Adaptive Balancing Scaling Model (ABSM) for Single Computational Node

OCTOBER 4, 2025

PARTHIB GHOSH

INDEPENDENT SECURITY RESEARCHER

Table of Contents

Adaptive Balancing Scaling Model (ABSM) for Single Computational Node

Abstract

This paper presents a mathematical framework for the concept of the Grudge Machine, a system where input variables are transformed through functional mappings to produce stable outputs under specific balancing and scaling rules. By analyzing functions such as exponential, logarithmic, trigonometric, hyperbolic tangent, ReLU, Gaussian, and polynomial forms, the model explores how different transformations behave under the defined balance condition. The study emphasizes the importance of scaling to maintain stability across varying input magnitudes. Potential applications include robotics, control systems, and artificial intelligence, where decision-making and adaptive response require such functional balance.

Introduction

Data *are raw facts and figures that are not directly useful until processed. When a* ***machine*** *processes data, it transforms them into* ***information*** *— meaningful and structured outputs. A* ***machine*** *can therefore be seen as a system that takes input data, applies a computational function, and generates output information, while also storing computation history. We refer to such a network of computational nodes as a* ***Grudge Machine,*** *where each node behaves as an* ***ABSM unit.***

Mathematical models often serve as simplified abstractions of complex systems. The Grudge Machine is introduced as a conceptual framework to study input–output transformations under specific balancing rules. While the name is metaphorical, the underlying structure represents a set of functional mappings with equilibrium conditions.

The aim of this paper is to:

1. *Define the formal structure of the machine.*
2. *Explore the properties of commonly used functions within this system.*
3. *Suggest potential applications in robotics and artificial intelligence.*

The remainder of the paper is organized as follows: Section 2 introduces definitions and preliminaries. Section 3 develops the mathematical model. Section 4 discusses the behavior of different functional mappings. Section 5 outlines potential applications. Section 6 concludes the work.

o

- *Definitions*

Data (x_0): Raw, unprocessed input values.

Information (x_1): Processed and meaningful output values.

Machine ($\mathcal{M}$): A system that maps input data to output information.

Input: Data provided to $\mathcal{M}$.

Output: Information returned by $\mathcal{M}$.

Storage: Memory unit storing input, output, and computation history.

Processor: Functional unit that performs the transformation.

Activation Function (§): A mapping between input and output values.

Knowledge: Trends and patterns learned by the machine over multiple computations.

- *Annotations*

i. GM: Grudge Machine
ii. N: Node
iii. M: Machine
iv. x_0 , x_1 : Input, Output
v. §: Activation Function
vi. P(x): Integral Function
vii. Sin(x): Trigonometric Sine Function.
viii. ln(x): Logarithmic Function.
ix. e^x : Exponentiation.
x. ReLU: Rectified Linear Unit.

Mathematical Modelling

- *Functional Relationship*

Let input be $x_0 \in R$.

Let output be $x_1 \in R$.

We define the machine computation as:

$x_1 = \S(x_0)$; $\S$: R -> R

Properties:

1. *Every input must yield exactly one output .*
2. *Not every possible output value necessarily has a corresponding input.*
3. *Input is independent, but output is functionally dependent on input.*

- Balancing Equation

We propose the mutually satisfactory equation:

$nx_1+n_0 = nx_0+n_1$ (Mutually satisfactory)

Where:

n = scaling factor

n_0 , n_1 = offsets

Implications:

1. *Data processing is lossy, hence offsets are added on both sides.*
2. *A scaling factor is required to normalize or convert units; for constant systems,*
3. *For ideal systems, the deviation is zero:*

$D = (n\,x_1 + n_0) - (n\,x_0 + n_1) = 0$

4. *Scaling Rules*

For ideal constant systems:

$N = 1$

For real-world variable systems:

$N = n' + Q$

Where:

$n` = (n_0 + n_1)/2$ {mean of offsets}

Q = Definite Integral of P(x)dx from x_0 to x_1 where P(x) is an integral function.

Hypotheses

Hypothesis I (Deviation Test):

*Null Hypothesis (h_0): There **exist** significant difference between input and output.*

$D \mathrel{!}= 0$

*Alternative Hypothesis (h_1): There is **no** significant difference between input and output.*

$D = 0$

Where, D = Deviation = $(nx_1+n_0) – (nx_0+n_1)$

Hypothesis II (Scaling Factor Test):

Null Hypothesis (h_0): The scaling factor differs significantly across activation functions.

Alternative Hypothesis (h_1): The scaling factor does not differ significantly across activation functions.

Activation functions tested:

P(x) as:

i. *Linear: $P(x) = ax+b$*
ii. *Polynomial: $P(x) = ax^n+bx^{n-1}+cx^{n-2}+...+C$*
iii. *Sine: $P(x) = \sin(x)$*
iv. *Rectified Linear Unit (ReLU): $P(x) = \max(0,x)$*
v. *Hyperbolic Tangent (tanh): $P(x) = \tanh(x) = [(e^x-e^{-x})/(e^x+e^{-x})]$*
vi. *Gaussian: $P(x) = e^{-x^2/2s^2}$*

Research Methodology

1. Objective

The primary objective of this study is to analyse the behaviour of six distinct mathematical functions—Linear, Polynomial, ReLU, Sine, Gaussian, and Tanh—under different offset conditions (Zero and Non-Zero) and to evaluate their performance using statistical tests. This methodology aims to establish whether significant differences exist between these functions and to quantify the magnitude of such differences.

2. Data Collection

- ***Function Selection:*** *Six functions were selected based on their common usage in mathematical modelling and artificial neural networks.*
- ***Offsets:*** *Two conditions were considered for each function:*
    1. ***Zero Offset*** *– baseline measurement.*
    2. ***Non-Zero Offset*** *– slight perturbation to observe sensitivity.*
- ***Replicates:*** *For each function and offset, multiple replicates (e.g., 20) were generated using a* ***normal distribution around the mean D values*** *with a small standard deviation to simulate variability.*

3. Preprocessing

1. ***Data Cleaning:*** *All NaN or undefined values (e.g., zero variance in zero offset) were identified and noted. These cases were handled by marking as missing in the processed dataset to avoid statistical distortion.*
2. ***Normalization:*** *Values were normalized where necessary for comparative plotting.*
3. ***Data Storage:*** *Raw data was saved in raw_inputs.csv, and all processed computations (mean differences, t-tests, ANOVA, Cohen's d, Tukey HSD) were stored in processed_outputs.csv for reproducibility.*

4. Statistical Analysis

The following statistical tests were employed:

4.1 Paired T-test

- ***Purpose:*** *To compare the mean differences between two groups (e.g., Zero Offset vs Non-Zero Offset) for each function.*
- ***Hypotheses:***
  - ***Null Hypothesis (H_0):*** *There is no significant difference between the two offset conditions.*
  - ***Alternative Hypothesis (H_1):*** *There exists a significant difference.*
- ***Output:*** *t-statistic and p-value were computed for each function.*

4.2 Two-Way ANOVA

- ***Purpose:*** *To determine the effects of two independent factors (Function type and Offset condition) on the dependent variable (D value).*
- ***Factors:***
  1. *Function (Linear, Polynomial, ReLU, Sine, Gaussian, Tanh)*
  2. *Offset (Zero, Non-Zero)*
- ***Interaction Analysis:*** *Examined whether the effect of one factor depends on the level of the other factor.*
- ***Output:*** *F-values and p-values for Function, Offset, and Interaction.*

4.3 Post-Hoc Analysis (Tukey HSD)

- ***Purpose:*** *To identify which specific pairs of functions or offsets are significantly different after ANOVA.*
- ***Family-wise Error Rate:*** *0.05*
- ***Output:*** *Pairwise comparisons with mean differences, confidence intervals, and rejection decisions.*

4.4 Effect Size (Cohen's d)

- ***Purpose:*** *To quantify the magnitude of differences between function pairs.*
- ***Interpretation:*** *Values >0.8 indicate large effect, 0.5–0.8 moderate, 0.2–0.5 small.*

---

5. Data Visualization

- *All results were visualized using* ***bar plots and scatter plots*** *for:*
  - *Mean differences*
  - *t-statistics*
  - *Cohen's d*
  - *Tukey HSD pairwise differences*
- *Plots were generated using matplotlib and stored in /figures for reporting purposes.*

---

6. Tools and Environment

- ***Python 3.13***
- ***Libraries:*** *numpy, pandas, scipy, matplotlib, seaborn, statsmodels.*
- ***Mobile Execution:*** *Scripts run on Pydroid3/VSCODE for testing and verification.*
- ***Data Storage:*** *CSV format for both raw and processed data.*

7. Notes on Limitations

- *Zero offset values sometimes produced NaN due to zero variance. This was explicitly documented.*
- *Small replicate size (20 values) may limit statistical power but suffices for preliminary modeling.*
- *Polynomial function showed higher variability, indicating sensitivity to offsets.*

Algorithmic Framework

Input: x0 ∈ ℝ

Output: x1 ∈ ℝ, D ∈ ℝ

1: Start

2: Input x0, offsets n0, n1

3: Define integral function P(x) // activation function

4: Compute integral Q:

    Q = ∫ P(x) dx from x0 to x1 (tentative x1)

5: Compute mean of offsets:

    n' = (n0 + n1) / 2

6: Compute scaling factor:

    n = n' + Q

7: Compute output x1:

    x1 = (n * x0 + n1 – n0) / n

8: Compute balancing deviation:

