GMM Implementation report

SOLUTION DEVELOPMENT.

- **Platform Selection.**

There are several reasons why Uniswap V2 was chosen as the foundation for developing a solution to the problem at hand.

- Uniswap V2 is one of the most widely used and popular DEXs in the ecosystem, which means it has a large user base and a broad community familiar with its operation.

- The CPMM model implemented in Uniswap V2 is simple, making it ideal for introducing modifications.

- It is highly compatible with ERC-20 tokens and the Ethereum network, ensuring that any development on top of Uniswap V2 is broadly interoperable within this ecosystem.

- The extensive documentation and available information about this platform greatly facilitates development.

- The stability of Uniswap V2 makes it an excellent option for implementing enhancements and modifications.

- Any improvements and modifications can be scaled to other versions of Uniswap, such as Uniswap V3.

- Being open-source makes development even more accessible.

- **Liquidity and Fee Management.**

The first issue addressed was liquidity management. This is due to the fact that Uniswap V2, with its CPMM formula, only requires the liquidity within its own AMM, so no further considerations are needed. However, to implement a more global solution, it is necessary to query the liquidity of other AMMs.

This does not present major concerns, as long as a smart contract has access to the addresses of other AMMs—it can query their liquidity without any problem or additional cost, since querying does not alter the blockchain.

This has been developed through a function that allows adding the addresses of other AMMs, enabling a smart contract to take into account the liquidity of as many AMMs as desired, in addition to its own. Once the addresses are in the list, a simple loop is used to average the external liquidity and sum up the total.

Fees will be set at 0.3%, as in Uniswap V2. This will be done simply by subtracting 0.3% from the user based on what the formula calculates, if the user provides an amount of one cryptocurrency and the program must calculate how much of the other they should receive, or by adding that 0.3% to the user, if the user requests a specific amount of one cryptocurrency and the program must calculate how much they need to pay.

- Application of the Formula.

To compute the output amount in a swap, the following formula is used:

- $\min(y / (x + \Delta x), yi / (xi + \Delta x)) * \Delta x$

Here, x and y represent the reserves of the token pair on the main platform, while xi and yi are the reserves on an external platform. Δx is the amount of the input token provided by the user. The formula calculates the output by taking the smaller exchange rate between

the two sources and multiplying it by the input amount. This ensures a conservative and stable output based on the most limiting liquidity condition.

Example 1: Minimum comes from the main AMM

Let’s say a user wants to swap 10 units of token X.

- Main AMM reserves: x = 100, y = 200
- External AMM reserves: xi = 100, yi = 400

We compute:

- y / (x + Δx) = 200 / (100 + 10) = 200 / 110 ≈ 1.82
- yi / (xi + Δx) = 600 / (200 + 10) = 600 / 210 ≈ 2.86

The minimum is 1.82 → Output = 1.82 * 10 = 18.18 tokens Y

Example 2: Minimum comes from the external AMM

Now consider:

- Main AMM reserves: x = 100, y = 200
- External AMM reserves: xi = 150, yi = 100

We compute:

- y / (x + Δx) = 200 / (100 + 10) = 200 / 110 ≈ 1.82

- $y_i / (x_i + \Delta x) = 300 / (250 + 10) = 300 / 260 \approx 1.15$

  The minimum is 1.15 → Output = 1.15 * 10 = 11.5 tokens Y

  - **Functional Requirements.**

Identifier	RF-01
Description	The system must be able to calculate the exchange rates between two tokens.
Type	Functional.
Priority	High.

Identifier	RF-02
Description	The system must be able to perform token swaps.
Type	Functional.
Priority	High.

Identifier	RF-03
Description	The system must be able to manage liquidity across multiple platforms.

Type	Functional.
Priority	High.

Identifier	RF-04
Description	Each smart contract must be able to query the liquidity of other platforms.
Type	Functional.
Priority	High.

Identifier	RF-05
Description	Each smart contract must have the addresses of the others.
Type	Functional.
Priority	High.

Identifier	RF-06
Description	The system must be able to calculate the exchange rates between two tokens.
Type	Functional.
Priority	High.

