HSHSP Research Proposal

Evaluating Sharpness: Exploration of Different Operators

**Razan Abdelrahman**

**Mentor: Nathan Haut**

**Department of Computational Mathematics, Science, and Engineering**

**Michigan State University**

**Abstract**

**Introduction**

Sharpness aware minimization (SAM) is a methodology used to enhance the efficiency of deep neural network models in discovering new things by focusing on how sharp the error is [2]. SAM functions decreases fitness sharpness which improves model stability, which then improves the generalization ability of the models. By using SAM in genetic programming, we are able to find which solution best fits *and* is stable. Genetic programming tends to find simpler equations as solutions, but when there isn’t much data, there can be gaps between the data points which make the response unstable with high sharpness levels, so we can use sharpness metrics to identify models that make smoother predictions. There are different ways that make programs run smoother: order of nonlinearity, model curvature, random sampling technique, RelaxGP, overfit repulsors, and SAM [2]. Order of non-linearity and model curvature explore the intrinsic characteristics of the models to measure how likely they are to overfitting. Random sampling techniques lessen overfitting by using multiple data subsets, which reduces exposure to the entire dataset at once. RelaxGP introduces a strategy to relax the model and penalize it for excessively close fits to response data; and overfit repulsors keep track of the models that had mistakes and construct new models different from the mistake prone ones [2]. This study investigates different sharpness techniques with multiple operators to evaluate and characterize how sharpness affects their program performance, and the reason for it.

**Materials and Methods**

In the study conducted, the authors utilized the StackGP system, accessible on GitHub [1], to investigate the sharpness of various different operators. Data ranges of [-1,1], [1,1000], and [100,105] were used to explore variations in operator sharpness across different data conditions. The [-1,1] range tested values near zero, [1,1000] covered a wider range of values, and [100,105] examined sharpness with large values within a narrow range. By incorporating these diverse data ranges, the authors gathered multiple results that were subsequently compared and depicted using bar graphs.

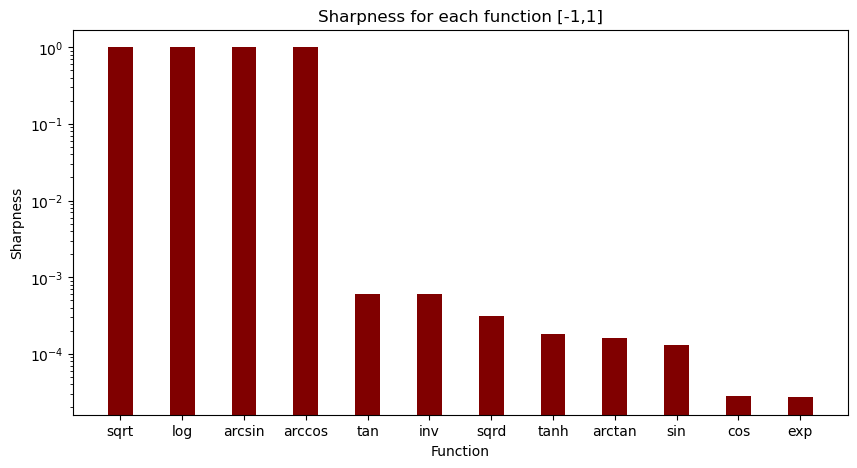
To assess operator sharpness, the study analyzed both single-layer and double-layer sharpnesses. Single-layer sharpness evaluated the performance of individual operators, while double-layer sharpness examined how the sharpness of operators changed when paired with others.

Additionally, variations in percent and numerical perturbations were adjusted to assess how sensitive the operators’ sharpness was to these parameters. Percent perturbation is the percent of the original value that is being changed. Changing the percent perturbation allows for managing how much noise is being introduced to the input data. Changing the number perturbation value by a specific number allows you to specify how many times the input values are changed. Being able to set an exact amount of perturbation can allow us to set a good boundary between accuracy and efficiency. How much you change the number value by can impact the stability and sharpness of the sharpness estimate. Number of perturbations included values of 5, 10, 50, and 100, while percent perturbations ranging from 0 to 1 were tested across all operators.Top of Form

