##### Sync MoM 2022.12.08.

##### Done until now

* **GAMM Balancer code reviewed for improvements**
  + **Suggestions, PRs, possible Issues, and e2e testing with Atomkraft**
  + **Started with Stableswap**

##### Findings/topics for discussion

1. Feature proposal: **Minimize records containing errors from the calculation of:**

* **GetArithmeticTwap()**Define a time span parameter (could be a twap module param) to check if there is a record with no errors (TWAP record younger <= to time parameter span.) If there is no record without errors in the span, use the closest record with error.
* This could minimize the situations when we do calculations with values that are with errors in **ComputeArithmeticTwap**() and **GetInterpolatedRecords**()
* Commit hash containing [PR](https://github.com/osmosis-labs/osmosis/pull/2923) was analyzed, record only inherits the error: b7b4db61f7d8ba15e3680a178092d2c310c77fec  
  <https://github.com/osmosis-labs/osmosis/blob/b7b4db61f7d8ba15e3680a178092d2c310c77fec/x/twap/logic.go#L205-L212>
  + We were aiming at minimizing calculations with errors.
* **getSpotPrices()**This is similar to the possibility to use the last spot price without error in cases when getSpotPrices() returns sp0/sp1 f containing the error. Currently, it is replaced with a 0 value.  
  <https://github.com/osmosis-labs/osmosis/blob/42d73f1cc1c52e85561518be1014b730ef6b7a12/x/twap/logic.go#L47-L59>

***UPDATE:*** there was a [proposal discussion](https://github.com/osmosis-labs/osmosis/issues/2689) and the issue is closed with [PR](https://github.com/osmosis-labs/osmosis/pull/2809).

1. **GAMM specification improvements - PR** [**link**](https://github.com/osmosis-labs/osmosis/pull/3654)
   * Holds changes in existing spec and suggestions for improvements
   * Diagrams for join and exit - basic concepts

*Further suggestions for:*

* ***creating balancer spec*** *- in the same manner as for the stableswap*
* ***Inline comments and func descriptions****: each function that does some calculation holds an explanation and formula*
* ***client/cli/docs*** *- contains .md with explanations and examples of tx for creating pools. Should this be expanded with all the other tx? What about queries?*
* *UPDATE, Dev: currently working on CLI refactoring…*

1. **Minor code improvements and reorganization - in PR** [**link**](https://github.com/osmosis-labs/osmosis/pull/3657)

* **Done**: utils/conv.go -> moved to osmoutils package (was used from incentives, lockup module…)
* **Done**: Create a pool with more elements in SmoothWeightChangeParams.TargetPoolWeights than Weights which could lead to index out of range at:   
  <https://github.com/osmosis-labs/osmosis/blob/42d73f1cc1c52e85561518be1014b730ef6b7a12/x/gamm/client/cli/tx.go#L392>
* **Done**: Defines 2 and 8 as constants:<https://github.com/osmosis-labs/osmosis/blob/b7b4db61f7d8ba15e3680a178092d2c310c77fec/x/gamm/pool-models/balancer/pool_asset.go#L72-L78>
* **Canceled:** MultihopSwapExactAmountIn() Refactoring ideas (Part of code that could be before for loop, …)
  + ***Update***: CHANGED ON MAIN in the meantime: <https://github.com/osmosis-labs/osmosis/blob/b7b4db61f7d8ba15e3680a178092d2c310c77fec/x/gamm/keeper/multihop.go#L20-L27>
* **Done**: Unsafe arithmetic operations during exiting of the pool in wrapping error functions - panics will be propagated

*We have found an issue “fixing” this - not sure this is however correct in any way, so we implemented our changes in the PR.*

* + - ExitSwapExactAmountOut - sharesIn is calculated as Int and later it is being casted to Int64, during the returning of error - which could lead to panicking in the part of the code not wrapped with error

<https://github.com/osmosis-labs/osmosis/blob/fa9293fa9267b1c7a95f645bcc092a8ad4cc03bc/x/gamm/pool-models/balancer/pool.go#L956>

* + - JoinSwapShareAmountOut

<https://github.com/osmosis-labs/osmosis/blob/f52cb5330776c42e92e72c81242da182da012ce3/x/gamm/keeper/pool_service.go#L362-L364>

1. **ValidateFutureGovernor() has flaws - Issue reported:** ValidateFutureGovernor() needs some polishing #[3664](https://github.com/osmosis-labs/osmosis/issues/3664)
2. **GAMM Balancer pools TLA spec and Atomkraft e2e testing :**

Atomkraft e2e test scenarios: [scenarios readme](https://github.com/Ethernal-Tech/OsmosisAtomkraft/blob/main/traces/scenarios/README.md)*(will be transferred to osmosis partnership collaboration repository)*

**Atomkraft findings:**

* ***Scenario8:*** **ExitPool min amount check is calculated incorrectly**
* Traces: traces/scenarios/scenario8.itf.json
* Outcome failed to execute the message - traces executed against osmosis v.12.3.0
* “(Code 7) failed to execute message; message index: 0: Exit pool returned 4uatom,1ujuno , minimum tokens out specified as 4uatom,2ujuno: calculated amount is lesser than min amount”

UPDATE Dev: This is expected. Not an issue.

