##### Sync MoM 2022.12.15

##### Done until now

* GAMM Balancer code reviewed for improvements - finished
* e2e testing with Atomkraft for swaps
* GAMM Stableswap code inspection for issues and possible improvements - finished
  + Suggestions for improvements prepared (Issues or PRs will be open)

##### Findings/topics for discussion

1. **GAMM Issues and PRs reported:**
   * JoinPoolNoSwap contains diff in calculation of the expected shares out and calculated shares out #[3705](https://github.com/osmosis-labs/osmosis/issues/3705)
   * GAMM module and balancer pool model spec improvements #[3706](https://github.com/osmosis-labs/osmosis/issues/3706)
   * x/gamm module minor code improvements #[3704](https://github.com/osmosis-labs/osmosis/pull/3704)
2. **GAMM Balancer pools TLA spec and Atomkraft e2e testing :**

We finished with adaptations of TLA spec (Create, Join, Exit and Swaps are modeled - JoinSwaps and ExitSwaps are omitted for now)

Stableswap: we analyzed the possibility to model it with TLA but due to complex calculations in stableswap code (x＾y e.g.) - we would need to use abstraction to model stableswap.

We suggest first getting feedback from your side regarding atomkraft findings…

1. **Collaboration repo will contain atomkraft and traces -** Should repo stay public or go private?
2. **Atomkraft findings:**

* ***Scenario16:* Swap using a pool via swap-exact-amount-out**

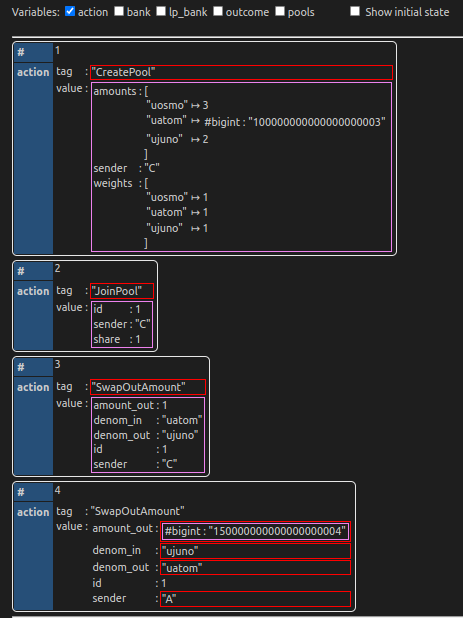
**Outcome:** Two panics:

**Failure**: (Code 111222) recovered: base must be lesser than two

Conclusion: Tx aborted: swap exact amount out for >= 50% of some token from the pool

**Failure**: (Code 111222) recovered: base must be greater than 0

Conclusion: Tx aborted: tried to swap out more than containing in the pool?

****

It seems that checks are missing for:

* tokenInMaxAmount being larger than 50% of the pool
* larger than the pool balance for denom\_out denom.

Panics are from: solveConstantFunctionInvariant -> osmomath.Pow()

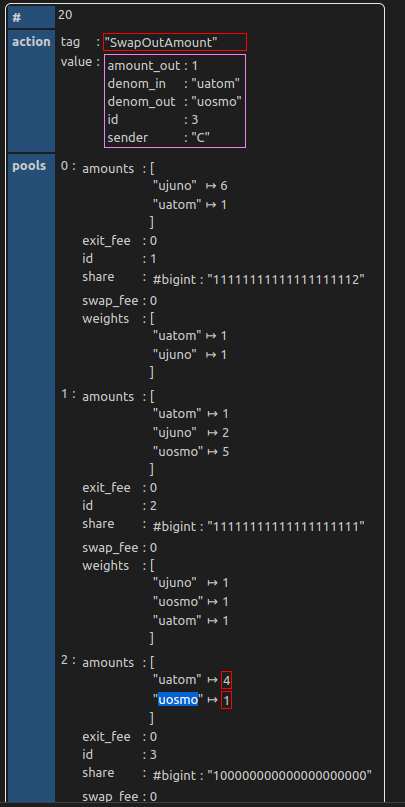
!? Form of issues reported for Atomkraft findings - check with Dev

* ***Scenario17*: Swap using a pool via swap-exact-amount-out**

**Outcome**: panic:  
**Failure**: (Code 111222) recovered: division by zero

...