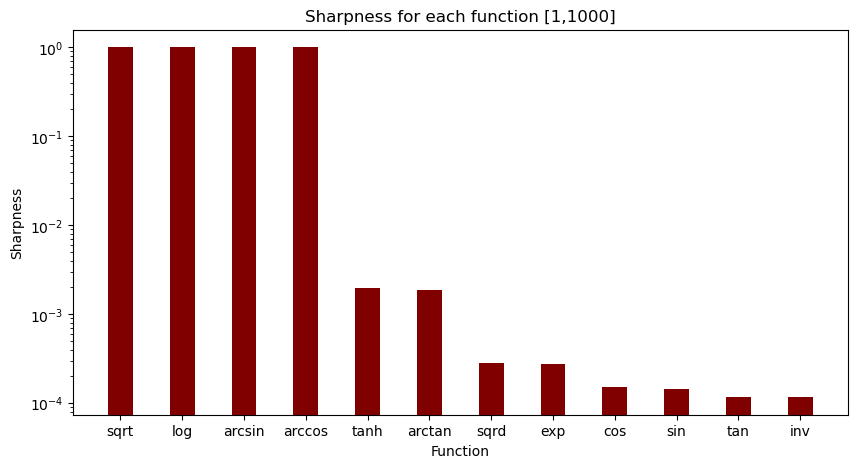
Bottom of Form

The study also delved into the sharpness characteristics of binary operators, namely *addition, subtraction, multiplication,* and *division.* As well, the authors explored the sharpness profiles across two layers within the three distinct ranges: [-1,1], [1,1000], and [100,105]. Each operator was evaluated when paired with various functions, and their respective sharpness measures were visually represented through bar graphs.

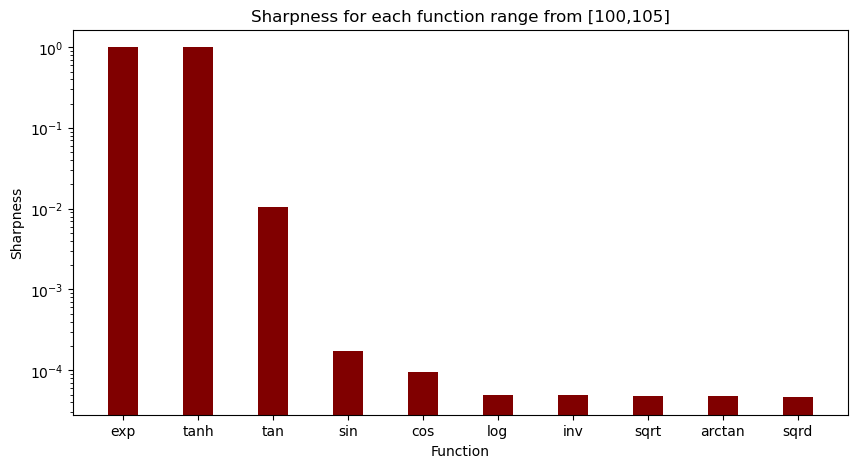
**Results**

****

This bar graph represents the sharpness for the data range [-1,1]. It indicates that the square root, arcsin, arccos, and logarithmic operators are all nan (not a number) values. Square root and logarithmic operators give nan values because they can only function for positive real numbers. Arcsin and arccos are nan because they can only function within a specific range of [-1,1].

****

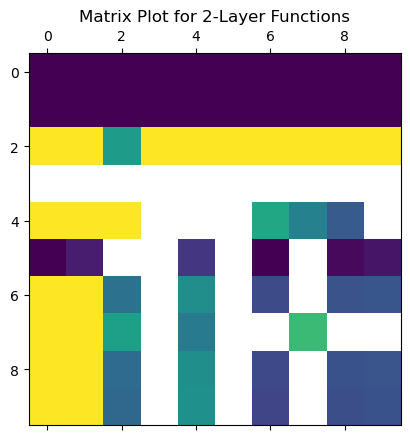
This bar graph shows the sharpness of the operators from the data range [1,1000]. Similar to the sharpness behavior for the data range [-1,1], the square root and logarithmic operators both give nan values because they are unable to run with negative numbers. However, for this data range, the trigonometric operators (sin, cos, tan) have greater stability than the other operators.

****This bar graph illustrates the sharpness across the range [100,105]. Both the exponent and hyperbolic tangent operators diverge to infinity. In contrast, the trigonometric functions (tan, cos, sin) exhibit relatively stable behavior with low sharpness values, while all other operators also demonstrate stable and low sharpness characteristics.

