for **interest rate swaps or fixed-rate loans**. Notional Finance uses a model where users trade a token representing a fixed debt or deposit (fCash) for variable assets today, effectively locking in a rate. The "interest rate model" in these systems is simply an order-book or AMM pricing mechanism – interest rates are determined by supply/demand of fixed vs. floating interest positions, often following formulas borrowed from bond markets (e.g. present value curves). For instance, Voltz built an AMM for interest swaps where liquidity providers make markets on forward interest rates, with the AMM using a curve model for the swap's payoff. While these platforms are fascinating, they are beyond the scope of algorithmic utilization curves – they treat interest rate as a tradable *market price* rather than setting it by an explicit function. Thus, they fall more under derivatives pricing than algorithmic rate *models*, strictly speaking.

In summary, *most DeFi platforms that directly set interest rates (as opposed to deriving them via market forces) are lending protocols*. The majority have converged on utilization-based models (linear, kinked, or dynamic) for determining variable rates, while a few offer fixed-rate products via separate mechanisms.

Performance and Risk Comparison of Models

Different interest rate models can be evaluated on several key criteria: **capital efficiency** (keeping the pool funds utilized to earn interest), **rate stability/predictability**, and **risk management** (preventing illiquidity or wild swings that could harm the protocol). There are inherent trade-offs:

- **Linear vs. Kinked:** Linear models are very predictable and smooth, but they are *suboptimal in both efficiency and risk*. They often result in lower average utilization (because rates at mid-utilization might be set higher than necessary to cover worst-case, leaving money on the table) and at the same time do not sufficiently deter near-100% utilization. Kinked models improve on both fronts: they keep rates relatively low until the optimal utilization is reached (maximizing usage and competitive borrowing costs), then rapidly increase rates to cap utilization in a safe zone [14] [66].

This design maintains a liquidity buffer (e.g. ~10–20% of funds idle) as a safety margin. Empirically, Compound and Aave saw utilization hover near their kink targets much of the time, indicating high efficiency. And instances of 100% utilization were rare and short-lived because the jump in rates quickly corrected the imbalance [41]. Thus, kinked models strike an effective balance and have become the *industry standard* for pooled lending. The trade-off is a slight discontinuity at the kink, but in practice that discontinuity is not a problem – in fact it's the intended feature to strongly signal when utilization is beyond the comfort zone.

- **Non-linear Continuous (Quadratic/Exponential):** These share a similar goal to kinked curves – making interest increasingly aggressive as utilization rises – but without a hard breakpoint. In terms of capital efficiency, a well-calibrated quadratic or exponential can mimic a kinked model's performance. For example, dYdX's quadratic model was designed such that at high utilization the rates shot up comparably to Compound's jump model [9]. One potential advantage is *smoothness*: no sudden rate jump at a specific point. That could theoretically avoid borrowers "gaming" around the kink (though in reality, borrowers can't instantly adjust their behavior at a magic number because utilization is a global metric). One disadvantage is that continuous non-linear models are harder for users to intuitively understand or predict – governance and users might prefer the clear two-phase regime ("below 80% util, expect ~X% APR; above 80%, expect much higher"). From a risk lens, both approaches cover the bases as long as the high-utilization tail of the curve is steep enough. There haven't been reports of major issues with quadratic models on-chain; they just were less common, possibly due to ease-of-use considerations. The **exponential model** in the user's code would behave similarly: if configured with a modest exponent factor, it would be close to linear early on, and very steep as $U \to 1$. It could potentially keep utilization *even more tightly* away from 100%, but if too steep it might also cause large swings (a small drop in utilization could drastically drop APR, then utilization surges, etc., leading to oscillation). Kinked models by contrast have a linear segment below the kink that acts as a buffer before the steep section, which might reduce oscillatory behavior. Overall, properly tuned non-linear curves can be as safe as kinked models, but the simplicity and proven track record of the kink model have made it more popular.