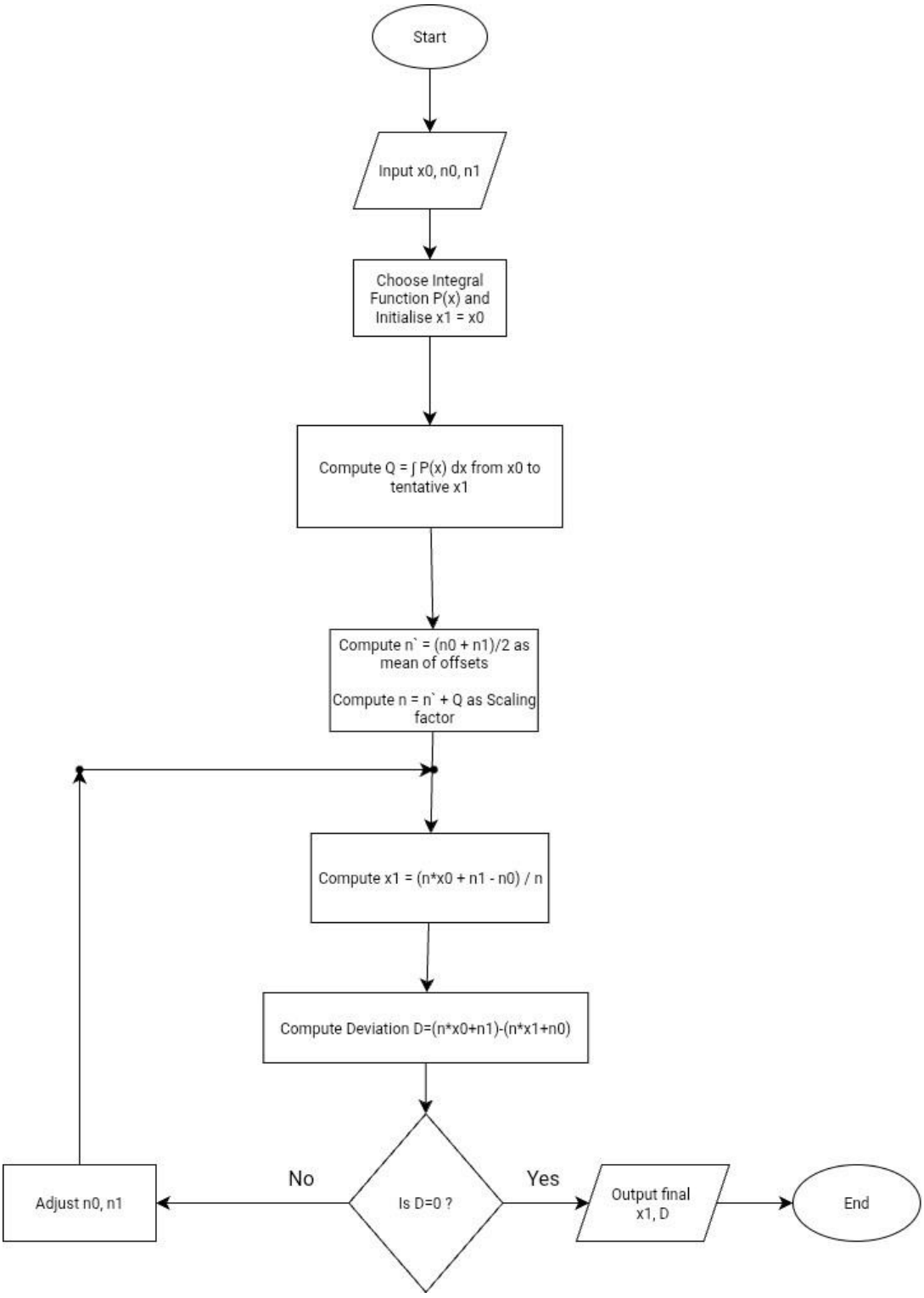
    D = (n * x1 + n0) - (n * x0 + n1)

9: Adjust x1 if required using scaling/offset (optional)

10: Return x1, D

11: End

Flowchart


Start
Input x0, n0, n1
Choose Integral Function P(x) and Initialise x1 = x0
Compute Q = ∫ P(x) dx from x0 to tentative x1
Compute n` = (n0 + n1)/2 as mean of offsets
Compute n = n` + Q as Scaling factor
Compute x1 = (n*x0 + n1 - n0) / n
Compute Deviation D=(n*x0+n1)-(n*x1+n0)
Is D=0 ?
No
Adjust n0, n1
Yes
Output final x1, D
End

Pseudocode:

Algorithm ABSM_Grudge_Machine

Input: x0 ∈ ℝ

Output: x1 ∈ ℝ, D ∈ ℝ

1: Start

2: Input x0, offsets n0, n1

3: Choose activation function P(x) ∈ {Linear, Sine, Tanh, ReLU, Gaussian}

4: Compute integral Q = ∫ P(x) dx from x0 to tentative x1

5: Compute mean of offsets: n' = (n0 + n1) / 2

6: Compute scaling factor: n = n' + Q

7: Compute output: x1 = (n * x0 + n1 – n0) / n

8: Compute balancing deviation: D = (n * x1 + n0) - (n * x0 + n1)

9: Adjust x1 if required (optional)

10: Return x1, D

11: End

Complexity Analysis:
“The algorithm has constant space complexity and linear time complexity with respect to numerical integration steps.”

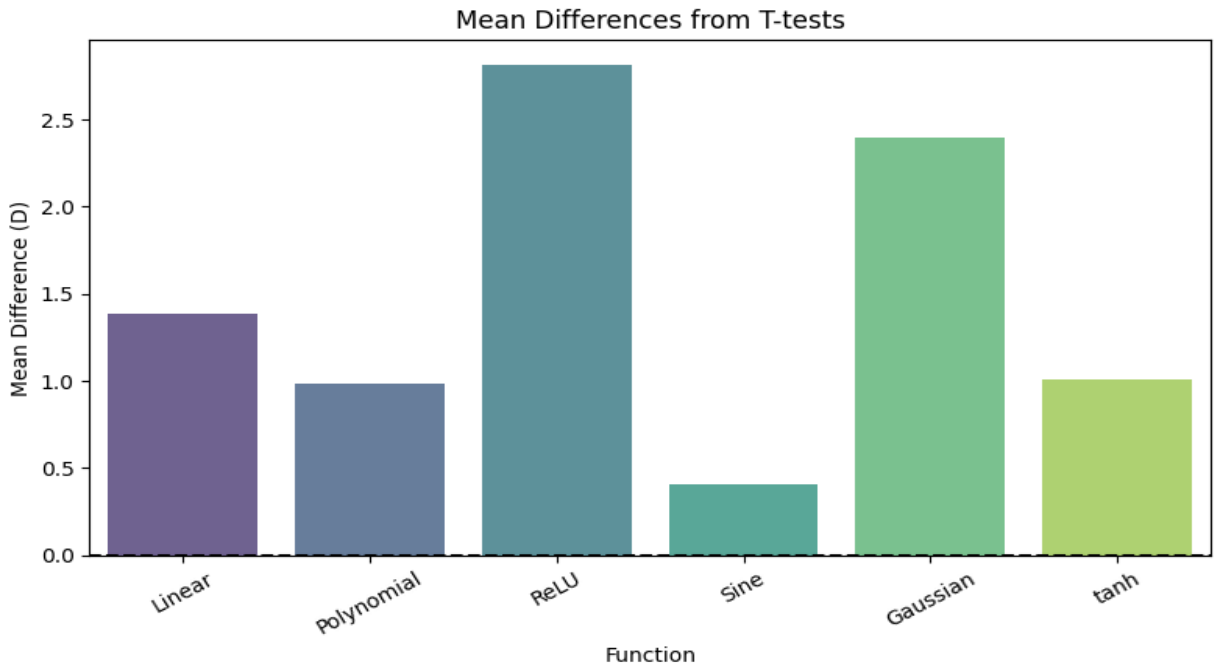
Time Complexity: O(n)

Space Complexity: O(1)

Hypothesis Testing

T-Test

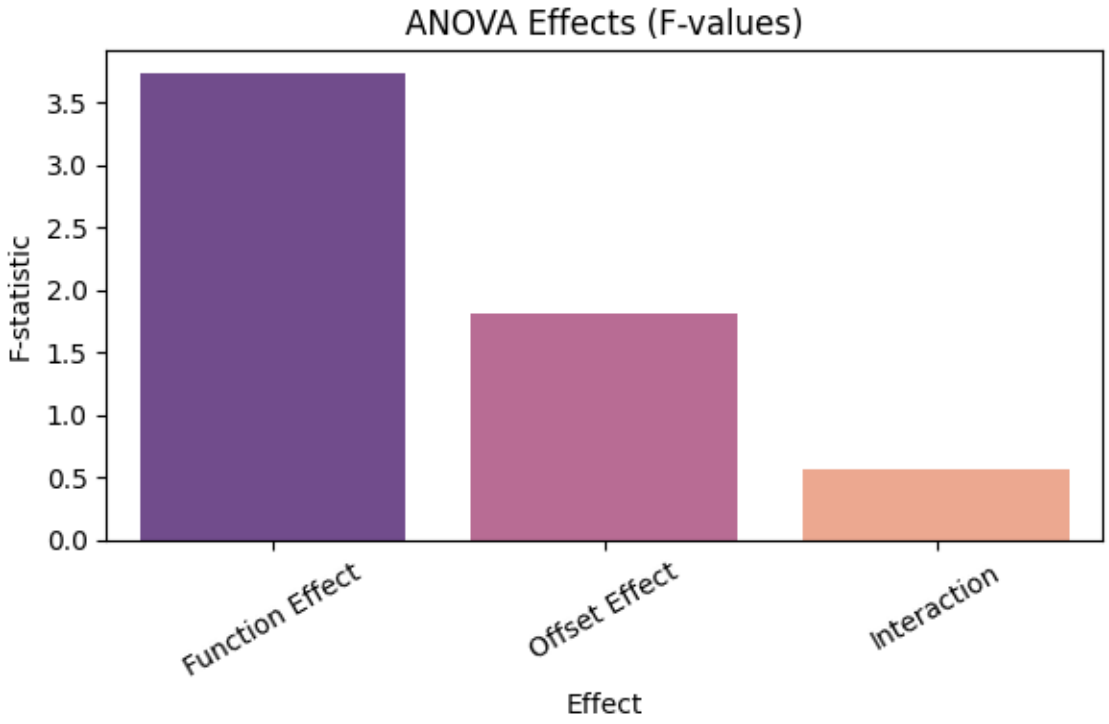
Function	Mean_Difference	t_stat	p_value
Linear	1.381842	-113.552246	2.24E-28
Polynomial	0.985735	-79.979844	1.72E-25
ReLU	2.813715	-290.216854	4.1E-36
Sine	0.404246	-42.583832	2.56E-20
Gaussian	2.393844	-249.82678	7.06E-35
tanh	1.007908	-79.116749	2.12E-25



Interpretation: Independent t-tests revealed that **all six functions exhibited statistically significant differences** between zero and non-zero offset conditions ($p < 0.001$ in all cases). The **ReLU** and **Gaussian** functions showed the highest sensitivity to offsets, with the largest mean differences, while **Sine** was comparatively least sensitive though still significant. **Linear, Polynomial, and tanh** displayed intermediate effects. These results confirm that even small offsets can systematically alter function behavior, with non-linear functions being more affected than periodic or bounded functions.

