Delorean

Buy and sell future yield at discount

gm@delorean.exchange
March 2023

Abstract

Delorean is a yield swap protocol for buying and selling future yield. Sellers of future yield can receive upfront tokens for future yield. Buyers of the same can get access to yield without exposure to the price of the underlying tokens. This transaction is facilitated by yield slices representing a continuous segment of yield. Yield slices direct payments into or out of a liquidity pool which applies a future discount rate to make asynchronous payments fungible. Market pricing enables price discovery on the discounted future payments. Delorean concentrates all yield liquidity, regardless of timing, in a single pool, making it useful for short term applications like overnight lending, as well as long term applications like invoice factoring. It is well suited to real yield applications that pay yield in a token different from protocol tokens.

Introduction

Time value of money dictates that money today is worth more than money tomorrow, and money tomorrow is worth more than money next year. Each incremental day that I wait for payment decreases its value. This is illustrated by the classic tradeoff: would you rather have $100 today, or $109 a year from now? The amount where you express no preference between the two options is your discount on future money.

The net present value formula allows us to map between streams of future payments, and money today. Would you rather have $10 a month for the next five years, or $500 today? Assuming a constant discount rate, the NPV formula will give you the answer. The formula is

$$NPV = \sum_{t=1}^{n} \frac{FV_t}{(1+i)^t}$$

where n is the number of payments, FV_t is the future payment in term t, and i is the discount rate.

The net present value formula can be applied to decentralized finance. Suppose you have a token TOK which pays out daily yield in ETH, at the rate of 0.01 ETH per TOK per day. We can compute the present value of slices of future daily payments, using the daily discount rate of 0.1% per day.

Number of tokens	Days of yield	Nominal value	Present value	Net discount
500	10	50	47.8	4.38%
500	20	100	91.0	8.95%
500	100	500	317.0	36.60%

For payments in the near future, for example the next 10 days, there is a small discount on the yield. However, the further into the future we reach, the net discount increases. Selling yield 100 days into the future results in the above example results in a large net discount on its nominal value.

Trading future yield is useful for both the buyer and the seller. The yield seller can access his future cash flow, and use that cash today. For a company, this means making investments. For an individual, this means consumption. Because the upfront cash is backed by future yield, there is no need for liquidation.

On the flip side, the yield buyer gets access to a return without exposure to the underlying token. Many yield generating tokens in decentralized finance have high price volatility. While a yield rate of 20% may be appealing, investors are turned off by the idea of risking large negative returns in the token price. A protocol that enables access to the yield without token price exposure can bring in new capital.

Prior work and requirements

There are existing protocols in decentralized finance enabling various ways to trade and transfer future yield. However, the existing protocols come with a variety of flaws and compromises that make them unusable in certain cases, or inefficient in others. Lets go over the requirements for a protocol to sell future yield, and see how prior work fails to meet these requirements.

Terms must be user specified

Many existing protocols make one of two mistakes. Either they have fixed, pre-defined terms (like Pendle), or they have no terms at all (like Timeless and Alchemix). Both of these approaches are wrong.

Pre-defined terms create two problems. First, any specific term length is unusable for some applications. An overnight lending use case doesn't work if your expiry is set six months out, and likewise the TVL allowed for overnight lending doesn't help fund investments in a business. The second problem arises when you solve the first by adding lots of different terms for each use case. The more terms you have, the more liquidity is split across them. This is fine in traditional finance where banks provide huge amounts of liquidity, but decentralized finance must do more with less. You need to concentrate liquidity whenever possible.

Some protocols have taken the opposite approach, and eliminated terms entirely. This is also wrong. If you create a token representing the entire future yield of the underlying, you simply recreate the underlying. Remember that the value of an equity is the net present value of all its future cash flows. Additionally, we need different discount rates for different terms. An overnight loan should have a smaller discount than a year-long loan.

The correct approach is to allow the user to specify the term, on a case-by-case basis. The user should be able to specify the length of his loan, and draw from the same liquidity pool whether it is for one day or one year. The protocol should dynamically price the different terms, with increasing discounts on yield that is further in the future.

Deep market priced liquidity

Some existing protocols lack market based pricing for future yield, or have limited liquidity. They may rely on capped vaults (like Alchemix), or on algorithmic pricing (like Flashstake). Neither of these approaches works. A capped vault is a market inefficiency. If the cap is hit, that means the price for the lender is too low, and should be increased. If the vault is underutilized, that indicates mis-pricing in the opposite direction. Algorithmic pricing is better, but it relies on proper tuning and at best approximates what the true market price is.

What we need is a system with two properties: deep liquidity, and market based price discovery. We already have tools for both these problems in the form of decentralized exchanges and AMM pools. Existing solutions include classic XY=K pools, stableswap curves, concentrated liquidity, single sided liquidity, and more. A standard ERC-20 token can plug into this ecosystem to enable price discovery and efficient liquidity management.