Identifier	RF-07
Description	GMM formula must be implemented.
Type	Functional.
Priority	High.

Identifier	RF-08
Description	The system must be compatible with all ERC20 tokens.
Type	Functional.
Priority	High.

Identifier	RF-09
Description	The system must be integrated into Uniswap V2.
Type	Functional.
Priority	Media.

Identifier	RF-10
Description	The system must manage the fees for each transaction.
Type	Functional.
Priority	High.

Identifier	RF-11
Description	The system must charge transaction fees in ETH.
Type	Functional.
Priority	High.

- **Non-Functional requirements.**

Identifier	RNF-01
Description	The system must be capable of handling a high volume of transactions.
Type	Non-Functional.

Priority	High.

Identifier	RNF-02
Description	Smart contracts must be secure.
Type	Non-Functional.
Priority	High.

Identifier	RNF-03
Description	Smart contracts must be optimized to minimize gas costs on the Ethereum network.
Type	Non-Functional.
Priority	Medium.

Identifier	RNF-04
Description	The user interface must be intuitive and accessible.
Type	Non-Functional.
Priority	Medium.

Identifier	RNF-05
Description	The smart contract must be well-structured.
Type	Non-Functional.

Priority	Medium.

Identifier	RNF-06
Description	The smart contract must be efficient.
Type	Non-Functional.
Priority	High.

Identifier	RNF-07
Description	The system must consider applicable regulations across different jurisdictions.
Type	Non-Functional.
Priority	High.

Identifier	RNF-08
Description	The system must handle exceptions in a way that no data is compromised in case of an error.
Type	Non-Functional.
Priority	High.

RESULTS

- Arbitrage.

The problem for which this solution is proposed is arbitrage. In order to observe if the proposed solution eliminates or minimizes this issue, a comparison can be made between the results, given by the cryptocurrency exchange of two users on two different platforms, with four liquidity sources being considered, where the first user makes exchanges at moments when the exchange rate is unfavorable to them, and the second user does so at the moments that are most beneficial for them.

For example, in one case, you can start with what is shown next.

User1	Token A (Both)	Token B (modified)
User 1	10	30
User 2	10	30
Plataform 1	100	400
Plataform 2	100	400
Plataform 3	135	430
Plataform 4	90	420

Now, the users will make the following transactions:

- Transaction 1: User 1 buys 5 tokens A from platform 1.
- Transaction 2: User 2 buys 2.5 tokens A from platform 2.

- Transaction 3: User 2 sells 2.5 tokens A to platform 1.

- Transaction 4: User 1 sells 2.5 tokens A to both platforms.

The results are shown below. The first table shows, for User 1, the amount of token A, token B using the original Uniswap V2, and token B using the proposed solution. The second table shows the same data but this time for User 2.

User1	Token A (both)	Token B (modified)	Token B (original)
Start	10	30	30
Transaction 1	15	10,36	8,89
Transaction 2	15	10,36	8,89
Transaction 3	15	10,36	8,89
Transaction 4	10	29,77	29,34

User2	Token A (both)	Token B (modified)	Token B (original)
Start	10	30	30
Transaction 1	10	30	30
Transaction 2	12,5	20,26	19,71
Transaction 3	10	30,02	30,47
Transaction 4	10	30,02	30,47

As can be seen in the tables, first, User 1, when buying 5 tokens A on Platform 1, simply exchanges them for the corresponding amount of tokens B. It can be observed that the amount is essentially the same in both the original and the proposed solution.

In the next two transactions, first, User 2 buys 2.5 tokens A on Platform 2, which still holds the initial tokens, and then sells them to Platform 1. As User 1 had previously purchased tokens A, these tokens now have a lower proportion than on Platform 2. Here, improvements can already be seen, as in the original, User 2 earns 0.47 token B, while in the proposed solution, they only earn 0.02, which is less than the fees charged.