Top of Form

Bottom of Form

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1st Layer Function | sqrt | log | exp | sqrd | inv | cos | sin | tan | arctan | tanh |
| sqrt | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| log | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| exp | 1 | 1 | 0.0033816 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| sqrd | 0.000029706 | 3.63161E-06 | 0.000046892 | 0.0000789 | 2.133E-06 | 6.861E-05 | 2.2975E-05 | 0.00006859 | 0.00001965 | 0.000018798 |
| inv | 1 | 1 | 1 | 2.1398E-06 | 7.71814E-05 | 3.1209E-05 | 0.0070232 | 0.001475 | 0.0007267 | 0.000475527 |
| cos | 0.0000318 | 0.000037243 | 1.9281E-05 | 1.8763E-05 | 5.3882E-05 | 0.000020363 | 3.175E-05 | 0.000016431 | 0.00033151 | 3.51718E-05 |
| sin | 1 | 1 | 0.0005032 | 1.8601E-05 | 0.0018892 | 0.00001742 | 0.0001905 | 0.00011725 | 0.0002182 | 0.0002279 |
| tan | 1 | 1 | 0.0046835 | 0.0000785 | 0.0012163 | 0.000057162 | 0.00008354 | 0.01914 | 0.00007503 | 0.000094817 |
| arctan | 1 | 1 | 0.00044311 | 0.0000177 | 0.0019235 | 0.00001656 | 0.000217 | 0.0000761 | 0.00024177 | 0.000256 |
| tanh | 1 | 1 | 0.00041541 | 0.0000157 | 0.002065 | 0.0000151 | 0.0002289 | 8.183E-05 | 0.000250766 | 0.000261726 |

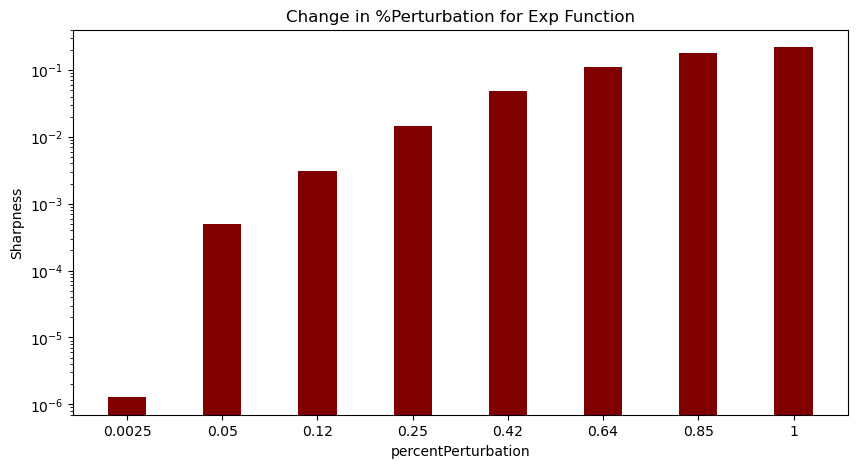


This matrix plot visualizes the sharpness interactions across 2-layers from the table above, comparing each operator with itself and with others. Dark purple squares highlight operators that were initially and consistently broken when paired with others. Yellow squares indicate operators that became broken upon pairing. Green squares denote operators that increased in sharpness, while white squares represent those that decreased in sharpness.

Top of Form

Bottom of Form

**Figure 1:**

Figure 1 represents the changes in percent perturbation within the exponent operator. Percent perturbation is the percent of the original value that is being changed. As you increase the percent perturbation, you are increasing the change from the original input data, which increases sharpness.

