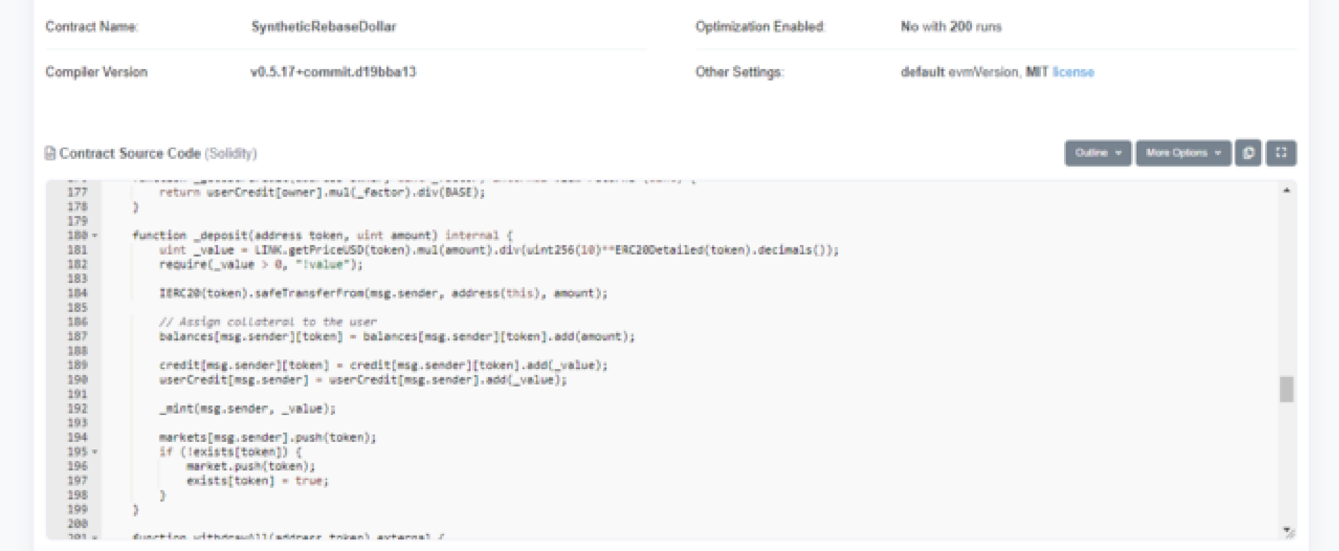
Introducing SyntheticRebaseDollar a credit based rebase index

Andre Cronje
Sep 17 · 2 min read



Contract Name:	SyntheticRebaseDollar	Optimization Enabled:	No with 200 runs
Compiler Version	v0.5.17+commit.d19bba13	Other Settings:	default evmVersion, MIT license

Contract Source Code (Solidity)

```
        return userCredit[owner].mul(_factor).div(BASE);
    }

    function _deposit(address token, uint amount) internal {
        uint _value = LINK.getPriceUSD(token).mul(amount).div(uint256(10)**ERC20Detailed(token).decimals());
        require(_value > 0, "!value");

        IERC20(token).safeTransferFrom(msg.sender, address(this), amount);

        // Assign collateral to the user
        balances[msg.sender][token] = balances[msg.sender][token].add(amount);

        credit[msg.sender][token] = credit[msg.sender][token].add(_value);
        userCredit[msg.sender] = userCredit[msg.sender].add(_value);

        _mint(msg.sender, _value);

        markets[msg.sender].push(token);
        if (!exists[token]) {
            market.push(token);
            exists[token] = true;
        }
    }
```

Full disclosure; we are not yet sure what we want to do with this, but it is cool, and we hope others can build off of it (or find uses we have not yet thought of)

SyntheticRebaseDollar is an auto rebasing index that tracks the dollar value of the collateral that creates it.

Example;