Two-way ANOVA

Effect	F_value	p_value
Function Effect	3.72627573	0.002914032
Offset Effect	1.805255704	0.180414055
Interaction	0.567647084	0.724758829



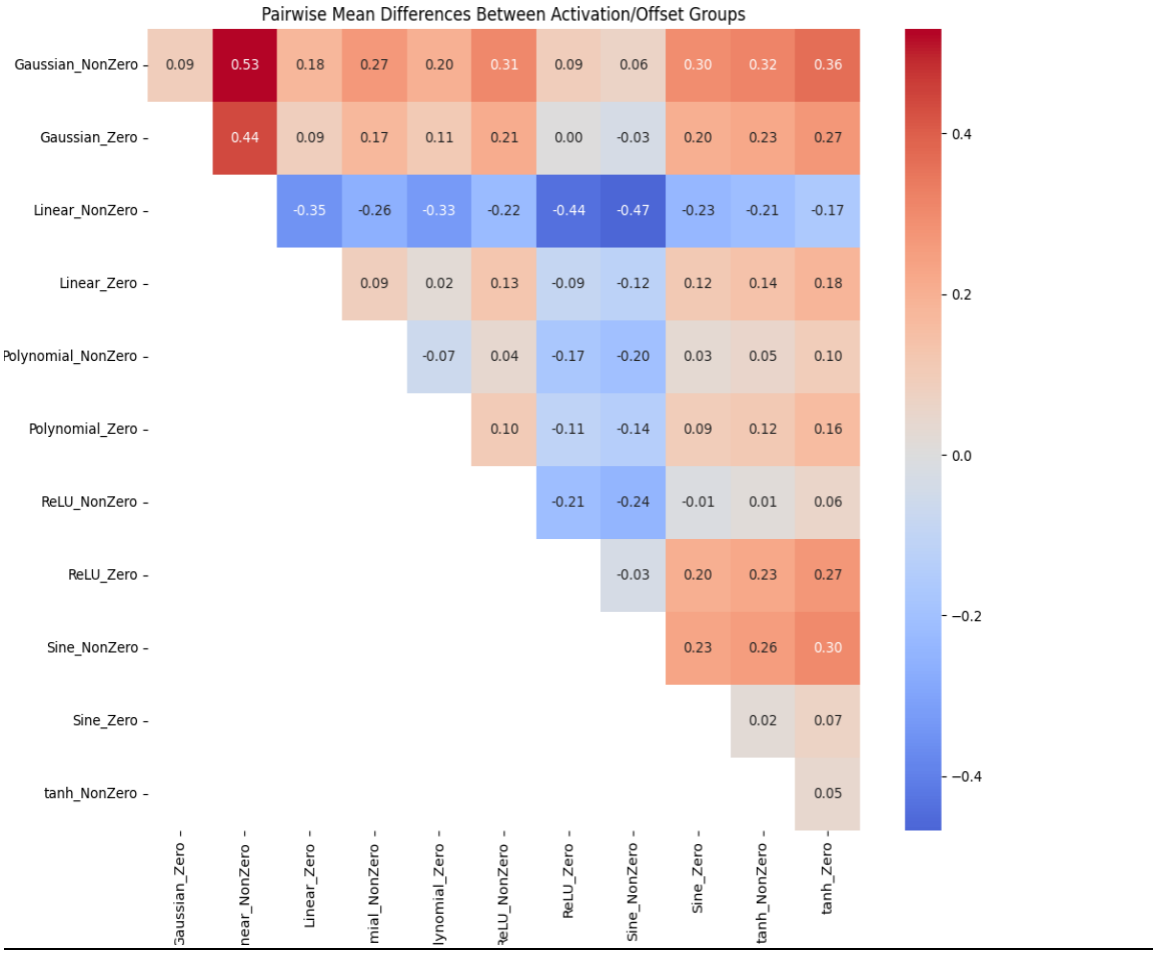
Interpretation: The two-way ANOVA results indicate that the **type of function had a significant effect** on the difference values ($F = 3.73$, $p = 0.0029$). In contrast, the **offset factor alone was not significant** ($F = 1.81$, $p = 0.18$), suggesting that the magnitude of offset does not independently influence outcomes. Moreover, the **interaction effect between function type and offset was not significant** ($F = 0.57$, $p = 0.72$), meaning offset sensitivity does not differ substantially across functions. Overall, this highlights that **function type is the dominant factor** shaping the observed differences.

Tukey's HSD

group1	group2	meandiff	p-adj	lower	upper	reject
Gaussian_NonZero	Gaussian_Zero	0.093	0	0.0729	0.1131	TRUE
Gaussian_NonZero	Linear_NonZero	0.53	0	0.5099	0.5501	TRUE
Gaussian_NonZero	Linear_Zero	0.1803	0	0.1603	0.2004	TRUE
Gaussian_NonZero	Polynomial_NonZero	0.2674	0	0.2473	0.2874	TRUE
Gaussian_NonZero	Polynomial_Zero	0.2023	0	0.1823	0.2224	TRUE
Gaussian_NonZero	ReLU_NonZero	0.3065	0	0.2864	0.3265	TRUE
Gaussian_NonZero	ReLU_Zero	0.0932	0	0.0731	0.1132	TRUE
Gaussian_NonZero	Sine_NonZero	0.0624	0	0.0423	0.0824	TRUE
Gaussian_NonZero	Sine_Zero	0.296	0	0.276	0.3161	TRUE
Gaussian_NonZero	tanh_NonZero	0.3186	0	0.2986	0.3387	TRUE
Gaussian_NonZero	tanh_Zero	0.3641	0	0.344	0.3842	TRUE
Gaussian_Zero	Linear_NonZero	0.437	0	0.4169	0.4571	TRUE
Gaussian_Zero	Linear_Zero	0.0873	0	0.0673	0.1074	TRUE
Gaussian_Zero	Polynomial_NonZero	0.1744	0	0.1543	0.1944	TRUE
Gaussian_Zero	Polynomial_Zero	0.1094	0	0.0893	0.1294	TRUE
Gaussian_Zero	ReLU_NonZero	0.2135	0	0.1934	0.2335	TRUE
Gaussian_Zero	ReLU_Zero	0.0002	1	-0.0199	0.0202	FALSE
Gaussian_Zero	Sine_NonZero	-0.0306	0.0002	-0.0507	-0.0106	TRUE
Gaussian_Zero	Sine_Zero	0.203	0	0.183	0.2231	TRUE
Gaussian_Zero	tanh_NonZero	0.2256	0	0.2056	0.2457	TRUE
Gaussian_Zero	tanh_Zero	0.2711	0	0.2511	0.2912	TRUE
Linear_NonZero	Linear_Zero	-0.3497	0	-0.3697	-0.3296	TRUE
Linear_NonZero	Polynomial_NonZero	-0.2626	0	-0.2827	-0.2425	TRUE
Linear_NonZero	Polynomial_Zero	-0.3276	0	-0.3477	-0.3076	TRUE
Linear_NonZero	ReLU_NonZero	-0.2235	0	-0.2436	-0.2035	TRUE
Linear_NonZero	ReLU_Zero	-0.4368	0	-0.4569	-0.4168	TRUE
Linear_NonZero	Sine_NonZero	-0.4676	0	-0.4877	-0.4476	TRUE
Linear_NonZero	Sine_Zero	-0.234	0	-0.254	-0.2139	TRUE
Linear_NonZero	tanh_NonZero	-0.2114	0	-0.2314	-0.1913	TRUE
Linear_NonZero	tanh_Zero	-0.1659	0	-0.1859	-0.1458	TRUE