- **Adaptive Dynamic Models:** The time-weighted and adaptive models (e.g. Fraxlend's) aim to maximize both utilization and stability by continuously nudging the rate toward where it needs to be. In terms of **capital efficiency**, these likely achieve the highest sustained utilization because they will keep adjusting rates downward during low demand until borrowing picks up, and adjust upward in high demand until it cools – ideally keeping utilization centered in the optimal band. This means less idle capital and more interest earned over time. In terms of **rate stability**, interestingly, these models can be gentler on users in the short term (no instant spikes, changes are gradual), but they introduce a new form of unpredictability over longer horizons (the entire curve can drift). Borrowers and lenders might find it harder to estimate future rates since it depends on the time-integral of utilization deviations. However, by design, if the system is near equilibrium, rates won't move much – they only swing when there's a persistent imbalance. So one could argue this *stabilizes the utilization* (primary goal) at the expense of more complex rate behavior. From a **risk management** perspective, adaptive models shine in preventing extreme conditions – they actively course-correct to avoid both liquidity dry-ups and long periods of under-utilization. The risk trade-off, as noted, is potential vulnerability to manipulation or just complexity risk (bugs in a more complex contract logic). The research by Bastankhah *et al.* warned that adaptive models can be gamed in theory [16]; a concrete example might be a scenario where a whale oscillates utilization to force the rate down then up for profit. So far, no major incident of this kind has occurred, but fewer protocols have battle-tested

dynamic rates compared to the thousands of user-years of static models like Compound's. So conservatively, many projects may stick to simpler models until adaptivity's safety is proven.

- **Fixed/Stable Rates vs Variable:** Offering stable rates (either internally like Aave V2 did, or via separate fixed-rate markets) is beneficial for user experience (predictability), but it introduces **interest rate risk** to the protocol or liquidity providers. Aave's stable rate mode ended up being problematic enough to remove [51]. Protocols like Notional handle that risk by letting users trade it (a lender who wants fixed yield essentially sells the variability to someone else). In terms of capital efficiency, fixed-rate pools usually saw lower utilization because they segregate liquidity by term and require over-collateralization with interest, etc. Thus, from the system design viewpoint, unless fixed rates are a core product requirement, it might be safer to avoid them or keep them as isolated offerings. A design like the user's code (which is a single-asset pooled lender) aligns more with the floating-rate model approach.

In practice, **the kinked variable-rate model has offered a very good blend of performance and safety**, which explains its dominance. It keeps pools sufficiently utilized (often 70–90%) so lenders earn yield, but has prevented catastrophic liquidity crises in volatile times by spiking rates when needed [41] [40]. More exotic models can improve on certain aspects (e.g. adaptivity can keep 85% utilization instead of 80% on average, squeezing more yield), but the improvements may be incremental while adding operational complexity.

Applicability to the User's DeFi Lending Architecture

The uploaded codebase (`mini-defi-main`) appears to implement a **modular interest rate model interface**, with three example models: a `LinearInterestRateModel`, a `KinkInterestRateModel` (Compound-style jump rate), and an `ExponentialInterestRateModel`. This design is wise – it allows plugging in different rate strategies without changing core lending logic. Given this architecture and the considerations above, here are some insights on which models or approaches might fit best:

- **Kink (Jump) Model as a Default:** The kinked model (as implemented in `KinkInterestRateModel.sol`) is a proven choice for a general-purpose lending pool. It would likely be the safest *default* for the system, especially if it's a pooled, over-collateralized lender similar to Compound/Aave. By setting an optimal utilization (say 80%) and tuning base and slopes appropriate to the asset's volatility and liquidity profile, the protocol can achieve high utilization and react to stress. For example, one could configure a stablecoin market with a low base APR (0–2%), a moderate slope1 (e.g. 5–10% at full utilization below kink), and a high slope2 (e.g. 100%+ at full utilization beyond kink) similar to Compound's USDC parameters [33]. This would keep stablecoin borrowing cheap in normal conditions but shoot up to >20% if the pool starts running dry, thereby protecting liquidity [35]. The code's implementation mirrors Compound's formula, so it should behave equivalently. Using this model aligns the protocol with industry-standard risk management, and there's a wealth of empirical data and research to guide parameter choices [14] [66].