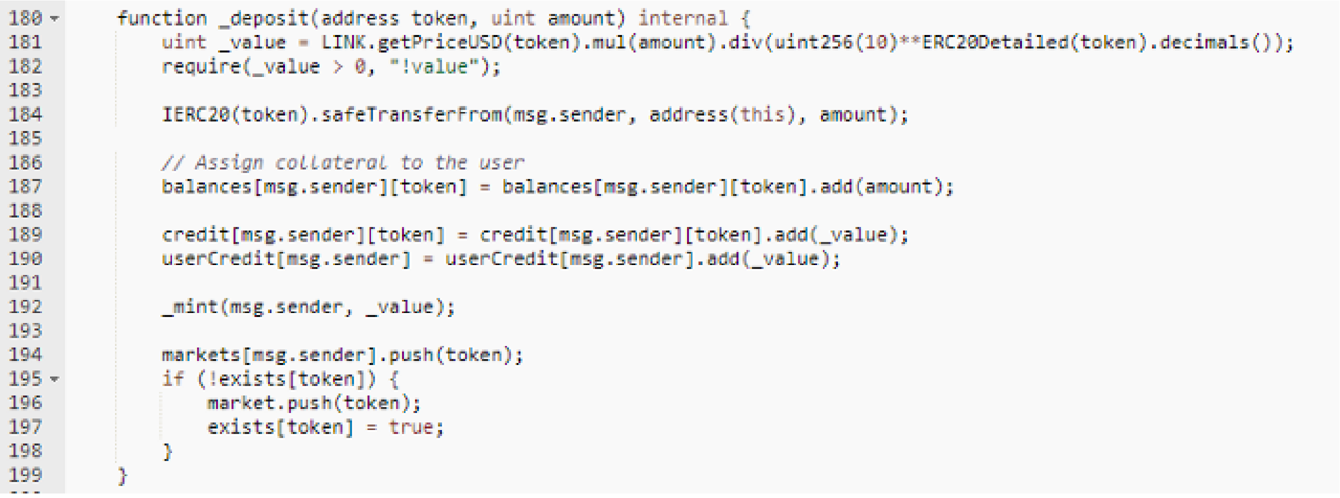
Deposit $100 worth of LINK and you receive 100 srUSD. If the value of LINK increases by +50%, you will have 150 srUSD. There is no rebase trigger, this happens automatically every time the value of underlying collateral changes. There is no need to manually trigger a rebase, values are adjusted in real time (or per block time to be exact).

srUSD is an index based on the collateral provided, current collateral includes;

- LINK, MKR, REN, SNX, YFI
- DAI, SUSD, TUSD, USDC, USDT

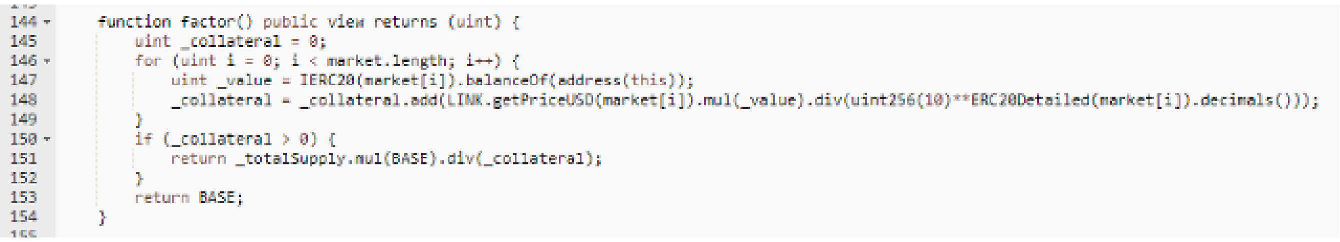
The index is entirely based on deposits. So if the majority of deposits are LINK, MKR, REN, SNX, YFI, then this is essentially a defi index. If it is DAI, SUSD, TUSD, USDC, USDT, it is a stable coin index.

How it works



```
function _deposit(address token, uint amount) internal {
    uint _value = LINK.getPriceUSD(token).mul(amount).div(uint256(10)**ERC20Detailed(token).decimals());
    require(_value > 0, "!value");

    IERC20(token).safeTransferFrom(msg.sender, address(this), amount);

    // Assign collateral to the user
    balances[msg.sender][token] = balances[msg.sender][token].add(amount);

    credit[msg.sender][token] = credit[msg.sender][token].add(_value);
    userCredit[msg.sender] = userCredit[msg.sender].add(_value);

    _mint(msg.sender, _value);

    markets[msg.sender].push(token);
    if (!exists[token]) {
        market.push(token);
        exists[token] = true;
    }
}
```

1. Deposit collateral
2. Calculate USD value of collateral based on Chainlink feeds
3. Mint an amount of srUSD equal to the value in USD



```
function factor() public view returns (uint) {
    uint _collateral = 0;
    for (uint i = 0; i < market.length; i++) {
        uint _value = IERC20(market[i]).balanceOf(address(this));
        _collateral = _collateral.add(LINK.getPriceUSD(market[i]).mul(_value).div(uint256(10)**ERC20Detailed(market[i]).decimals()));
    }
    if (_collateral > 0) {
        return _totalSupply.mul(BASE).div(_collateral);
    }
    return BASE;
}
```

The adjustment factor is calculated in real time. This measures the value of collateral whenever viewed and creates a 1e4 base adjustment factor.

This allows function such as balanceOf() and totalSupply() to adjust in real time to match the value of collateral.

As simple as that.

Here is the contract; 0xaa90d9cc2f41a150489da5fffba2060a2938ffdc

At time of creation this was a prototype part of another ecosystems we are working on;

SyntheticTrader: A permissionless USD settled leveraged long/short synthetic derivative protocol built ontop of StableCredit.

We look forward to suggestions on how to further develop this product.