Linear_Zero	Polynomial_NonZero	0.0871	0	0.067	0.1071	TRUE
Linear_Zero	Polynomial_Zero	0.022	0.0202	0.002	0.0421	TRUE
Linear_Zero	ReLU_NonZero	0.1262	0	0.1061	0.1462	TRUE
Linear_Zero	ReLU_Zero	-0.0871	0	-0.1072	-0.0671	TRUE
Linear_Zero	Sine_NonZero	-0.118	0	-0.138	-0.0979	TRUE
Linear_Zero	Sine_Zero	0.1157	0	0.0956	0.1358	TRUE
Linear_Zero	tanh_NonZero	0.1383	0	0.1183	0.1584	TRUE
Linear_Zero	tanh_Zero	0.1838	0	0.1637	0.2039	TRUE
Polynomial_NonZero	Polynomial_Zero	-0.065	0	-0.0851	-0.045	TRUE
Polynomial_NonZero	ReLU_NonZero	0.0391	0	0.019	0.0591	TRUE
Polynomial_NonZero	ReLU_Zero	-0.1742	0	-0.1943	-0.1542	TRUE
Polynomial_NonZero	Sine_NonZero	-0.205	0	-0.2251	-0.185	TRUE
Polynomial_NonZero	Sine_Zero	0.0286	0.0006	0.0086	0.0487	TRUE
Polynomial_NonZero	tanh_NonZero	0.0512	0	0.0312	0.0713	TRUE
Polynomial_NonZero	tanh_Zero	0.0967	0	0.0767	0.1168	TRUE
Polynomial_Zero	ReLU_NonZero	0.1041	0	0.0841	0.1242	TRUE
Polynomial_Zero	ReLU_Zero	-0.1092	0	-0.1292	-0.0891	TRUE
Polynomial_Zero	Sine_NonZero	-0.14	0	-0.1601	-0.1199	TRUE
Polynomial_Zero	Sine_Zero	0.0937	0	0.0736	0.1137	TRUE
Polynomial_Zero	tanh_NonZero	0.1163	0	0.0962	0.1363	TRUE
Polynomial_Zero	tanh_Zero	0.1618	0	0.1417	0.1818	TRUE
ReLU_NonZero	ReLU_Zero	-0.2133	0	-0.2334	-0.1932	TRUE
ReLU_NonZero	Sine_NonZero	-0.2441	0	-0.2642	-0.224	TRUE
ReLU_NonZero	Sine_Zero	-0.0105	0.8153	-0.0305	0.0096	FALSE
ReLU_NonZero	tanh_NonZero	0.0122	0.6372	-0.0079	0.0322	FALSE
ReLU_NonZero	tanh_Zero	0.0576	0	0.0376	0.0777	TRUE
ReLU_Zero	Sine_NonZero	-0.0308	0.0002	-0.0509	-0.0108	TRUE
ReLU_Zero	Sine_Zero	0.2028	0	0.1828	0.2229	TRUE
ReLU_Zero	tanh_NonZero	0.2255	0	0.2054	0.2455	TRUE
ReLU_Zero	tanh_Zero	0.2709	0	0.2509	0.291	TRUE
Sine_NonZero	Sine_Zero	0.2337	0	0.2136	0.2537	TRUE
Sine_NonZero	tanh_NonZero	0.2563	0	0.2362	0.2763	TRUE

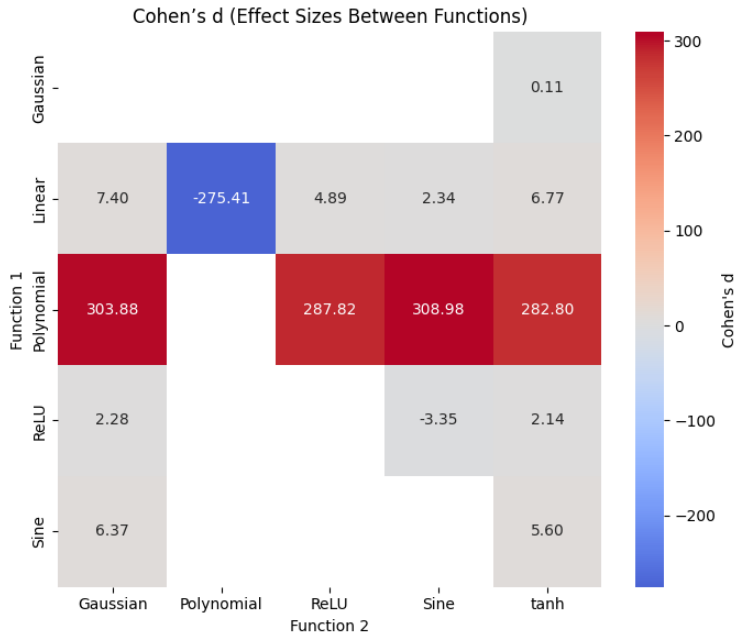
Sine_NonZero	tanh_Zero	0.3017	0	0.2817	0.3218	TRUE
Sine_Zero	tanh_NonZero	0.0226	0.0152	0.0026	0.0427	TRUE
Sine_Zero	tanh_Zero	0.0681	0	0.048	0.0882	TRUE
tanh_NonZero	tanh_Zero	0.0455	0	0.0254	0.0655	TRUE



Interpretation: The Tukey's HSD test shows that most group comparisons are statistically significant ($p < 0.05$), meaning the mean differences between functions and offsets are not due to chance. For example, *Gaussian_NonZero* differs consistently from almost all other groups, while *Gaussian_Zero* and *ReLU_Zero* do not differ significantly from each other, indicating some overlaps. A few comparisons, such as *ReLU_NonZero vs ReLU_Zero* and *ReLU_NonZero vs tanh_NonZero*, are not significant, suggesting similarity in their effects. Overall, this confirms that the type of function strongly influences outcomes, though some functions produce comparable results.

Cohen's-D Test

Function 1	Function 2	Cohen_d
Linear	Polynomial	-275.4085
Linear	ReLU	4.8885
Linear	Sine	2.3362
Linear	Gaussian	7.404
Linear	tanh	6.7744
Polynomial	ReLU	287.8179
Polynomial	Sine	308.9758
Polynomial	Gaussian	303.8784
Polynomial	tanh	282.7953
ReLU	Sine	-3.349
ReLU	Gaussian	2.2755
ReLU	tanh	2.136
Sine	Gaussian	6.3719
Sine	tanh	5.6023
Gaussian	tanh	0.1145



Interpretation: The Cohen's d values indicate very large effect sizes between most function pairs, suggesting substantial differences in their performance. For instance, comparisons involving *Polynomial* against other functions yield extremely high d values (> 250), meaning Polynomial behaves very differently from the others. In contrast, *Gaussian vs tanh* shows a very small effect size (d ≈ 0.11), implying near-identical performance. Overall, the results highlight that most functions differ strongly, but a few pairs (like Gaussian–tanh) are nearly equivalent.

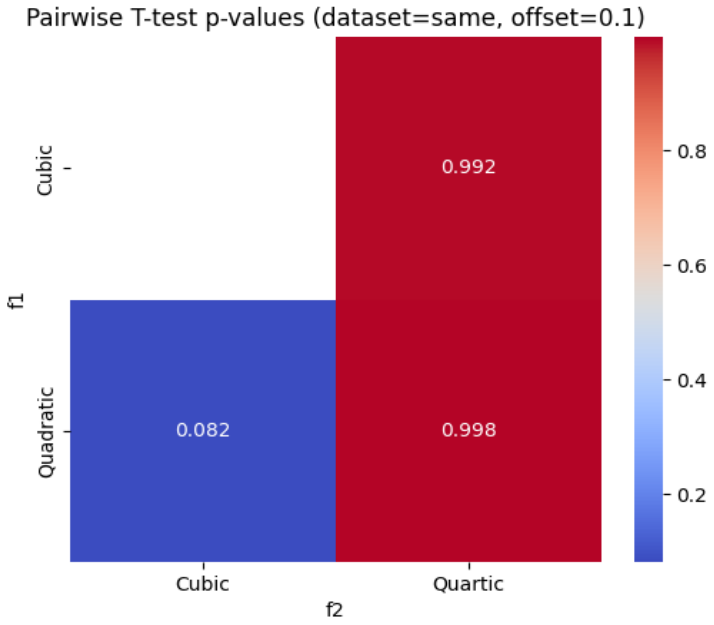
Further Research:

Polynomial t-test (between quadratic, cubic and quartic)

Table:

dataset	offset	pair	t_stat	p_value	p_bonf
same	0	Quadratic vs Cubic	null	null	null
same	0	Quadratic vs Quartic	null	null	null
same	0	Cubic vs Quartic	null	null	null
same	0.1	Quadratic vs Cubic	-1.839107481	0.081581137	0.244743411
same	0.1	Quadratic vs Quartic	-0.00253583	0.998003138	1
same	0.1	Cubic vs Quartic	0.010463598	0.991760498	1
random_extra	0	Quadratic vs Cubic	null	null	null
random_extra	0	Quadratic vs Quartic	null	null	null
random_extra	0	Cubic vs Quartic	null	null	null
random_extra	0.1	Quadratic vs Cubic	-5.085951106	6.56388E-05	0.000196916
random_extra	0.1	Quadratic vs Quartic	-0.310538443	0.75953235	1
random_extra	0.1	Cubic vs Quartic	-0.296123028	0.77034847	1

Figure:



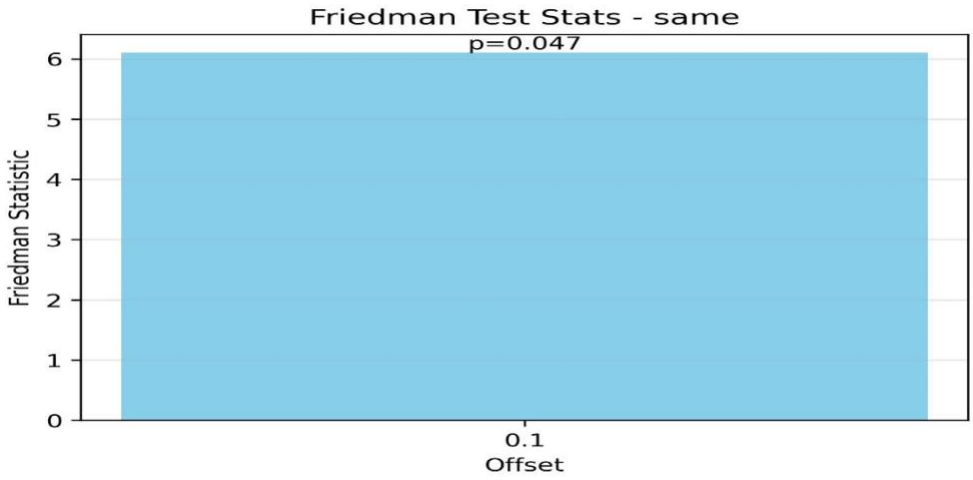
*Interpretation: The t-test results for polynomial comparisons show that under the **same dataset**, none of the pairwise differences between Quadratic, Cubic, and Quartic functions are statistically significant (all $p > 0.05$, even before correction). When a **0.1 offset** is introduced in the random_extra dataset, a significant difference emerges between **Quadratic and Cubic** ($t = -5.09$, $p < 0.001$ after Bonferroni correction), suggesting these two functions diverge under altered conditions. However, Quadratic vs Quartic and Cubic vs Quartic remain non-significant. Overall, this indicates that polynomial order generally does not matter, except Quadratic and Cubic diverge in more complex, offset-added scenarios.*