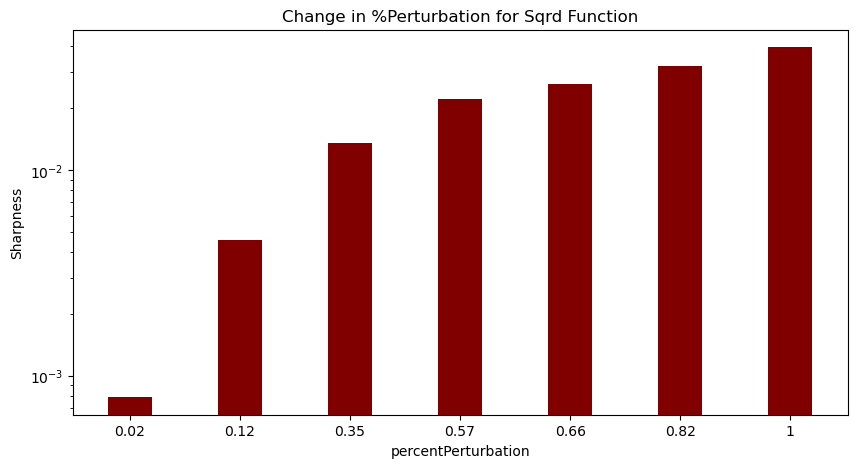
**Figure 1.2:**  

Figure 1.2 represents the different percent perturbations for the squared function. As you increase percent perturbation, you increase the amount of noise present, which increases sharpness of the analysis. This occurs for all of the operators.

**Figure 1.3:**

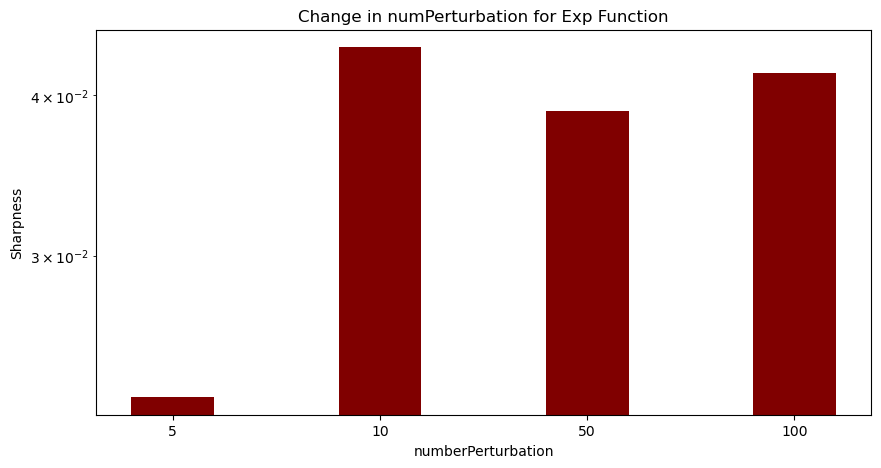


Figure 1.3 represents the change in the number values. The number perturbation goes from low values to higher ones. In this example, the exponent operator increases dramatically in the first change but later becomes stable.

**Figure 1.4**

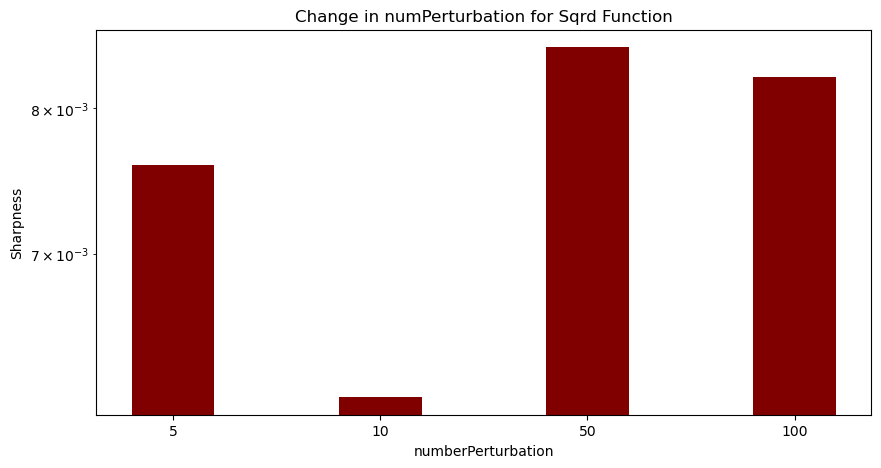


Figure 1.4 shows the different number perturbations for the squared operator. It follows an oscillating pattern. When making large changes to the input values, the sharpness increases and decreases in a wave like pattern, so we can infer it is due to how big the changes are. However, more data needs to be collected in order to make a solid claim.