Dual tokens must be supported

Some existing protocols (like Flashstake) only support yield which is paid out in the same token as the underlying asset. While this is the case for many tokens, such as Lido's stETH paying out more stETH, it is not the case for many others, such as GLP paying out yield in ETH. As decentralized finance markets mature and protocols begin to generate cash flows, we expect more and more projects to pay out yield in ETH to holders of their protocol token. For this reason, a protocol for trading future yield must have dual token support.

Summary

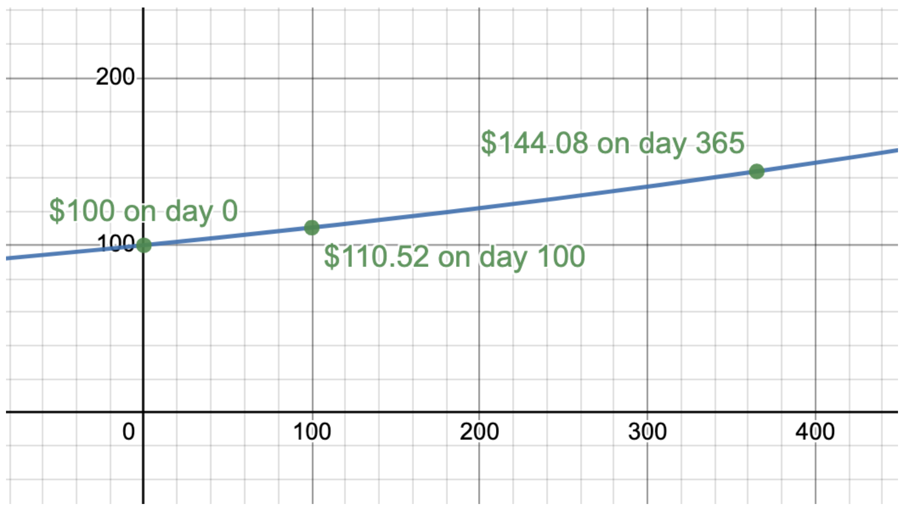
The table below summarizes flaws in existing protocols, and the complementary requirements for the Delorean yield swap protocol.

Prior Work Flaw	**Complementary requirement**	**Reason**
Fixed terms. Some prior work has a set of pre-defined terms that users must pick from	**User specified terms.** The yield seller must be able to specify any term he desires, up to a maximum	Avoid splitting liquidity, allow multiple use cases to pull from same liquidity pool
No terms at all. Some prior work separates yield and principal indefinitely.	**Time bound terms.** There must be a bound unlock condition for a yield slice.	Pricing indefinitely separated yield is difficult.
Single token only. Some prior work assumes yield and principal are the same token.	**Dual token support.** There must be support for yield paid out in a different token.	Many protocols pay out yield in a different token than the yield generator.
Non-market pricing. Some prior work lacks market based pricing for future yield.	**Market based pricing.** The value of future yield must be determined by market forces.	Efficient pricing.
Capped vaults. Some prior work caps loans outstanding based on vault size.	**Liquidity management.** Existing DeFi tools enable lenders to manage and price their liquidity.	Efficient liquidity allocation.

Time indifference curve

The central assumption of Delorean is the time indifference curve. This indifference curve maps future payments to their present value. Market participants are assumed to be indifferent between the various points on the curve. That is, each point on the curve represents a different payment amount at a different time, and under the time indifference curve market participants find the combinations interchangeable.

As an example, assume a daily discount rate of 0.1%. We can derive a time indifference curve for comparing $100 today to amounts in the future.



In this example, we are indifferent between the choices of receiving $100 today, $110.52 on day 100, or $144.08 in 365 days. The blue line indicates all the equivalently acceptable points on the curve.

If our assumption holds, and the market is indeed indifferent along the curve, then we can freely swap payments along the curve. If a seller wishes to offload $110.52 paid out in 100 days, and exchange it for $144.08 in 365 days, we should be able to find a buyer or market maker to take the opposite side of that trade.

The time indifference curve can be represented by the present value formula:

$$Present\ Value\ =\ Future\ Payment(i)\ \times\ Discount\ Rate(i)$$

Where *Future Payment(i)* is the amount paid out in the i'th epoch, and *Discount Rate(i)* is the relative discount route in the i'th epoch compared to the 0'th epoch.

Because payments sooner are always better than payments later, the following constraint applies:

$$Discount\ Rate(i) \prec Discount\ Rate(i + 1)$$

The most common discount rate formula is the epochal compounding discount rate:

$$Discount\ Rate(i) = (1 - r)^{i}$$

However, other discount rate formulas are possible as they satisfy the previous constraint.