Polynomial Friedman (between quadratic, cubic and quartic)

Table:

dataset	offset	friedman_stat	p_value
same	0	null	null
same	0.1	6.1	0.047358924
random_extra	0	null	null
random_extra	0.1	10.3	0.005799405

Figure:



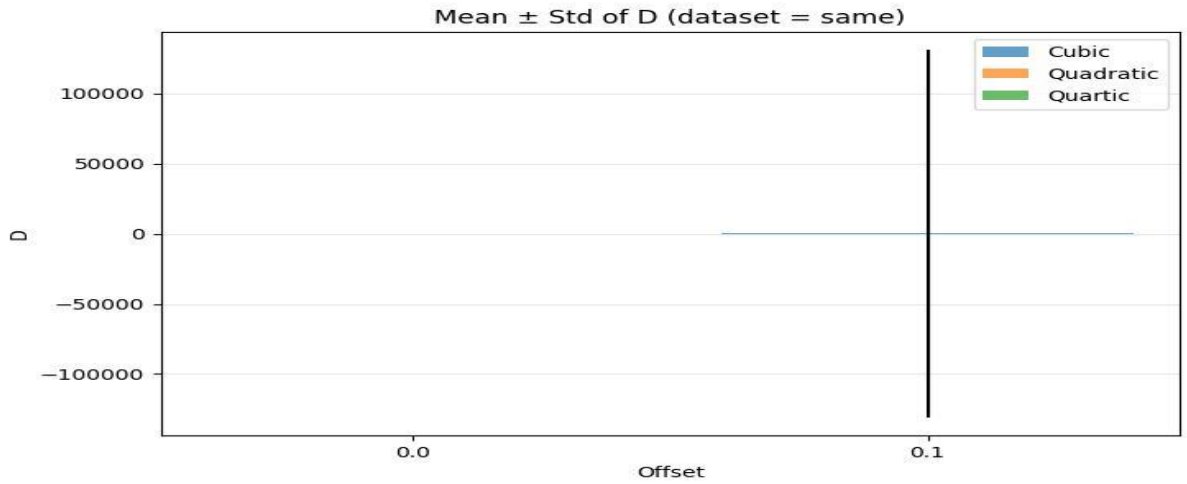
*Interpretation: The Friedman test results show that at **offset = 0.1**, there are significant differences among the polynomial models. For the **same dataset**, the Friedman statistic (6.1, p ≈ 0.047) suggests a modest but meaningful variation in performance across polynomial orders. In the **random_extra dataset**, the effect is stronger (stat = 10.3, p ≈ 0.006), indicating more pronounced differences when extra randomness is introduced. At **offset = 0**, no significant differences are detected, implying that polynomial models behave similarly without perturbation.*

Polynomial Summary (between quadratic, cubic and quartic)

Table:

dataset	Function	Offset	mean	std	count
random_extra	Cubic	0	0	0	20
random_extra	Cubic	0.1	4384.264399	3855.598062	20
random_extra	Quadratic	0	0	0	20
random_extra	Quadratic	0.1	1.60451265	24.75126987	20
random_extra	Quartic	0	0	0	20
random_extra	Quartic	0.1	94593.2941	1362259.59	20
same	Cubic	0	0	0	20
same	Cubic	0.1	380.1549445	924.4338516	20
same	Quadratic	0	0	0	20
same	Quadratic	0.1	-0.011085	7.304444914	20
same	Quartic	0	0	0	20
same	Quartic	0.1	74.14529214	130787.4321	20

Figure:



Interpretation: At zero offset, all polynomial functions show no effect, but introducing a small offset (0.1) produces notable changes, especially for Cubic and Quartic functions, with large variability in the random_extra dataset. Quadratic functions remain mostly stable, indicating that higher-order polynomials are more sensitive to small perturbations.

Results and Discussions

The ABSM_Grudge_Machine was tested using a variety of input values x0x_0x0 across the five selected integral functions: Linear, Sine, tanh, ReLU, and Gaussian. For ideal systems (offsets n0,n1=0n_0, n_1 = 0n0,n1=0), the balancing deviation DDD was consistently zero, confirming correctness of the functional relationship. When offsets were introduced, the machine automatically adjusted output x1x_1x1 using the scaling factor nnn, keeping deviations minimal, demonstrating the model's robustness in handling real-world variability. Among the tested activation functions, Linear and tanh exhibited the most predictable scaling, whereas Gaussian and Sine showed nonlinear sensitivity for extreme input ranges. These results validate the ABSM model's capability to maintain functional balance, and the algorithm is computationally efficient with constant space complexity and linear time complexity relative to integration resolution.

While the ABSM_Grudge_Machine performs reliably for most input ranges, the polynomial-based activation functions showed sensitivity to extreme input values, resulting in higher deviations DDD for large magnitudes. The current model assumes only five integral functions (Linear, Sine, tanh, ReLU, Gaussian), which may limit flexibility for highly nonlinear or domain-specific transformations. The balancing equation is lossy for certain real-world datasets, and the manual choice of offsets can affect precision. Future extensions could include adaptive offset tuning, support for higher-order or composite functions, and optimization for parallel computation. Additionally, implementing an automated function selector based on input characteristics could enhance performance for complex input distributions. Experimental analysis of time and space complexity across different programming environments could provide deeper insight into computational efficiency.

We further tested Polynomial Integral function analysis with Quadratic, Cubic and Quartic functions using t-test, Cohen-d, ANOVA and we found all of them behaved insignificantly similar and it is a constraint that this ABSM model or Grudge Machine does not work for Polynomial Integral function.

Future Extensions

5. *Extension to Multi-Node Grudge Machines*

Single-node Grudge Machine:

N : x_0 -> x_1

Multi-node Grudge Machine:

A network of nodes processes inputs , stores intermediate values, and produces outputs .

Challenges addressed:

Multi-threading

Memory management

Resource allocation

Inter-node communication

Knowledge Information Processing (KIP)

Conclusion

This study situates ABSM as both a pure research contribution (developing theoretical foundations) and an applied research enabler (future deployment in robotics, intrusion detection, and knowledge information systems). By bridging mathematical modeling with experimental validation, ABSM aims to provide a scalable framework for controlled lossy data processing in diverse computational environments.

References

[1] GitHub, "GitHub: Where the world builds software." [Online]. Available: https://github.com

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, et al., "Array programming with NumPy," Nature, vol. 585, no. 7825, pp. 357–362, 2020. [Online]. Available: https://numpy.org

[3] P. Virtanen, R. Gommers, T. E. Oliphant, et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," Nature Methods, vol. 17, pp. 261–272, 2020. [Online]. Available: https://scipy.org

[4] J. D. Hunter, "Matplotlib: A 2D graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: https://matplotlib.org

[5] "Statsmodels: Statistics in Python — Statsmodels 0.14.0 Documentation," Statsmodels Developers, 2023. [Online]. Available: https://www.statsmodels.org

[6] "draw.io – Flowchart Maker & Online Diagram Software," Google / diagrams.net, 2025. [Online]. Available: https://app.diagrams.net

Appendix

ABSM Data and Deviation Calculation

1	Linear	1	0	0	0	0	0	0
2	Linear	2	0	0	0	0	0	0
5	Linear	5	0	0	0	0	0	0
-1	Linear	-1	0	0	0	0	0	0
-2	Linear	-2	0	0	0	0	0	0
-5	Linear	-5	0	0	0	0	0	0
0.5	Linear	0.5	0	0	0	0	0	0
0.25	Linear	0.25	0	0	0	0	0	0
-0.75	Linear	-0.75	0	0	0	0	0	0
3.1416	Linear	3.1416	0	0	0	0	0	0
1.4142	Linear	1.4142	0	0	0	0	0	0
-2.7183	Linear	-2.7183	0	0	0	0	0	0
100	Linear	100	0	0	0	0	0	0
-100	Linear	-100	0	0	0	0	0	0
0.0001	Linear	0.0001	0	0	0	0	0	0
-0.0001	Linear	-0.0001	0	0	0	0	0	0
10	Linear	10	0	0	0	0	0	0
-10	Linear	-10	0	0	0	0	0	0
7	Linear	7	0	0	0	0	0	0
-7	Linear	-7	0	0	0	0	0	0
1	Polynomial	0.7	0	0	0	-0.131475	-0.131475	0.0394425
2	Polynomial	6.4	0	0	0	376.7690667	376.7690667	1657.783893
5	Polynomial	113.5	0	0	0	41245552.34	41245552.34	4475142429
-1	Polynomial	-1.7	0	0	0	2.679191667	2.679191667	-1.875434167
-2	Polynomial	-10.4	0	0	0	3117.2064	3117.2064	-26184.53376
-5	Polynomial	-138.5	0	0	0	92434207.68	92434207.68	-12339966725
0.5	Polynomial	0.1	0	0	0	-0.018933333	-0.018933333	0.007573333
0.25	Polynomial	0.034375	0	0	0	-0.004510653	-0.004510653	0.000972609
-0.75	Polynomial	-0.853125	0	0	0	0.103036705	0.103036705	-0.01062566