: panic



**Conclusion**:

Panics are from: solveConstantFunctionInvariant ->....-> decimal.Dec()

<https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/balancer/amm.go#L93-L94>

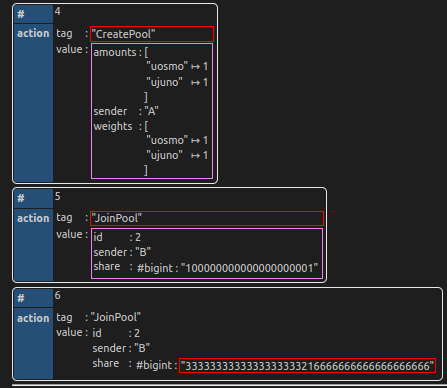
Situation where tokenBalanceFixedAfter is 0  
<https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/balancer/amm.go#L105>

* ***Scenario18 (and Scenario20)*:** Join pool

Outcome:   
Failure: (Code 1) failed to execute message; message index: 0: unexpected error in MaximalExactRatioJoin

It seems that this condition is not satisfied:

<https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/internal/cfmm_common/lp.go#L78-L81>

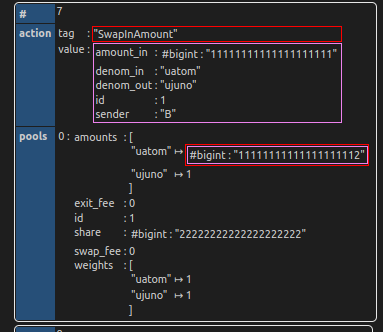


* ***Scenario19 (and Scenario20):*** Swap in amount

Outcome: panic  
Failure: (Code 1) failed to execute message; message index: 0: function swapExactAmountIn failed due to internal reason: base must be greater than 0

...

: panic



It seems that this is panicking due to taking out >50% amount from the pool.

1. **Stableswap spec - minor fixes, issues with pseudo code** - we will openPR

General Impression is that stableswap spec is clearly written. We already shared out suggestions (through Issue #[3706](https://github.com/osmosis-labs/osmosis/issues/3706)) on creating better documentation of implementation (diagrams, connect spec and code) - There is *Code structure* section in the spec - that could be used for that purposes - we will update the existing issue.

1. **Unit Test coverage - could be improved**

We would suggest:

* adding some corner cases, such as **e.g.**
  + 0 <= swapFee < 1
  + 0 <= exitFee < 1
  + 2 <= len(liquidity) <= 8 // liquidity == poolAssets
  + 0 < scaledFactor <= 9223372036854775807  
    1 <= scaledAmount <= 10^34 // = liquidity[denom] / scalingFactors[denom]
* **Increasing Coverage**: currently for balancer and stableswap code is around 70-80% (we will check once again the situation on the main, prior to reporting the issue, this is from v.13.1.0 analysis and v12.3.0)
* **Panic and errors are usually not covered**
* Some functions are also not covered, e.g.
  + - calcPoolSharesInGivenSingleAssetOut() - could be covered in TestCalcSingleAssetInAndOut\_InverseRelationship

1. **Add Gas to stableswap pool’s swap execution function**

Currently the balancer has BalancerGasFeeForSwap defined, and gas is consumed in [applySwap()](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/balancer/pool.go#L594-L595)

Maybe this would be an issue during testing - since this is constant.

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

**Balancer vs Stableswap code**

1. ***CalcJoinPoolNoSwapShares*** - potentially could be refactored to use common function, the only difference in implementations in balancer and stableswap code are the following lines (checks specific to the pool structure):

stableswap/pool.go [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L351-L354) vs. balancer/pool.go [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/balancer/pool.go#L784-L798)

**Suggestion**: implement ensureDenomInPool() func on Pool interface, which would perform specific checks

Btw. GetTotalPoolLiquidity(ctx) - does not need context, should be removed…. (there are more functions like this one) -> This is for a reason! Do not touch this :)

1. ***JoinPoolNoSwap*** - the implementations are practically the same. Only this two functions could be done over an interface:  
   Stableswap [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L246-L253) vs Balancer [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/balancer/pool.go#L285-L291)

**Suggestion**: Checks and part of the implementation could be common for both pool models, and update of the pool could be implemented over the pool interface.