Slicing yield

A yield slice represents a continuous output of yield in accordance with a time indifference curve. Yield slices come in two varieties: debt and credit. Debt yield slices lock yield generating tokens in escrow, and redirect yield flow from the slice's minter to a beneficiary. Credit yield slices redirect yield flow from a pool to the slice's minter. In each case, the amount of payment is determined by the net present value of the yield, relative to the mint time. In the case of debt slices, the locked tokens are released once the slices payments are completed.

As an example, suppose I have 100 XYZ tokens, and every day they produce 1 ETH of yield. I can create a yield slice for these tokens for the segment between the 10'th ETH and 25'th ETH produced, represented by the function call *DebtSlice(100 XYZ, 10 ETH, 25 ETH)*. This slice represents 15 ETH of future yield, to be delivered starting 10 days from now, and ending 25 days from now. As another example, I can create a second yield slice from the same tokens that is also for 15 ETH, but further in the future: *DebtSlice(100 XYZ, 50 ETH, 65 ETH)*. In this case, the slice pays out yield starting in 50 days, and ending in 65 days.

Yield slices and their components are unfortunately not directly tradable. This is because the yield slice and its components are non-fungible. Two yield slices may have the same nominal payout, but if one has fewer generating tokens locked, it is worth less. This is because it will take longer to pay out, and thus has a higher discount rate. In the example above, the first slice we described is worth more than the second for this reason.

Trading yield slices would be a convenient way to exchange the timing of yield payouts between counterparties, but to do so we need a way to make them fungible.

Net present value tokens

Recall that the time indifference curve plots out equivalent payment values over time. We can map a yield slice onto the time indifference curve to calculate its net present value:

$$Net\ Present\ Value = \sum_{i=1}^{t} Payment(i) \times Discount\ Rate(i)$$

The result of this formula gives a single value that we can use to compare yield slices. The table below gives some example valuations, using the assumption that each XYZ token produces 0.01 ETH in daily yield, and the daily discount rate is 0.1%.

Tokens locked	Starting at yield	Ending at yield	NPV
200	100	200	92.7341352
100	100	200	86.05717432
50	0	100	90.58490968
50	0	50	47.55632252
50	500	600	33.30765684

Notice that the value of yield slices varies based on the number of tokens locked as well as the start and end range of the yield included.

Mapping yield slices to net present value is how we make yield slices tradable. In the Delorean protocol, locking tokens into a yield slice also simultaneously mints NPV tokens equal to the net present value of that yield slice, as determined by the expected rate of yield, and the discount rate. These tokens are standard ERC-20 tokens, and they can be used in two ways.

Time swap for NPV tokens

The first use of NPV tokens is to swap them for the complement of debt yield slices: credit yield slices.

Whereas a debt yield slice is a locker that directs yield away from the original holder, a credit yield slice does the opposite. It directs yield from a general pool towards the beneficiary of that slice. As with debt slices, credit slices assume the time indifference curve is correct, and payout on that curve. The beneficiary of the credit slice is assumed to be indifferent towards payouts at various points on the curve.

Credit yield slices are obtained by taking an amount of NPV tokens, and converting them into a credit slice. As an example, suppose you have 100 NPV tokens. You can claim a credit slice *CreditSlice(100 NPV)*, which creates the corresponding yield slice with you as the beneficiary. It may pay out 100 ETH today, or 110.52 ETH in 100 days, or it may give a series of payments over the next 500 days that in total have an NPV of 100 ETH. By minting the yield slice, you necessarily accept all of these possibilities as equally valuable.

In practice, the payout schedule of credit slices depends on two factors: the composition of the debt slices sold, and the performance of the underlying asset.

Liquidity for NPV tokens

The second use for NPV tokens is to sell them on the open market, for example on decentralized exchanges like Uniswap or Curve.

Recall that NPV tokens are minted when someone locks tokens into a debt slice. This person is unlikely to want to create a credit slice with those tokens: that would leave him in the same position that he started with, owning a stream of payments in the future. Instead, this person is likely looking to get liquidity today for his future payments.

To do this, the minter of the NPV tokens can take those tokens, and sell them into a AMM pool on a decentralized exchange.

How do we price NPV tokens? Recall that NPV tokens represent payout in some other token, such as ETH, and they are the net present value of that payout based on an indifference curve. If the indifference curve is correctly calibrated, meaning the discount rate at each epoch reflects market preferences, then a 1 NPV token should be worth exactly 1 of the yield token, for example 1 ETH. In this scenario, the buyer of the NPV tokens wants them so he can mint a credit slice and obtain a stream of future payments.