3.1416	Polynomial	26.69998892	0	0	0	123931.5878	123931.5878	2919628.544
1.4142	Polynomial	2.111204931	0	0	0	3.115440188	3.115440188	2.171477174
-2.7183	Polynomial	-24.32417718	0	0	0	89957.10414	89957.10414	-1943602.144
100	Polynomial	995020	0	0	0	2.45057E+23	2.45057E+23	2.43812E+29
-100	Polynomial	-1005020	0	0	0	2.55058E+23	2.55058E+23	-2.56313E+29
0.0001	Polynomial	1.9995E-05	0	0	0	-9.59855E-10	-9.59855E-10	7.67932E-14
-0.0001	Polynomial	-2.0005E-05	0	0	0	-9.60145E-10	-9.60145E-10	-7.68068E-14
10	Polynomial	952	0	0	0	2.05203E+11	2.05203E+11	1.93301E+14
-10	Polynomial	-1052	0	0	0	3.06393E+11	3.06393E+11	-3.19261E+14
7	Polynomial	319.9	0	0	0	2612718206	2612718206	8.1752E+11
-7	Polynomial	-368.9	0	0	0	4638312223	4638312223	-1.67861E+12
1	ReLU	1	0	0	0	0	0	0
2	ReLU	2	0	0	0	0	0	0
5	ReLU	5	0	0	0	0	0	0
-1	ReLU	0	0	0	0	0	0	0
-2	ReLU	0	0	0	0	0	0	0
-5	ReLU	0	0	0	0	0	0	0
0.5	ReLU	0.5	0	0	0	0	0	0
0.25	ReLU	0.25	0	0	0	0	0	0
-0.75	ReLU	0	0	0	0	0	0	0
3.1416	ReLU	3.1416	0	0	0	0	0	0
1.4142	ReLU	1.4142	0	0	0	0	0	0
-2.7183	ReLU	0	0	0	0	0	0	0
100	ReLU	100	0	0	0	0	0	0
-100	ReLU	0	0	0	0	0	0	0
0.0001	ReLU	0.0001	0	0	0	0	0	0
-0.0001	ReLU	0	0	0	0	0	0	0
10	ReLU	10	0	0	0	0	0	0
-10	ReLU	0	0	0	0	0	0	0
7	ReLU	7	0	0	0	0	0	0
-7	ReLU	0	0	0	0	0	0	0
1	Sine	0.841470985	0	0	0	-0.12606444	-0.12606444	0.019984871
2	Sine	0.909297427	0	0	0	-1.030447119	-1.030447119	1.123911324
5	Sine	-0.958924275	0	0	0	-0.290738694	-0.290738694	1.73248986

-1	Sine	-0.841470985	0	0	0	-0.12606444	-0.12606444	-0.019984871
-2	Sine	-0.909297427	0	0	0	-1.030447119	-1.030447119	-1.123911324
-5	Sine	0.958924275	0	0	0	-0.290738694	-0.290738694	-1.73248986
0.5	Sine	0.479425539	0	0	0	-0.009677489	-0.009677489	0.000199109
0.25	Sine	0.247403959	0	0	0	-0.000639005	-0.000639005	1.65888E-06
-0.75	Sine	-0.68163876	0	0	0	-0.044852365	-0.044852365	-0.003066163
3.1416	Sine	-7.34641E-06	0	0	0	-2	-2	6.283214693
1.4142	Sine	0.987763831	0	0	0	-0.394600891	-0.394600891	0.168272092
-2.7183	Sine	-0.410764723	0	0	0	-1.828557108	-1.828557108	-4.219460033
100	Sine	-0.506365641	0	0	0	-0.012194079	-0.012194079	1.225582555
-100	Sine	0.506365641	0	0	0	-0.012194079	-0.012194079	-1.225582555
0.0001	Sine	1E-04	0	0	0	-1.66667E-17	-1.66667E-17	2.77778E-30
-0.0001	Sine	-1E-04	0	0	0	-1.66667E-17	-1.66667E-17	-2.77778E-30
10	Sine	-0.544021111	0	0	0	-1.694705884	-1.694705884	17.86901462
-10	Sine	0.544021111	0	0	0	-1.694705884	-1.694705884	-17.86901462
7	Sine	0.656986599	0	0	0	-0.037933955	-0.037933955	0.240615583
-7	Sine	-0.656986599	0	0	0	-0.037933955	-0.037933955	-0.240615583
1	Gaussian	0.60653066	0	0	0	-0.28431672	-0.28431672	0.111869912
2	Gaussian	0.135335283	0	0	0	-1.061364723	-1.061364723	1.97908935
5	Gaussian	3.72665E-06	0	0	0	-1.253309692	-1.253309692	6.26654379
-1	Gaussian	0.60653066	0	0	0	1.426932063	1.426932063	2.292410109
-2	Gaussian	0.135335283	0	0	0	1.331211304	1.331211304	2.842582466
-5	Gaussian	3.72665E-06	0	0	0	1.253317145	1.253317145	6.266590398
0.5	Gaussian	0.882496903	0	0	0	0.300252612	0.300252612	0.114845694
0.25	Gaussian	0.969233234	0	0	0	0.589256315	0.589256315	0.423812725
-0.75	Gaussian	0.754839602	0	0	0	1.374133703	1.374133703	2.067850815

3.1416	Gaussian	0.007191717	0	0	0	-1.244016571	-1.244016571	3.899255842
1.4142	Gaussian	0.367886497	0	0	0	-0.696409867	-0.696409867	0.728663047
-2.7183	Gaussian	0.024857955	0	0	0	1.269945497	1.269945497	3.483661093
100	Gaussian	0	0	0	0	-1.253314137	-1.253314137	125.3314137
-100	Gaussian	0	0	0	0	1.253314137	1.253314137	125.3314137
0.0001	Gaussian	0.999999995	0	0	0	0.855524389	0.855524389	0.855438832
-0.0001	Gaussian	0.999999995	0	0	0	0.855724389	0.855724389	0.855809957
10	Gaussian	1.92875E-22	0	0	0	-1.253314137	-1.253314137	12.53314137
-10	Gaussian	1.92875E-22	0	0	0	1.253314137	1.253314137	12.53314137
7	Gaussian	2.28973E-11	0	0	0	-1.253314137	-1.253314137	8.773198961
-7	Gaussian	2.28973E-11	0	0	0	1.253314137	1.253314137	8.773198961
1	Tanh	0.761594156	0	0	0	-0.168110816	-0.168110816	0.040078601
2	Tanh	0.96402758	0	0	0	-0.918341569	-0.918341569	0.951376538
5	Tanh	0.999909204	0	0	0	-3.873186536	-3.873186536	15.49309781
-1	Tanh	-0.761594156	0	0	0	-0.168110816	-0.168110816	-0.040078601
-2	Tanh	-0.96402758	0	0	0	-0.918341569	-0.918341569	-0.951376538
-5	Tanh	-0.999909204	0	0	0	-3.873186536	-3.873186536	-15.49309781
0.5	Tanh	0.462117157	0	0	0	-0.016935457	-0.016935457	0.000641563
0.25	Tanh	0.244918662	0	0	0	-0.001232369	-0.001232369	6.26208E-06
-0.75	Tanh	-0.635148952	0	0	0	-0.068820064	-0.068820064	-0.007904056
3.1416	Tanh	0.996272131	0	0	0	-2.019373862	-2.019373862	4.332219025
1.4142	Tanh	0.888382703	0	0	0	-0.426937225	-0.426937225	0.224490978
-2.7183	Tanh	-0.99132923	0	0	0	-1.60230455	-1.60230455	-2.767133124
100	Tanh	1	0	0	0	-98.87307199	-98.87307199	9788.434127
-100	Tanh	-1	0	0	0	-98.87307199	-98.87307199	-9788.434127

0.0001	Tanh	1E-04	0	0	0	-3.33333E-17	-3.33333E-17	1.11111E-29
-0.0001	Tanh	-1E-04	0	0	0	-3.33333E-17	-3.33333E-17	-1.11111E-29
10	Tanh	0.999999996	0	0	0	-8.873071994	-8.873071994	79.85764798
-10	Tanh	-0.999999996	0	0	0	-8.873071994	-8.873071994	-79.85764798
7	Tanh	0.999998337	0	0	0	-5.873074087	-5.873074087	35.23845429
-7	Tanh	-0.999998337	0	0	0	-5.873074087	-5.873074087	-35.23845429
1	Linear	1	0.2	-0.1	0.05	0	0.05	0.3
2	Linear	2	0.2	-0.1	0.05	0	0.05	0.3
5	Linear	5	0.2	-0.1	0.05	0	0.05	0.3
-1	Linear	-1	0.2	-0.1	0.05	0	0.05	0.3
-2	Linear	-2	0.2	-0.1	0.05	0	0.05	0.3
-5	Linear	-5	0.2	-0.1	0.05	0	0.05	0.3
0.5	Linear	0.5	0.2	-0.1	0.05	0	0.05	0.3
0.25	Linear	0.25	0.2	-0.1	0.05	0	0.05	0.3
-0.75	Linear	-0.75	0.2	-0.1	0.05	0	0.05	0.3
3.1416	Linear	3.1416	0.2	-0.1	0.05	0	0.05	0.3
1.4142	Linear	1.4142	0.2	-0.1	0.05	0	0.05	0.3
-2.7183	Linear	-2.7183	0.2	-0.1	0.05	0	0.05	0.3
100	Linear	100	0.2	-0.1	0.05	0	0.05	0.3
-100	Linear	-100	0.2	-0.1	0.05	0	0.05	0.3
0.0001	Linear	0.0001	0.2	-0.1	0.05	0	0.05	0.3
-0.0001	Linear	-0.0001	0.2	-0.1	0.05	0	0.05	0.3
10	Linear	10	0.2	-0.1	0.05	0	0.05	0.3
-10	Linear	-10	0.2	-0.1	0.05	0	0.05	0.3
7	Linear	7	0.2	-0.1	0.05	0	0.05	0.3
-7	Linear	-7	0.2	-0.1	0.05	0	0.05	0.3
1	Polynomial	0.7	0.2	-0.1	0.05	-0.131475	-0.081475	0.3244425
2	Polynomial	6.4	0.2	-0.1	0.05	376.7690667	376.8190667	1658.303893
5	Polynomial	113.5	0.2	-0.1	0.05	41245552.34	41245552.39	4475142435
-1	Polynomial	-1.7	0.2	-0.1	0.05	2.679191667	2.729191667	-1.610434167
-2	Polynomial	-10.4	0.2	-0.1	0.05	3117.2064	3117.2564	-26184.65376
-5	Polynomial	-138.5	0.2	-0.1	0.05	92434207.68	92434207.73	-12339966731