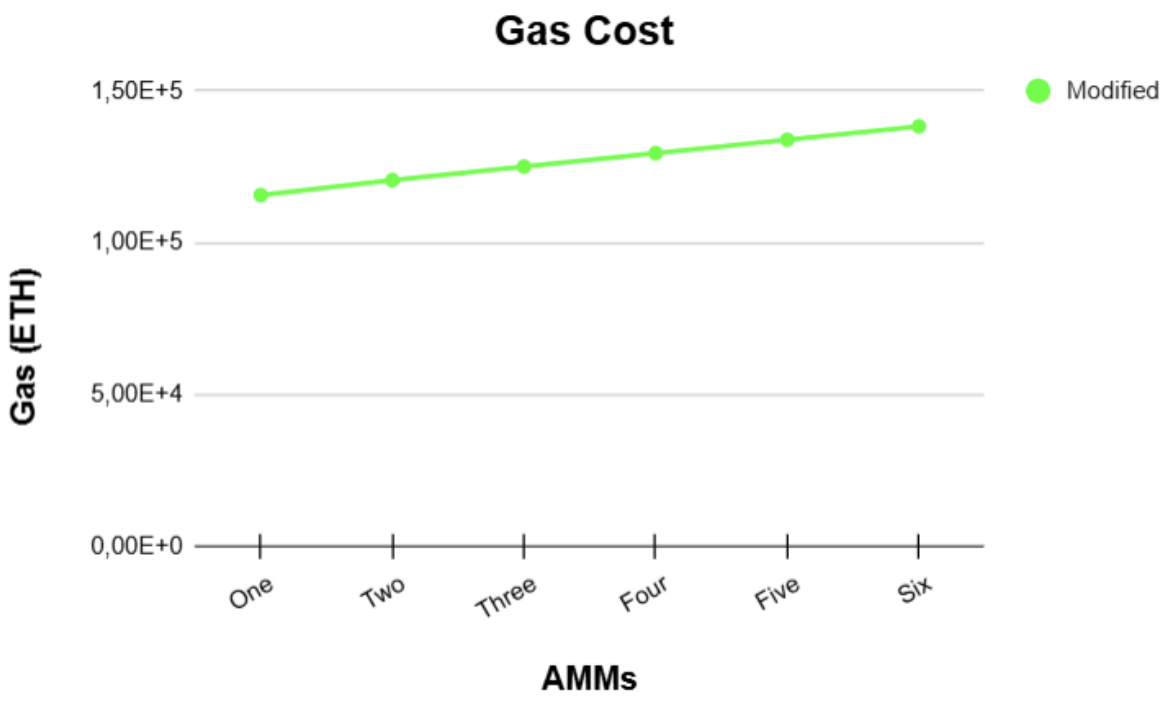
Finally, in Transaction 4, User 1 tries to recover the initial tokens by selling their tokens A. However, since User 2 has already sold them, the proportion between the two platforms equalizes, resulting in a loss of 0.66 tokens B in the original case, while in the proposed case, only a loss of 0.33 occurs.

To summarize, it can be observed that arbitrage is eliminated, as User 2 earns 0.47 token B in the original case, while in the proposed case, they only earn 0.02, which is less than the fees charged.

Gas Cost.

"Gas is the term for the amount of ether (ETH), the native cryptocurrency of Ethereum, required by the network for a user to interact with it. These fees are used to compensate Ethereum miners for the energy needed to verify a transaction and to provide a layer of security to the Ethereum network by making it too costly for malicious users to spam the network."

This is why it is important that it does not increase too much, as it would mean extra costs for users with every transaction they make. The following is an approximate graph, which corresponds to the cost of one transaction when involving one to six AMMs. This helps to observe how the gas cost evolves.