**Invariant to avoid any malicious attacks:**  
 For any type of rounding error: the pool should profit (as in the example above)

The user should be the one losing money for rounding errors - this is invariant to avoid any malicious attacks.

We have (simply) specified the invariant here in the [TLA spec model](https://github.com/Ethernal-Tech/OsmosisAtomkraft/blob/87ae1c5bc9ecb252376da70231bccd15cf45537b/models/test_mp_gamm.tla#L236-L252)

* ***Scenario9***: **ExitPool can leave with zero assets with low share**
* *Traces*: traces/scenarios/scenario9.itf.json
* Outcome success - but numbers are different than expected - traces executed against osmosis v.12.3.0

Update Dev: : this should not be possible - if it is it is an issue. Report issue.

* ***Scenario15: Swap using a pool via swap-exact-amount-out***
  + Traces: traces/scenarios/scenario15.itf.json
  + Outcome: panic recovered from [osmomath.pow](https://github.com/osmosis-labs/osmosis/blob/61ae9e7587d92279577797921a320d9b60cbab34/osmomath/math.go#L63-L66) func called from [line](https://github.com/osmosis-labs/osmosis/blob/61ae9e7587d92279577797921a320d9b60cbab34/x/gamm/pool-models/balancer/amm.go#L224) - trace executed against osmosis **v.13.1.0**
  + **Explanation: panic occurs in cases when swapping of >= 50% of some token from the pool is tried**
    - **In v.13 - defers are added for panic recovery**
    - **In v.12 - no defers**

Update Dev:

*(Further TLA spec modeling was done for MsgSwapExactAmountIn and   
MsgSwapExactAmountOut)*

1. **Suggestions for structural changes in GAMM module - QUESTIONS**:

*Check this with Dev first or maybe just open the Issue, or communicate through Github*

*Our goal is to try to compare code organization with Stableswap - should be better and newer with the intent to maybe see/create a unified approach*

* + balancer/pool.go: JoinPoolTokenInMaxShareAmountOut() not used - similar to CalcTokenInShareAmountOut()
  + Some functions expect Coins as arguments and check if these Coins contain only 1 element. Other functions, which call them, have only one Coin from which to make a set. Occurrences:
* applySwap: <https://github.com/osmosis-labs/osmosis/blob/b7b4db61f7d8ba15e3680a178092d2c310c77fec/x/gamm/pool-models/balancer/pool.go#L596-L597>
* CalcOutAmtGivenIn: <https://github.com/osmosis-labs/osmosis/blob/b7b4db61f7d8ba15e3680a178092d2c310c77fec/x/gamm/pool-models/balancer/pool.go#L491>
  + this function is called from SwapOutAmtGivenIn, and this execution is called from swapExactAmountIn , where we created Coins set from one single coin: [code](https://github.com/osmosis-labs/osmosis/blob/a562b52872d83d3675c7a80d2ac0c4931ad9da26/x/gamm/keeper/swap.go#L52)

https://github.com/osmosis-labs/osmosis/blob/b7b4db61f7d8ba15e3680a178092d2c310c77fec/x/gamm/pool-models/stableswap/pool.go#L221-L223

* CalcInAmtGivenOut:   
  https://github.com/osmosis-labs/osmosis/blob/b7b4db61f7d8ba15e3680a178092d2c310c77fec/x/gamm/pool-models/balancer/pool.go#L546

1. **What is the expected possible diff in calculations in order to be ok to ignore the LP shares diff in GAMM JoinPoolNoSwap()**

<https://github.com/osmosis-labs/osmosis/blob/42d73f1cc1c52e85561518be1014b730ef6b7a12/x/gamm/keeper/pool_service.go#L234-L238>

1. **The error returned due to introduced changes around *TokenInMaxs* breaking changes listed in** [**breaking\_changes.md**](https://github.com/osmosis-labs/osmosis/blob/main/x/gamm/breaking_changes_notes.md) **is misleading.** Breaking changes:

“*JoinPoolNoSwap*

* *TokenInMaxs must either contain every token in the pool or no tokens*
  + *Before it could just apply a max constraint on one input token.*”

And the error it actually returns in that case: <https://github.com/osmosis-labs/osmosis/blob/f52cb5330776c42e92e72c81242da182da012ce3/x/gamm/keeper/pool_service.go#L400-L404>

1. **QUESTIONS** *(for a better understanding of balancer and GAMM code)* **:**

* CleanupBalancerPool Msg under comments?
* Planned expanding pools should be possible with PokePool tx?

1. **Stableswap pool questions and clarifications**

##### Next Steps

* Stableswap code and specification inspection
* Providing more suggestions for code organization improvements, once we compare the stableswap to the balancer code and spec.
* Report all the issues listed above, and open PRs for smaller things we can fix quickly