In the case of a well calibrated time indifference curve, liquidity providers for NPV:ETH tokens can concentrate most of their liquidity around at or slightly below 1 ETH per NPV. Because money today always has equal or greater value than money tomorrow, the value of 1 NPV token should never exceed 1 ETH. This is enforced by concentrated liquidity settings, or arbitrage. The tight concentration of liquidity enables good capital efficiency.

Discounted NPV tokens

If the time indifference curve is misconfigured, NPV tokens will trade inefficiently and lose their expected 1:1 price parity with ETH. As an example, let's look at what happens when the discount rate is too low.

If the discount rate is too low, that means we are overvaluing future payments, and the NPV numbers we get are inflated. In this case, the price of the NPV tokens in relation to ETH will drop until they reflect the true discount rate. For example, suppose the time indifference curve gives a 10% discount on ETH payments one year from now, but the true market discount rate is 25%. A price of 0.83 ETH per NPV will balance out this discrepancy.

Unfortunately, while this discount solves the problem of payments one year from now, it creates a problem at the front of the curve: a payment of 1 ETH tomorrow is valued at a 17% discount. This is unlikely to be the case. The general problem with a single NPV token market is that it cannot do price discovery between epochs. To create an efficient market we need something more granular.

Epochal NPV tokens

Recall that money today is always worth more than money in the future. Also, let's imagine that the time indifference curve for a Delorean pool represents the *lower bound* on the future discount, not necessarily the true future discount. At any point in time on the curve, a payment is worth *at most* what the curve tells us, but maybe less.

Additionally, let's add a new type of NPV token, the epochal NPV token. These tokens pay out no sooner than a specified epoch, and the value of their payout is computed as if that epoch were actually time = 0. For example, if you have 9.9 NPV_3 tokens, you can mint a credit slice that starts paying out no sooner than epoch 3, and it will pay out 9.9 in net present value thereafter. If the discount rate is 0.9, that may mean a 11 ETH payment in epoch 4.

We can combine the assumptions above with epochal NPV tokens to add two new mint() functions to Delorean:

1. Take present ETH, and mint an equivalent number of NPV_0 tokens.
2. Take NPV_t tokens, and mint $\frac{1}{r}NPV_{t+1}$ tokens

Generally, present ETH can be used to mint $\left(\frac{1}{r}\right)^t NPV_t$ tokens. If the true value of NPV_t tokens is below the market rate in any epoch, arbitrageurs will be able to take advantage, resulting in price discovery.

Because the number of epochs will be very high, potentially in the thousands, we cannot expect liquidity in individual NPV_t:ETH markets. The use of epochal NPV tokens requires a modified AMM pool with an analog to concentrated liquidity that allows liquidity providers to choose between making their capital available for all epochs, or a specific range.

Applications

Delorean is a general protocol with a variety of applications relating to the exchange of value over time. Let's go over a few common use cases.

Invoice factoring

Invoice factoring in traditional finance is a transaction that exchanges future accounts receivables for cash today, paid for by a third party called a factor. Using Delorean, we can facilitate the same transaction in decentralized finance. Future accounts receivables are represented by the future yield on a set of tokens, and the third party factor underwriting the transaction becomes the liquidity provider who does market making between NPV tokens and ETH. There are two classes of token holders who would use this application. Investors are the first. They can use Delorean for invoice factoring to get liquidity for consumption without losing their exposure to the underlying token. The company or DAO behind the protocol token is the second. It may wish to access future cash flow to invest in development and marketing.

Overnight lending

Overnight and short term lending is a common activity in traditional finance. The commercial paper and repo lending markets are examples of short term lending that has low interest rate with a high confidence of repayment. These are not direct analogs to Delorean, but the protocol can be used in similar ways because of its flexible term system and dynamic discount rates. You can use Delorean to borrow from the next few days for a fee well below 1%. This can be used in other protocols, for example to pre-pay insurance premiums or options premiums.

Coupon stripping

A zero coupon bond in traditional finance is a bond which has no interest payments, but pays some principal amount on maturity. You can use Delorean to create zero coupon bonds by taking a set of tokens, selling their yield, and trading the associated debt slice. The debt slice pays out a principal, namely the locked yield generating tokens. This is a zero coupon bond, with a maturity determined by the amount of yield sold. These positions can be traded on a one-off basis, or into an Delorean time swap liquidity pool similar to the one used to trade the yield.

Vesting token buyback

Tokens are often granted to beneficiaries with a vesting schedule. The tokens released on this vesting schedule represent future yield that can be traded on Delorean, assuming the vesting is transferable. The seller wraps his future vesting tokens in a yield slice, and sells it into an Delorean time swap exchange, receiving upfront liquidity for the future tokens. The natural counterparty for this trade is the company or DAO that issued the vesting tokens in the first place, as they can execute a discounted buyback of their own equity.