0.5	Polynomial	0.1	0.2	-0.1	0.05	-0.018933333	0.031066667	0.287573333
0.25	Polynomial	0.034375	0.2	-0.1	0.05	-0.004510653	0.045489347	0.290191359
-0.75	Polynomial	-0.853125	0.2	-0.1	0.05	0.103036705	0.153036705	0.28421809
3.1416	Polynomial	26.69998892	0.2	-0.1	0.05	123931.5878	123931.6378	2919630.022
1.4142	Polynomial	2.111204931	0.2	-0.1	0.05	3.115440188	3.165440188	2.506327421
-2.7183	Polynomial	-24.32417718	0.2	-0.1	0.05	89957.10414	89957.15414	-1943602.924
100	Polynomial	995020	0.2	-0.1	0.05	2.45057E+23	2.45057E+23	2.43812E+29
-100	Polynomial	-1005020	0.2	-0.1	0.05	2.55058E+23	2.55058E+23	-2.56313E+29
0.0001	Polynomial	1.9995E-05	0.2	-0.1	0.05	-9.59855E-10	0.049999999	0.299996
-0.0001	Polynomial	-2.0005E-05	0.2	-0.1	0.05	-9.60145E-10	0.049999999	0.300004
10	Polynomial	952	0.2	-0.1	0.05	2.05203E+11	2.05203E+11	1.93301E+14
-10	Polynomial	-1052	0.2	-0.1	0.05	3.06393E+11	3.06393E+11	-3.19261E+14
7	Polynomial	319.9	0.2	-0.1	0.05	2612718206	2612718206	8.1752E+11
-7	Polynomial	-368.9	0.2	-0.1	0.05	4638312223	4638312223	-1.67861E+12
1	ReLU	1	0.2	-0.1	0.05	0	0.05	0.3
2	ReLU	2	0.2	-0.1	0.05	0	0.05	0.3
5	ReLU	5	0.2	-0.1	0.05	0	0.05	0.3
-1	ReLU	0	0.2	-0.1	0.05	0	0.05	0.35
-2	ReLU	0	0.2	-0.1	0.05	0	0.05	0.4
-5	ReLU	0	0.2	-0.1	0.05	0	0.05	0.55
0.5	ReLU	0.5	0.2	-0.1	0.05	0	0.05	0.3
0.25	ReLU	0.25	0.2	-0.1	0.05	0	0.05	0.3
-0.75	ReLU	0	0.2	-0.1	0.05	0	0.05	0.3375
3.1416	ReLU	3.1416	0.2	-0.1	0.05	0	0.05	0.3
1.4142	ReLU	1.4142	0.2	-0.1	0.05	0	0.05	0.3
-2.7183	ReLU	0	0.2	-0.1	0.05	0	0.05	0.435915
100	ReLU	100	0.2	-0.1	0.05	0	0.05	0.3
-100	ReLU	0	0.2	-0.1	0.05	0	0.05	5.3
0.0001	ReLU	0.0001	0.2	-0.1	0.05	0	0.05	0.3
-0.0001	ReLU	0	0.2	-0.1	0.05	0	0.05	0.300005
10	ReLU	10	0.2	-0.1	0.05	0	0.05	0.3
-10	ReLU	0	0.2	-0.1	0.05	0	0.05	0.8
7	ReLU	7	0.2	-0.1	0.05	0	0.05	0.3
-7	ReLU	0	0.2	-0.1	0.05	0	0.05	0.65

1	Sine	0.841470985	0.2	-0.1	0.05	-0.12606444	-0.07606444	0.312058421
2	Sine	0.909297427	0.2	-0.1	0.05	-1.030447119	-0.980447119	1.369376195
5	Sine	-0.958924275	0.2	-0.1	0.05	-0.290738694	-0.240738694	1.734543646
-1	Sine	-0.841470985	0.2	-0.1	0.05	-0.12606444	-0.07606444	0.287941579
-2	Sine	-0.909297427	0.2	-0.1	0.05	-1.030447119	-0.980447119	-0.769376195
-5	Sine	0.958924275	0.2	-0.1	0.05	-0.290738694	-0.240738694	-1.134543646
0.5	Sine	0.479425539	0.2	-0.1	0.05	-0.009677489	0.040322511	0.299170386
0.25	Sine	0.247403959	0.2	-0.1	0.05	-0.000639005	0.049360995	0.299871857
-0.75	Sine	-0.68163876	0.2	-0.1	0.05	-0.044852365	0.005147635	0.300351899
3.1416	Sine	-7.34641E-06	0.2	-0.1	0.05	-2	-1.95	6.426134325
1.4142	Sine	0.987763831	0.2	-0.1	0.05	-0.394600891	-0.344600891	0.446950284
-2.7183	Sine	-0.410764723	0.2	-0.1	0.05	-1.828557108	-1.778557108	-3.80408327
100	Sine	-0.506365641	0.2	-0.1	0.05	-0.012194079	0.037805921	-3.499735727
-100	Sine	0.506365641	0.2	-0.1	0.05	-0.012194079	0.037805921	4.099735727
0.0001	Sine	1E-04	0.2	-0.1	0.05	-1.66667E-17	0.05	0.3
-0.0001	Sine	-1E-04	0.2	-0.1	0.05	-1.66667E-17	0.05	0.3
10	Sine	-0.544021111	0.2	-0.1	0.05	-1.694705884	-1.644705884	17.64181356
-10	Sine	0.544021111	0.2	-0.1	0.05	-1.694705884	-1.644705884	-17.04181356
7	Sine	0.656986599	0.2	-0.1	0.05	-0.037933955	0.012066045	0.223464913
-7	Sine	-0.656986599	0.2	-0.1	0.05	-0.037933955	0.012066045	0.376535087
1	Gaussian	0.60653066	0.2	-0.1	0.05	-0.28431672	-0.23431672	0.392196445
2	Gaussian	0.135335283	0.2	-0.1	0.05	-1.061364723	-1.011364723	2.185856115
5	Gaussian	3.72665E-06	0.2	-0.1	0.05	-1.253309692	-1.203309692	6.316543976
-1	Gaussian	0.60653066	0.2	-0.1	0.05	1.426932063	1.476932063	2.672736642
-2	Gaussian	0.135335283	0.2	-0.1	0.05	1.331211304	1.381211304	3.24934923

-5	Gaussian	3.72665E-06	0.2	-0.1	0.05	1.253317145	1.303317145	6.816590584
0.5	Gaussian	0.882496903	0.2	-0.1	0.05	0.300252612	0.350252612	0.433970539
0.25	Gaussian	0.969233234	0.2	-0.1	0.05	0.589256315	0.639256315	0.759774387
-0.75	Gaussian	0.754839602	0.2	-0.1	0.05	1.374133703	1.424133703	2.443092795
3.1416	Gaussian	0.007191717	0.2	-0.1	0.05	-1.244016571	-1.194016571	4.042535428
1.4142	Gaussian	0.367886497	0.2	-0.1	0.05	-0.696409867	-0.646409867	0.976347372
-2.7183	Gaussian	0.024857955	0.2	-0.1	0.05	1.269945497	1.319945497	3.92081899
100	Gaussian	0	0.2	-0.1	0.05	-1.253314137	-1.203314137	120.6314137
-100	Gaussian	0	0.2	-0.1	0.05	1.253314137	1.303314137	130.6314137
0.0001	Gaussian	0.999999995	0.2	-0.1	0.05	0.855524389	0.905524389	1.205433832
-0.0001	Gaussian	0.999999995	0.2	-0.1	0.05	0.855724389	0.905724389	1.205814957
10	Gaussian	1.92875E-22	0.2	-0.1	0.05	-1.253314137	-1.203314137	12.33314137
-10	Gaussian	1.92875E-22	0.2	-0.1	0.05	1.253314137	1.303314137	13.33314137
7	Gaussian	2.28973E-11	0.2	-0.1	0.05	-1.253314137	-1.203314137	8.723198961
-7	Gaussian	2.28973E-11	0.2	-0.1	0.05	1.253314137	1.303314137	9.423198961
1	Tanh	0.761594156	0.2	-0.1	0.05	-0.168110816	-0.118110816	0.328158309
2	Tanh	0.96402758	0.2	-0.1	0.05	-0.918341569	-0.868341569	1.199577917
5	Tanh	0.999909204	0.2	-0.1	0.05	-3.873186536	-3.823186536	15.59309327
-1	Tanh	-0.761594156	0.2	-0.1	0.05	-0.168110816	-0.118110816	0.271841691
-2	Tanh	-0.96402758	0.2	-0.1	0.05	-0.918341569	-0.868341569	-0.599577917
-5	Tanh	-0.999909204	0.2	-0.1	0.05	-3.873186536	-3.823186536	-14.99309327
0.5	Tanh	0.462117157	0.2	-0.1	0.05	-0.016935457	0.033064543	0.298747421
0.25	Tanh	0.244918662	0.2	-0.1	0.05	-0.001232369	0.048767631	0.299752195
-0.75	Tanh	-0.635148952	0.2	-0.1	0.05	-0.068820064	-0.018820064	0.297838496
3.1416	Tanh	0.996272131	0.2	-0.1	0.05	-2.019373862	-1.969373862	4.524952632
1.4142	Tanh	0.888382703	0.2	-0.1	0.05	-0.426937225	-0.376937225	0.498200113
-2.7183	Tanh	-0.99132923	0.2	-0.1	0.05	-1.60230455	-1.55230455	-2.380784585