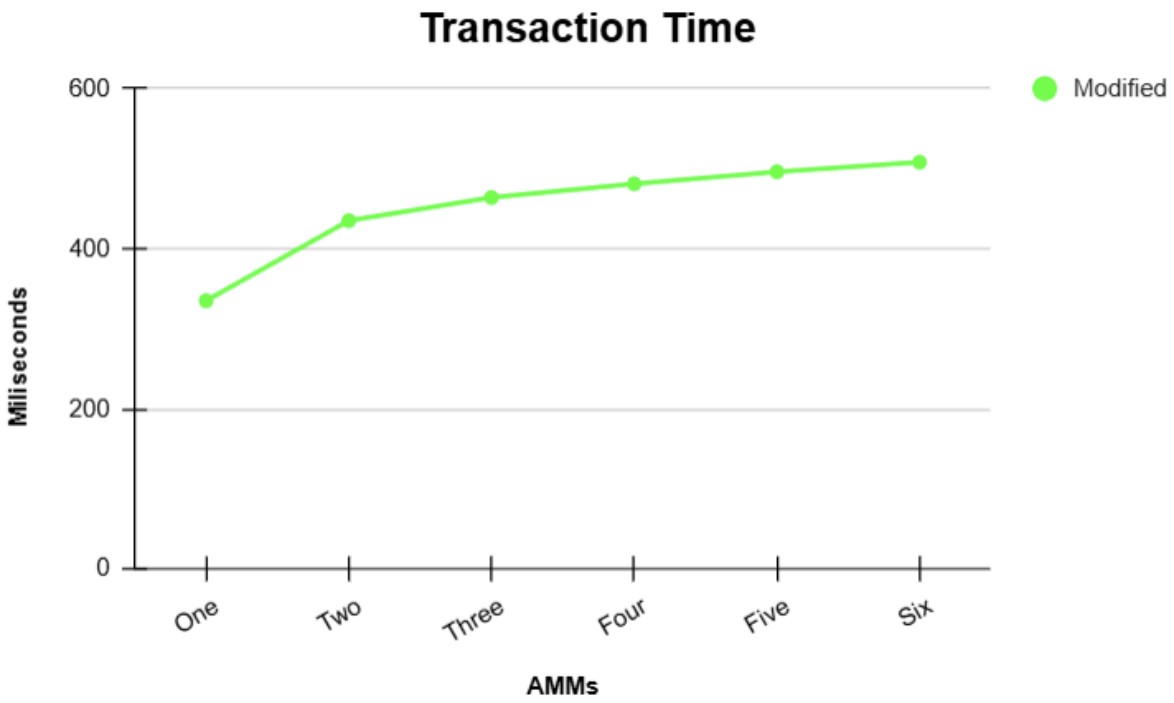
As shown, as more AMMs are taken into account, the gas cost increases, but not in a disproportionate way. Moreover, it can be adjusted since any number of AMMs can be considered. This means that the creator of each smart contract can choose the extent to which they prefer to address arbitrage or prioritize minimizing gas costs.

In summary, from the perspective of gas expenditure, it is viable, as the cost grows very slowly.

- Efficiency.

Another important aspect is code efficiency, since the calculations must be carried out across many nodes (computers) to validate a transaction. Therefore, any increase in computation time will affect all nodes.

The following figure shows a graph of the time it takes to execute a transaction of the proposed solution. In this context, the exact execution times are not particularly important, but rather how they grow. Since the same equipment is used for all tests, we can reliably assess the relative growth in computation time.



As shown, computation times increase slightly between using a single AMM and two AMMs, as this is where the most significant change occurs. However, beyond that, the increase is minimal each time liquidity from another AMM is added. Since the growth is barely noticeable, from an efficiency standpoint, this project is indeed viable. Using an excessively large number of AMMs might introduce some issues, but there's no practical reason to reach such high numbers.

Lessons learned

The prototype implementation of the Global Market Maker (GMM) on top of UniswapV2 has exposed a set of practical insights that may guide future researchers attempting to integrate blockchain-wide liquidity into AMMs. Below we summarise what worked well, what proved difficult, and where open questions remain.

1. Start on a simple, well-known AMM.
Uniswap V2's constant-product code is short, clear, and heavily audited, so adding new logic was quick. Porting the same idea to newer designs such as Uniswap V3 or Curve would take much more engineering. Future work should begin on a simple base, then migrate once core ideas are proven.

2. Reading outside reserves is cheap but finding pools is not.
A contract can read reserves from other pools with no extra gas. The hard part is keeping a reliable list of pool addresses. Manual whitelists work in tests but will not scale. A public registry or oracle that lists active pools is the next step.

3. The "minimum-of-two" pricing rule works.
Quoting the lower of (i) the local CPMM price and (ii) the global-reserve price keeps the pool from being drained and removes arbitrage in simple two-pool tests. The rule is easy to code and keeps the familiar CPMM price curve. It still needs a full game-theory study for many pools with different fees.