**Figure 2**

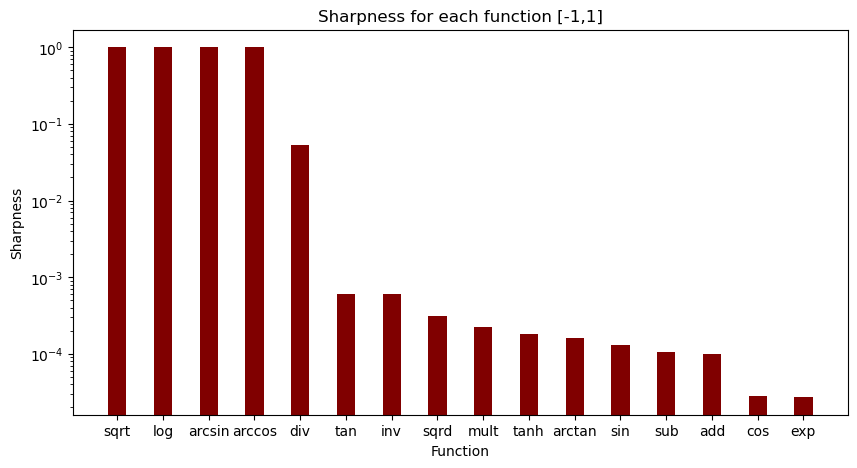


Figure 2 shows the sharpness for each function within a range of [-1,1] along with the binary operators (addition, subtraction, multiplication, division). The division operator has a relatively high sharpness level because in the narrow range of [-1,1], the outcome being 0 is very probable because the denominator has a high possibility of being 0. Multiplication is relatively stable and subtraction and addition are stable probably because of their simplicity.

**Figure 2.1**

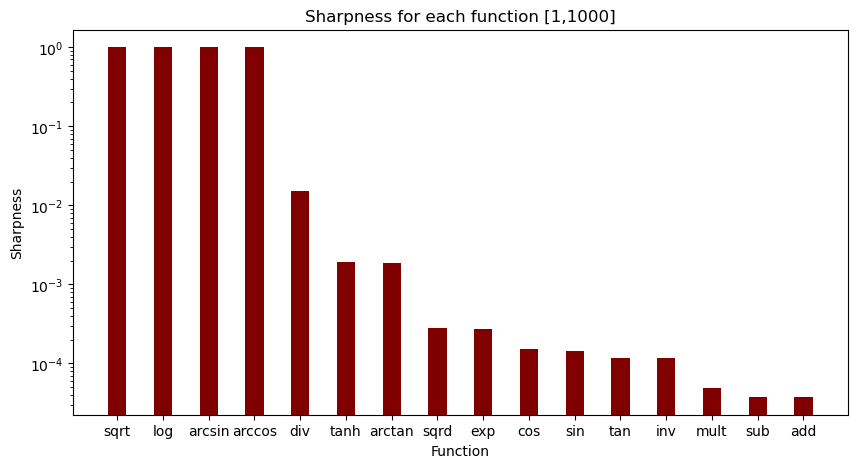


Figure 2.1 shows the sharpness for both unary and binary operators within the range of [1,1000]. Like figure 2, the division operator’s sharpness level is high and unstable, and the other operators have low sharpnesses and are stable.

**Figure 2.2**

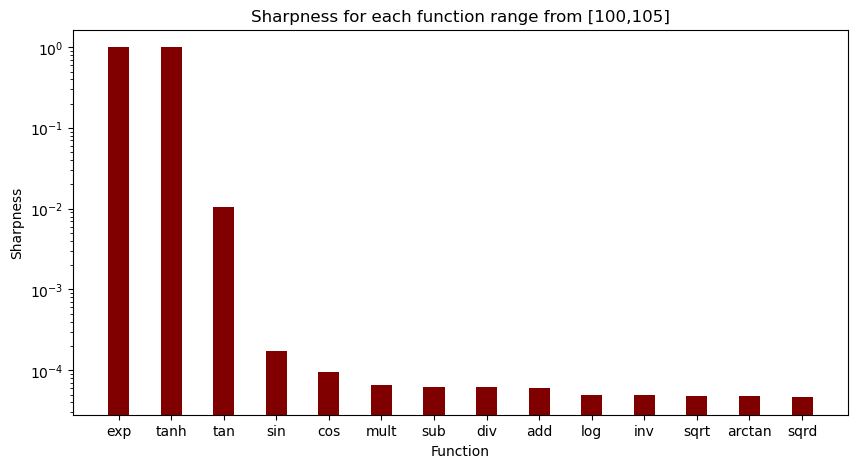


Figure 2.2 illustrates the sharpness for both unary and binary operators across the range [100,105]. Unlike the two other ranges, the [100,105] range depicted the division operator on the low end of the sharpness level. This is likely due to the range being so far from 0, so it probably won’t break (nan). However, now all of the operators seem to be relatively stable with around the same sharpness levels.