- **Linear Model for Simplicity or Special Cases:** The linear model in the code (`LinearInterestRateModel.sol`) might be useful for certain cases, but generally it's outclassed by the kinked model. One scenario where a linear model could be acceptable is if the protocol expects utilization never to approach dangerous levels (for instance, if there's always excess liquidity

or external incentives keep utilization low). In that case, the kink's second slope might never activate, and a linear model would suffice. However, that's hard to guarantee. Another use might be for **testing or educational purposes** – linear is easier to predict, so it could help users understand how borrowing affects rates in a sandbox. In production, though, running a pure linear model means at 100% utilization you only get at most `base + slope` APR. If that number isn't extremely high, the pool could become fully depleted because the interest didn't rise fast enough to dissuade borrowing. Thus, unless one sets an unnaturally steep slope, linear model could compromise liquidity safety. It's telling that virtually all major pools migrated from linear to kinked models around 2019–2020 once they experienced liquidity crunches with linear rates.

- **Exponential/Non-Linear Model for Research or Niche Markets:** The `ExponentialInterestRateModel` provided could be used if one wanted to experiment with a continuously curved rate or to imitate a model like dYdX's. It may fit niche markets – for instance, perhaps a very volatile asset where you want an extremely cautious stance as utilization grows (exponential rise might keep utilization lower but add safety). It could also be part of a **two-token system**: imagine a scenario where the protocol's own governance token is lent/borrowed; one might use a custom curve to, say, strongly discourage borrowing beyond a point to prevent governance attacks. Generally, though, an exponential model would need careful parameter tuning to avoid either being too mild (and behaving basically linear) or too aggressive (essentially acting like a kink but without clarity). Since the code's exponential model is unbounded as $U \to 1$ (aside from practical limits of exponential calculation), it will create very high interest near full utilization, which is safe but could lead to instability in rate calculations or confusion for users if it ever enters that regime. A kinked model caps the worst-case rate to a linear function, which is easier to reason about (e.g. "max borrow APR will be ~X% at 100% utilization").

- **Incorporating Adaptive Features:** The current architecture doesn't explicitly include a time-component in interest updates (the interface `getBorrowRatePerSecond(utilization)` is memoryless – it calculates based only on current utilization). Adding a dynamic model like Fraxlend's would require maintaining state (e.g. the last interest rate or some moving utilization average) and updating it over time. This is more complex but feasible to integrate. If the project's goal is to innovate on rate models, one could implement an `AdaptiveInterestRateModel` contract that on each block (or each borrow/repay action) adjusts an internal rate variable toward a target. The modular design would allow swapping this in for a given asset's pool. That said, unless the team has strong controls and testing, it may be wise to first deploy a simpler model (like the kink model) and observe its performance, only later introducing adaptive behavior. Starting with a well-known model provides a baseline; any adaptive enhancements could then be measured against that baseline in terms of increased utilization or stability. The research suggests that adaptivity yields diminishing returns with added risk of manipulation [16], so the benefits should be weighed carefully. It might be "best fit" for the architecture in a controlled way – e.g., perhaps for a stablecoin market that is critical to keep around a peg or a target utilization, an adaptive model could be very useful, whereas for a small volatile asset market, the simpler kink might suffice.

- **Risk Parameters and Governance:** Whichever model is used, it's crucial to choose parameters in line with the protocol's risk appetite. The code's models are `Ownable` (likely allowing an admin or governance to update APR parameters). This means the system can adjust base rates or slopes as needed. For instance, if market conditions change (say, generally higher interest across crypto markets), governance might raise baseAPRs a bit so that lending yields stay competitive [42]. Or if an

asset is consistently at too high utilization, one might raise slope2 (making post-kink rates even higher) to force it down, as Llama Risk noted: increasing `slope2` makes the pool respond more sharply to high utilization [40]. The ability to tweak on the fly is important for risk management – indeed, many DeFi incidents have been mitigated by prompt governance action to adjust interest rates (or pause markets). The user's architecture, by keeping the model pluggable and updatable, is aligned with this practice.

- **Monitoring and Iteration:** Finally, whichever model is initially chosen for deployment, the team should closely monitor metrics like average utilization, volatility of utilization, and user feedback. If using the kink model and finding that utilization sits well below the optimal most of the time, it could indicate rates are too high (maybe lower base or slopes to encourage more borrowing). If utilization spikes to 100% and sticks, it suggests rates or the kink are set too low, requiring more aggressive parameters. With an adaptive model, monitoring is even more important to ensure it behaves as expected and doesn't oscillate or create opportunities for exploitation.