4. Gas cost rises slowly—up to a point.
Gas and run time grow roughly linearly with the number of pools checked. Up to six pools the added cost is small; beyond that the benefit falls off. An adaptive design that only queries the pools that matter most could keep costs low.

5. Arbitrage drops, but fee rules may need tuning.
Simulations show that most arbitrage profit disappears and the savings are shared between traders and liquidity providers. The fixed 0.3 % fee from Uniswap V2 may no longer be ideal. A fee that adjusts to the new depth might share value even better.

6. External data adds new risks.
Depending on outside reserve data creates oracle risk. The contract also needs a full check for flash-loan edge cases. A formal audit and a public bug bounty are advised before a main-net launch.

7. Strong test tools are essential.
Small "toy" trades help build intuition, but only large random test batches revealed rounding and fee-tracking bugs. A script that sends many varied trades, including adversarial ones, was key to finding corner cases.

Overall, GMM shows that a global-liquidity view can cut arbitrage and slippage without major cost, but it needs better pool discovery, careful cost control, and thorough security checks before production use.

Test Vectors

Test	x	y	x_i	y_i	Δx	Expected Result = min(y / (x + Δx), y_i / (x_i + Δx)) * Δx
1	1445	275	1195	1357	67	min(275 / 1512, 1357 / 1262) * 67 = 12.19
2	1173	835	1924	1236	68	min(835 / 1241, 1236 / 1992) * 68 = 42.19
3	1149	160	322	228	135	min(160 / 1284, 228 / 457) * 135 = 16.82
4	361	1023	1461	618	140	min(1023 / 501, 618 / 1601) * 140 = 54.04
5	918	331	787	592	53	min(331 / 971, 592 / 840) * 53 = 18.07
6	1412	625	679	410	114	min(625 / 1526, 410 / 793) * 114 = 46.69
7	1065	1763	1748	638	108	min(1763 / 1173, 638 / 1856) * 108 = 37.12
8	1036	170	1995	1058	65	min(170 / 1101, 1058 / 2060) * 65 = 10.04
9	1145	1350	337	414	89	min(1350 / 1234, 414 / 426) * 89 = 86.49
10	304	1477	1341	1347	128	min(1477 / 432, 1347 / 1469) * 128 = 117.37
11	339	262	1211	440	51	min(262 / 390, 440 / 1262) * 51 = 17.78
12	1705	1294	1696	600	126	min(1294 / 1831, 600 / 1822) * 126 = 41.49
13	1294	1465	495	1014	75	min(1465 / 1369, 1014 / 570) * 75 = 80.26
14	574	689	149	1529	123	min(689 / 697, 1529 / 272) * 123 = 121.59
15	1008	1686	1178	1577	50	min(1686 / 1058, 1577 / 1228) * 50 = 64.21
16	765	1714	664	842	105	min(1714 / 870, 842 / 769) * 105 = 114.97
17	368	1503	528	695	107	min(1503 / 475, 695 / 635) * 107 = 117.11
18	1186	1179	1822	1365	68	min(1179 / 1254, 1365 / 1890) * 68 = 49.11
19	111	1892	1275	1440	90	min(1892 / 201, 1440 / 1365) * 90 = 94.95