1. ***BinarySearchSingleAssetJoin*** changes - this function defined with [code](https://github.com/ljah8/osmosis/blob/fa9293fa9267b1c7a95f645bcc092a8ad4cc03bc/x/gamm/pool-models/stableswap/amm.go#L437-L441) could be moved to *lp.go* as a standard function.
2. ***BinarySearchSingleAssetJoin*** () and ***SwapAllCoinsToSingleAsset***() - called only for stableswap code

**Suggestion**: if common/lp.go should contain only common functions for both pool model types, maybe we should move this func.

*SwapAllCoinsToSingleAsset -> should be used more by exits*

*This should be helper functions - maybe osmosis will convert balancer code to use the binary search.*

1. ***Refactor stableswap/amm.go joinPoolSharesInternal()***

***First option: OK***

*Since we are returning from if statement code block* [*here*](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/amm.go#L492-L506) *We could move* [*this part*](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/amm.go#L500-L506) *of the code from if - else block and add one more else to execute MaximalExactRatioJoin calculation* [*here*](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/amm.go#L512-L516)

***Second option:***

*Or since, this second part of the function resembles the JoinPoolNoSwap* [*code*](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L378-L387) *- maybe it could be refactored to use this function. Here, we also have the necessary checks implemented in CalcJoinPoolNoSwapShares*() [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L351-L354)

1. **Suggestions for GAMM stableswap code minor improvements:**
2. Stabelswap/pool.go: Move this [part of the code](https://github.com/osmosis-labs/osmosis/blob/v13.0.0/x/gamm/pool-models/stableswap/pool.go#L18-L34) in error.go - it’s all about building errors
3. validateScalingFactors(): This first *scalingFactor == 0* check is unnecessary: [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L445)
4. Stabelswap/pool.go: *getLiquidityIndexMap*() [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L158-L165) - suggestion to refactor it to:
   * getLiquidityIndexByDenom(string) int
   * Or even better create, GetScalingFactorByDenom() - the easily get scaling Factor for liquidity denom.
   * Current usage: [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L135-L154); in case of refactoring we would need to worry about: [Comment](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/pool.go#L157)
5. stableswap/pool.go: applyScalingFactorMultiplier()

Current multiplier is = 1, if the value increases it can create an unwanted overflow situation (big\_num \*2 = small\_num) in [code](https://github.com/osmosis-labs/osmosis/blob/8a7f8e07c6094b40aa546d0e44e24227ec03644c/x/gamm/pool-models/stableswap/pool.go#L487-L494)

* Maybe we should work with bigger Int numbers and check if we can fit in uInt64?

1. stableswap/amm.go: iterKCalculator()

Current implementation returns func with returning BigDec and error values.

func(osmomath.BigDec) (osmomath.BigDec, error)

But then we return nil: [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/amm.go#L272-L279)

If we remove this error, since it is not necessary we need to change osmoutils/binary\_search.go: BinarySearchBigDec() [func](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/osmoutils/binary_search.go#L187)

1. osmoutils/binary\_search.go: BinarySearchBigDec () / BinarySearch()

Move setup base case in for loop, at the beginning: [code](https://github.com/ljah8/osmosis/blob/30460bacef932b03854f59938aa1d29db9a8f99b/osmoutils/binary_search.go#L194-L199), instantiate curEstimate and curOutput prior to the for loop.

1. **QUESTIONS:**

* targetKCalculator is doing subtraction and it seems it can be zero, maybe we are missing smth. ([code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/x/gamm/pool-models/stableswap/amm.go#L252-L263))

CompareBigDec() concerning part of the: [code](https://github.com/osmosis-labs/osmosis/blob/7374795e0de22f3a291ca59c5faffa7851acf3bd/osmoutils/binary_search.go#L126)

##### Next Steps

* Reporting Issues - atomkraft findings and our refactoring suggestions
* Finishing PRs - changes in Stableswap spec and code
* Wrapping everything up in collaboration repository (transfering TLA spec, and Atomkraft Tool)
* For the next phase we could agree to further analyze stableswap (focus on the calculations) and maybe work on issues we will report (the unit tests, refactoring, spec improvements), or go to some other module.
  + Update: We do have our next meeting on 12.01. And will agree on the scope then….