**Figure 3**

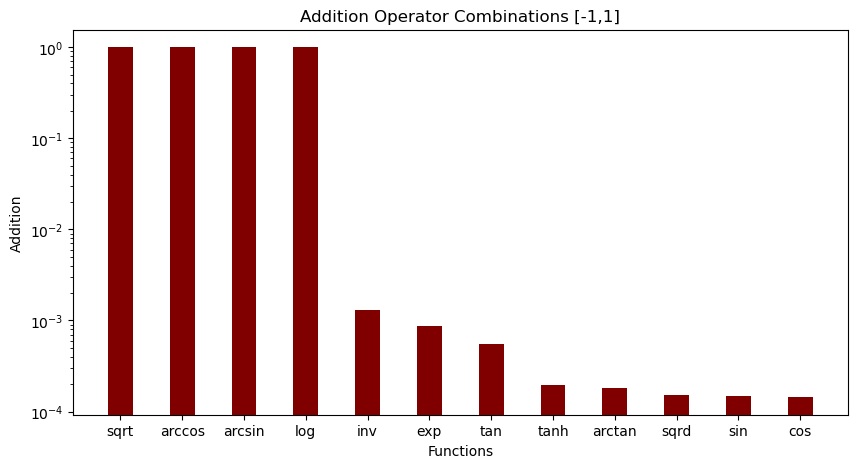


Figure 3 represents the unary operators combined with the addition operator within the range of [-1,1]. Sin and cos seem to be stable and square root, arccos, arcsin, and log are all broken. Sin and cos are stable because their boundaries are predictable since the range is [-1,1].

**Figure 3.1**

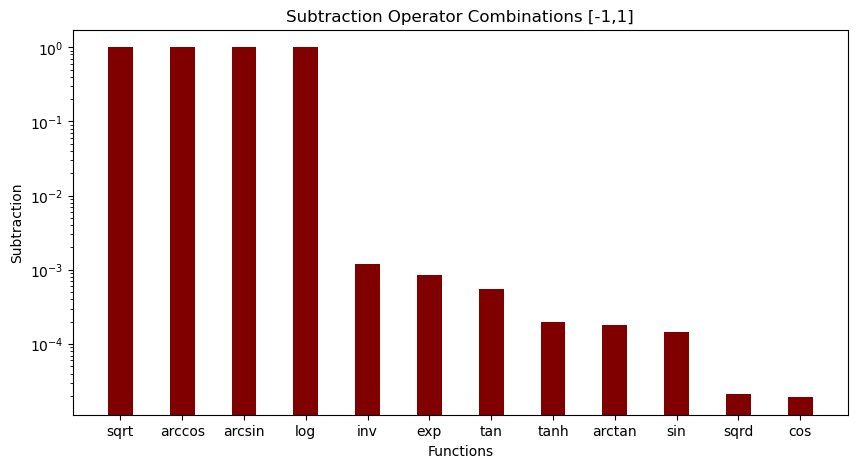
****

Figure 3.1 shows the sharpness for unary operators combined with the subtraction operator. Like the addition combination graph, sin and cos are stable and the first four operators are broken. Squared is probably stable because when you square a number between [-1,1], the result is always positive and stays within the same range, so it most likely keeps it stable.