20	1204	1345	1670	1645	82	min(1345 / 1286, 1645 / 1752) * 82 = 76.99
21	1026	1086	1541	974	147	min(1086 / 1173, 974 / 1688) * 147 = 84.82
22	938	469	485	1438	79	min(469 / 1017, 1438 / 564) * 79 = 36.43
23	1097	579	1893	1324	141	min(579 / 1238, 1324 / 2034) * 141 = 65.94
24	1975	730	1210	1867	139	min(730 / 2114, 1867 / 1349) * 139 = 48.00
25	479	541	1428	1715	58	min(541 / 537, 1715 / 1486) * 58 = 58.43
26	1311	1878	1292	130	116	min(1878 / 1427, 130 / 1408) * 116 = 10.71
27	1063	1015	971	1404	101	min(1015 / 1164, 1404 / 1072) * 101 = 88.07
28	1043	1348	941	1537	144	min(1348 / 1187, 1537 / 1085) * 144 = 163.53
29	1598	1013	551	1360	122	min(1013 / 1720, 1360 / 673) * 122 = 71.85
30	1058	623	1021	1627	141	min(623 / 1199, 1627 / 1162) * 141 = 73.26
31	1636	467	824	1659	86	min(467 / 1722, 1659 / 910) * 86 = 23.32
32	695	126	989	1004	81	min(126 / 776, 1004 / 1070) * 81 = 13.15
33	1869	857	274	408	79	min(857 / 1948, 408 / 353) * 79 = 34.76
34	1256	666	702	1625	74	min(666 / 1330, 1625 / 776) * 74 = 37.06
35	985	1635	1599	1499	133	min(1635 / 1118, 1499 / 1732) * 133 = 115.11
36	1276	1274	1701	1121	82	min(1274 / 1358, 1121 / 1783) * 82 = 51.55
37	1324	1050	300	1087	112	min(1050 / 1436, 1087 / 412) * 112 = 81.89
38	413	314	753	1762	109	min(314 / 522, 1762 / 862) * 109 = 65.57
39	642	851	1381	1802	83	min(851 / 725, 1802 / 1464) * 83 = 97.42
40	1310	1260	1278	1018	132	min(1260 / 1442, 1018 / 1410) * 132 = 95.30
41	1444	330	632	1744	61	min(330 / 1505, 1744 / 693) * 61 = 13.38
42	226	1597	385	1739	139	min(1597 / 365, 1739 / 524) * 139 = 461.30
43	1559	1333	283	1489	61	min(1333 / 1620, 1489 / 344) * 61 = 50.19

44	1268	219	1451	1663	98	min(219 / 1366, 1663 / 1549) * 98 = 15.71
45	1182	892	1181	594	150	min(892 / 1332, 594 / 1331) * 150 = 66.94
46	836	981	596	1468	115	min(981 / 951, 1468 / 711) * 115 = 118.63
47	361	214	129	893	82	min(214 / 443, 893 / 211) * 82 = 39.61
48	409	129	1969	315	117	min(129 / 526, 315 / 2086) * 117 = 17.67
49	1976	1392	1609	1077	93	min(1392 / 2069, 1077 / 1702) * 93 = 58.85
50	1270	991	1907	1953	117	min(991 / 1387, 1953 / 2024) * 117 = 83.60
51	1183	904	1378	711	60	min(904 / 1243, 711 / 1438) * 60 = 29.67
52	1555	1768	928	1712	62	min(1768 / 1617, 1712 / 990) * 62 = 67.79
53	472	1138	1425	1959	53	min(1138 / 525, 1959 / 1478) * 53 = 70.25
54	1928	1036	512	1928	65	min(1036 / 1993, 1928 / 577) * 65 = 33.79
55	978	1649	731	104	89	min(1649 / 1067, 104 / 820) * 89 = 11.29
56	1162	1600	917	1474	88	min(1600 / 1250, 1474 / 1005) * 88 = 112.64
57	1318	1616	575	1825	107	min(1616 / 1425, 1825 / 682) * 107 = 121.34
58	1161	291	1111	1600	132	min(291 / 1293, 1600 / 1243) * 132 = 29.71
59	1983	368	814	842	76	min(368 / 2059, 842 / 890) * 76 = 13.58
60	1497	1188	762	931	147	min(1188 / 1644, 931 / 909) * 147 = 106.23
61	496	1062	440	2000	86	min(1062 / 582, 2000 / 526) * 86 = 156.93
62	849	1769	105	1467	138	min(1769 / 987, 1467 / 243) * 138 = 247.34
63	1007	413	591	579	143	min(413 / 1150, 579 / 734) * 143 = 51.36
64	1879	396	305	932	90	min(396 / 1969, 932 / 395) * 90 = 18.10
65	500	562	361	123	96	min(562 / 596, 123 / 457) * 96 = 25.84
66	249	1369	1513	331	118	min(1369 / 367, 331 / 1631) * 118 = 23.95
67	1634	1438	907	1397	138	min(1438 / 1772, 1397 / 1045) * 138 = 111.99
68	1846	1473	1054	1875	92	min(1473 / 1938, 1875 / 1146) * 92 = 69.93
69	1275	1656	928	1991	65	min(1656 / 1340, 1991 / 993) * 65 = 80.33