In conclusion, for a generic Ethereum-based lending pool, the **piecewise linear (kink) model is a robust choice** backed by years of industry use [67] [68]. It would likely "fit" best as the starting point in the user's system. The linear and exponential models can be valuable for comparison, testing, or specialized situations, but they are less commonly used as the primary model in leading protocols. As the platform matures, one could consider **integrating dynamic rate adjustments** akin to Fraxlend's model to further optimize utilization, but this should be done with careful design to avoid new risks. The modular interest rate framework in the codebase is well-suited to iterating on these ideas – it provides the flexibility to evolve the model as more data and research insights become available. With prudent parameter governance and community input, the interest rate model can be tuned to achieve the protocol's desired balance of **performance (high earning and lending activity), stability (smooth rate changes), and risk management (robust response to stress)**.

References and Further Reading

- Leshner, R. & Hayes, G. (2019). *Compound Whitepaper* – Algorithmic interest rate model overview.
- Aave Documentation (2020–2023). *Interest Rate Strategy* – Optimal utilization and two-slope model details [37] [10].
- Gudgeon et al. (2020). *DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency* – taxonomy of interest rate models and cross-protocol analysis [7] [9].
- Cohen et al. (2023). *Interest Rate Rules in DeFi* – equilibrium analysis and target utilization model proposal [13].
- Bastankhah et al. (2024). *Adaptive Interest Rate Mechanisms* – user elasticity concept and adaptive model with manipulation trade-offs [16].
- Baude et al. (2025). *Optimal Risk-Aware Interest Rates for DeFi Lending* – agent-based optimal control approach and benchmarking of Compound/Aave curves [19] [17].
- RareSkills (2023). *Interest Rate Model of Aave V3 and Compound V2* – developer-focused explainer on utilization, kinked curves, and parameter examples [14] [36].
- Ian Macalinao (2020). *Understanding Compound's Interest Rates* – illustrated guide to Compound's linear and jump-rate formulas with examples [32] [34].
- Frax Finance Docs (2023). *Fraxlend Interest Rate Models* – description of Linear, Time-Weighted Variable, and Variable Rate V2 models [56] [61].

- Llama Risk (2024). *Aave IRM and TradFi Symbiosis* – risk analysis of Aave's interest model and recent stable rate deprecation [40] [49].

---

[1] [5] [14] [36] [43] [44] [45] [46] [47] [67] [68] The interest rate model of AAVE V3 and Compound V2 | By RareSkills – RareSkills
https://rareskills.io/post/aave-interest-rate-model

[2] [3] [6] [37] [41] [42] [48] Aave Interest Rate Model Explained
https://www.krayondigital.com/blog/aave-interest-rate-model-explained

[4] [20] [21] [24] [25] [26] [27] [28] [29] [30] [32] [33] [34] [35] Understanding Compound protocol's interest rates | Ian Macalinao
https://ianm.com/posts/2020-12-20-understanding-compound-protocols-interest-rates

[7] [8] [9] [31] DeFi Protocols for Loanable Funds: Interest Rates, Liquidity and Market Efficiency
https://berkeley-defi.github.io/assets/material/DeFi%20Protocols%20for%20Loanable%20Funds.pdf

[10] [12] [38] [39] [40] [49] [50] [51] [66] AAVE Interest Rate Model and the TradFi Symbiosis - Llama Risk
https://www.llamarisk.com/research/aave-irm

[11] [13] [15] [16] [17] [18] [19] [22] [23] Optimal risk-aware interest rates for decentralized lending protocols
https://arxiv.org/html/2502.19862v1

[52] An introduction to Notional Finance, a fixed-rate lending protocol
https://scapital.medium.com/an-introduction-to-notional-finance-a-fixed-rate-lending-protocol-be46b01788d6

[53] Notional Finance Price, NOTE Price, Live Charts, and Marketcap
https://www.coinbase.com/price/notional-finance

[54] cmDeFi #005 A Comprehensive Study of the Frax Stablecoin ...
https://www.gate.com/learn/articles/a-comprehensive-study-of-the-frax-stablecoin-ecosystem/1088

[55] [56] [57] [58] [59] [60] [61] [62] [63] Interest Rates | Frax Finance ¤
https://docs.frax.finance/fraxlend/advanced-concepts/interest-rates

[64] Default funding rates on dYdX
https://help.dydx.trade/en/articles/166992-default-funding-rates-on-dydx

[65] Funding rate and skew formula - HackMD
https://hackmd.io/@anhdungle/H1BOjfAs3