**Figure 3.2**

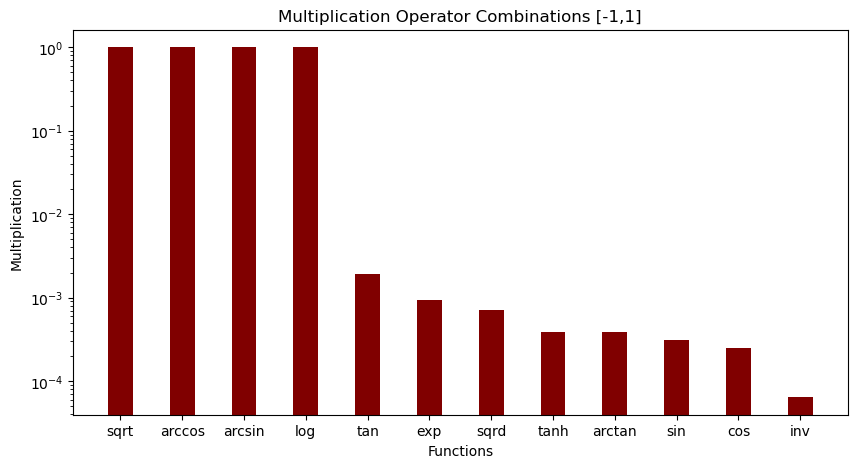
****

Figure 3.2 shows the multiplication operator combined with the unary operators within the range [-1,1]. Unlike the addition and subtraction operators, the multiplication operator shows the inverse function as being the most stable with the lowest sharpness level.

**Figure 3.3**

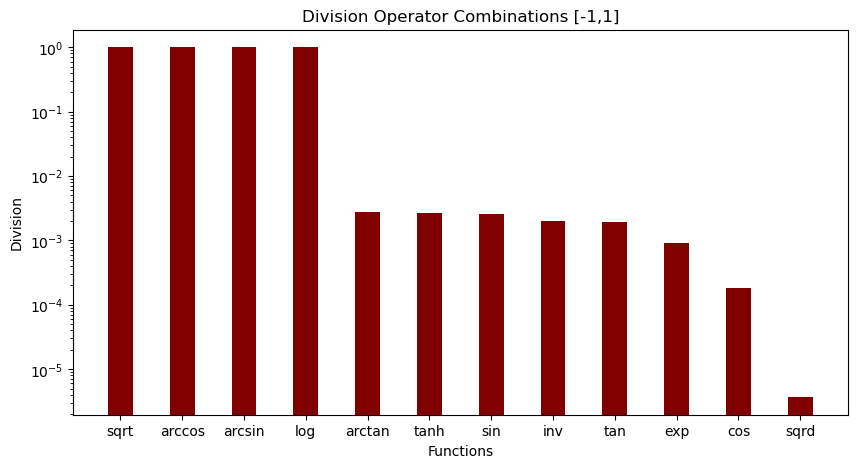
****

Figure 3.3 displays the division operator combinations with the unary operators. The only function that seems to be stable is the squared function. This is probably due to the fact that the results don’t grow too big or small, they are within a reasonable range.

Figures 4 – 4.3 (bar graphs for [1,1000]) **still running**

Figures 5-5.3 (bar graphs for [100,105]) **still running**

**Discussion**

In analyzing the behavior of the sharpness of various mathematical operators under different data ranges, several patterns emerge. The square root and logarithmic operators consistently display broken behavior across both the [-1,1] and [1,1000] ranges. The inverse operator shows variability, spreading across instances of being broken, increasing sharpness, and decreasing sharpness. Being broken means that the operator returned either an imaginary number or not a real value (nan). Operators paired as the second layer with square root and logarithmic operations frequently become broken (indicated in yellow), indicating a systemic issue with these combinations due to negative numbers.

Interestingly, exponentiation emerges as particularly vulnerable among paired operators, remaining functional only when paired with itself and placing lowest in a comparative bar graph for both data ranges. This breakdown is attributed to the exponential growth of numbers exceeding manageable limits. In contrast, cosine maintains stability as it adjusts in sharpness when paired with other operators, avoiding breaks. Sin behaves similarly but with slightly less stability due to occasional breaks when interacting with square root and logarithmic functions.

Tan, along with the squared operator, appear less stable overall, consistently decreasing in sharpness across most paired operators. Arctan and tanh show relative stability, breaking only when paired with square root and logarithm functions but otherwise maintaining consistent sharpness levels across different operators. This analysis reveals complex dynamics in how mathematical operations interact and perform under varying computational conditions.

**Conclusion**

Recap SAM, brief overview of methodology along with the trends for all sections of graphs (ranges trends, discussion summary -which operators work best for SAM and which ones don’t), closing.

**References**

[StackGP/StackGP.py at main · hoolagans/StackGP · GitHub](https://github.com/hoolagans/StackGP/blob/main/StackGP.py)

Bakurov, I. Haut, N. Banzhaf, W. (2024). Sharpness-Aware Minimization in Genetic Programming. *ArXiv.org*. https://doi.org/10.48550/arXiv.2405.10267