70	385	257	422	1423	121	min(257 / 506, 1423 / 543) * 121 = 61.46
71	1821	870	1239	346	122	min(870 / 1943, 346 / 1361) * 122 = 31.02
72	986	839	897	1436	65	min(839 / 1051, 1436 / 962) * 65 = 51.89
73	548	1763	475	995	126	min(1763 / 674, 995 / 601) * 126 = 208.60
74	1563	206	1969	907	89	min(206 / 1652, 907 / 2058) * 89 = 11.10
75	1416	127	124	1164	136	min(127 / 1552, 1164 / 260) * 136 = 11.13
76	405	1035	1630	762	126	min(1035 / 531, 762 / 1756) * 126 = 54.68
77	338	536	801	924	146	min(536 / 484, 924 / 947) * 146 = 142.45
78	857	170	1251	256	110	min(170 / 967, 256 / 1361) * 110 = 19.34
79	702	926	572	569	110	min(926 / 812, 569 / 682) * 110 = 91.77
80	885	173	1189	175	97	min(173 / 982, 175 / 1286) * 97 = 13.20
81	535	616	1925	944	56	min(616 / 591, 944 / 1981) * 56 = 26.69
82	733	960	1771	1297	116	min(960 / 849, 1297 / 1887) * 116 = 79.73
83	639	608	1366	520	101	min(608 / 740, 520 / 1467) * 101 = 35.80
84	173	409	741	1858	86	min(409 / 259, 1858 / 827) * 86 = 135.81
85	567	1804	1954	770	86	min(1804 / 653, 770 / 2040) * 86 = 32.46
86	248	1451	182	329	62	min(1451 / 310, 329 / 244) * 62 = 83.60
87	1550	1525	1398	1964	103	min(1525 / 1653, 1964 / 1501) * 103 = 95.02
88	346	1375	1793	1364	149	min(1375 / 495, 1364 / 1942) * 149 = 104.65
89	610	793	702	1271	89	min(793 / 699, 1271 / 791) * 89 = 100.97
90	1646	961	582	583	96	min(961 / 1742, 583 / 678) * 96 = 52.96
91	1718	666	949	771	54	min(666 / 1772, 771 / 1003) * 54 = 20.30
92	314	617	845	892	116	min(617 / 430, 892 / 961) * 116 = 107.67
93	458	419	1915	878	55	min(419 / 513, 878 / 1970) * 55 = 24.51
94	1819	1632	887	1684	133	min(1632 / 1952, 1684 / 1020) * 133 = 111.20

95	176	304	1594	184	116	min(304 / 292, 184 / 1710) * 116 = 12.48
96	1683	102	1260	452	148	min(102 / 1831, 452 / 1408) * 148 = 8.24
97	1056	689	1098	678	91	min(689 / 1147, 678 / 1189) * 91 = 51.89
98	330	1812	830	1202	86	min(1812 / 416, 1202 / 916) * 86 = 112.85
99	1640	1806	245	1571	127	min(1806 / 1767, 1571 / 372) * 127 = 129.80
100	726	1864	407	1722	77	min(1864 / 803, 1722 / 484) * 77 = 178.74
101	1000	1000	1000	1000	100	min(1000 / 1100, 1000 / 1100) * 100 = 90.91
102	1000	1000	800	1000	100	min(1000 / 1100, 1000 / 900) * 100 = 90.91
103	1000	1000	2000	500	200	min(1000 / 1200, 500 / 2200) * 200 = 45.45
104	500	2000	1000	3000	50	min(2000 / 550, 3000 / 1050) * 50 = 181.81
105	2000	500	1000	1500	100	min(500 / 2100, 1500 / 1100) * 100 = 23.81

In this test vector, x and y refer to the tokens in the main platform, while x_i and y_i refer to the total amount of those tokens across all the platforms considered. Δx represents the variation of x, meaning the amount a user buys or sells.