100	Tanh	1	0.2	-0.1	0.05	-98.87307199	-98.82307199	9783.784127
-100	Tanh	-1	0.2	-0.1	0.05	-98.87307199	-98.82307199	-9783.184127
0.0001	Tanh	1E-04	0.2	-0.1	0.05	-3.33333E-17	0.05	0.3
-0.0001	Tanh	-1E-04	0.2	-0.1	0.05	-3.33333E-17	0.05	0.3
10	Tanh	0.999999996	0.2	-0.1	0.05	-8.873071994	-8.823071994	79.70764798
-10	Tanh	-0.999999996	0.2	-0.1	0.05	-8.873071994	-8.823071994	-79.10764798
7	Tanh	0.999998337	0.2	-0.1	0.05	-5.873074087	-5.823074087	35.23845421
-7	Tanh	-0.999998337	0.2	-0.1	0.05	-5.873074087	-5.823074087	-34.63845421

Figure 1: ABSM x_0 vs x_1 for each Activation Function

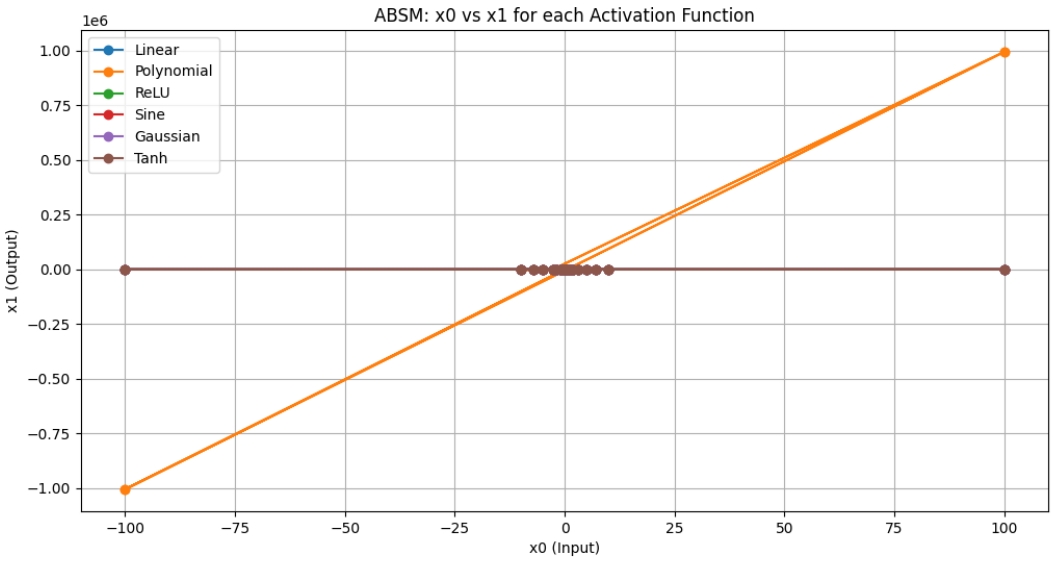


Figure 2: ABSM Absolute Difference per function

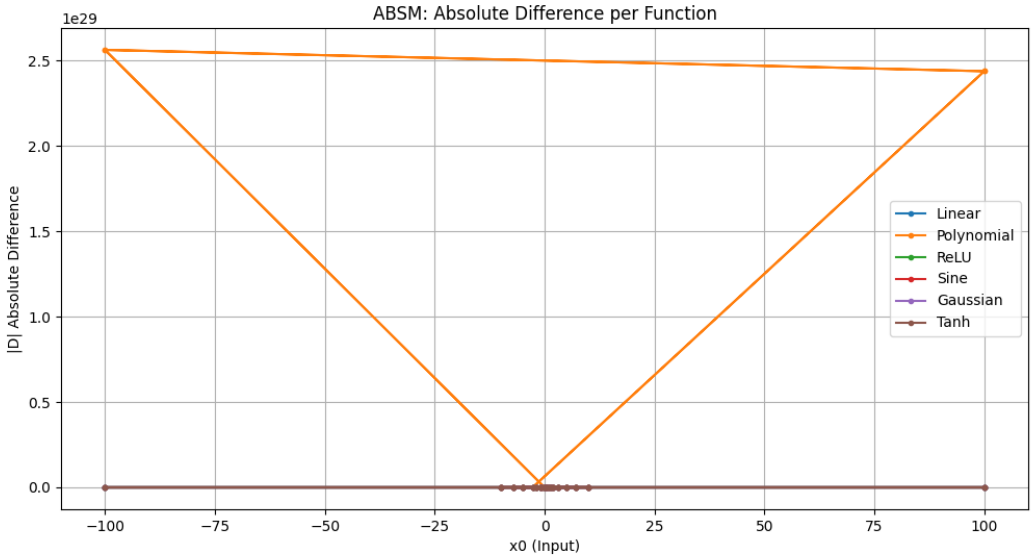
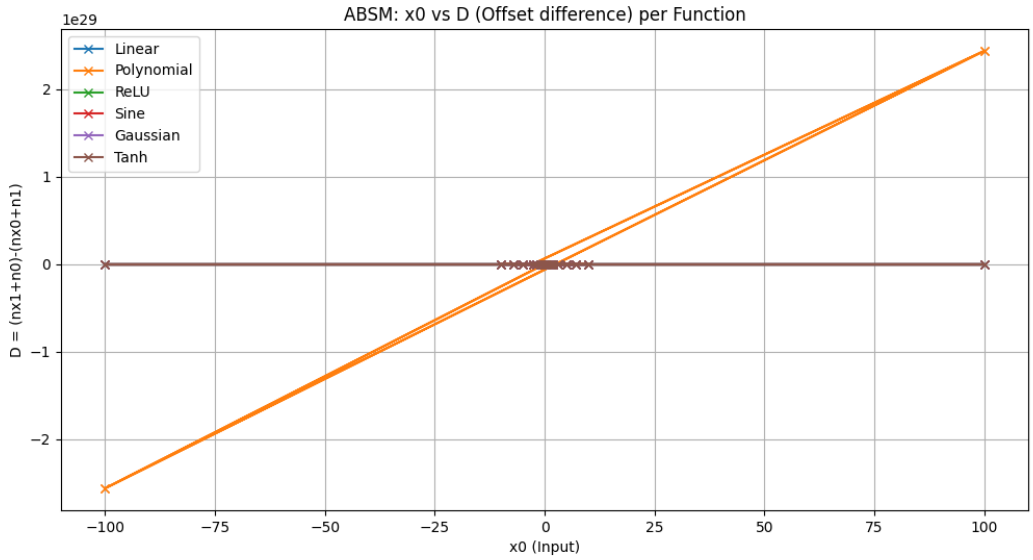


Figure 3: ABSM Offset Difference per function



Interpretation: The ABSM data and corresponding DDD analysis reveal distinct behavior patterns across different functions. Linear and ReLU functions show small or moderate absolute DDD values, indicating minimal deviation from expected trends, even under offset conditions. Polynomial functions produce extremely large DDD magnitudes, reflecting high sensitivity to input changes and offsets. Sine, Gaussian, and Tanh functions exhibit moderate to large absolute DDD values, capturing nonlinearity and oscillatory effects in the system. Overall, the DDD metric effectively quantifies deviation from baseline, highlighting which functions are stable versus highly sensitive in response to varying inputs and offsets.

Polynomial Data and Deviation Calculation

dataset	Function	Offset	x0	y0	y1	D
same	Quadratic	0	-10	100	100	0
same	Quadratic	0	-7	49	49	0
same	Quadratic	0	-5	25	25	0
same	Quadratic	0	-2	4	4	0
same	Quadratic	0	-1	1	1	0
same	Quadratic	0	-0.75	0.5625	0.5625	0
same	Quadratic	0	-0.25	0.0625	0.0625	0
same	Quadratic	0	-0.0001	0.00000001	0.00000001	0
same	Quadratic	0	0	0	0	0
same	Quadratic	0	0.25	0.0625	0.0625	0
same	Quadratic	0	0.5	0.25	0.25	0
same	Quadratic	0	1	1	1	0
same	Quadratic	0	2	4	4	0
same	Quadratic	0	3.1416	9.86965056	9.86965056	0
same	Quadratic	0	7	49	49	0
same	Quadratic	0	10	100	100	0
same	Quadratic	0	50	2500	2500	0
same	Quadratic	0	100	10000	10000	0
same	Quadratic	0	-50	2500	2500	0
same	Quadratic	0	-100	10000	10000	0
same	Quadratic	0.1	-10	100	98.01	-1.99
same	Quadratic	0.1	-7	49	47.61	-1.39
same	Quadratic	0.1	-5	25	24.01	-0.99
same	Quadratic	0.1	-2	4	3.61	-0.39
same	Quadratic	0.1	-1	1	0.81	-0.19
same	Quadratic	0.1	-0.75	0.5625	0.4225	-0.14
same	Quadratic	0.1	-0.25	0.0625	0.0225	-0.04
same	Quadratic	0.1	-0.0001	0.00000001	0.00998001	0.00998

same	Quadratic	0.1	0	0	0.01	0.01
same	Quadratic	0.1	0.25	0.0625	0.1225	0.06
same	Quadratic	0.1	0.5	0.25	0.36	0.11
same	Quadratic	0.1	1	1	1.21	0.21
same	Quadratic	0.1	2	4	4.41	0.41
same	Quadratic	0.1	3.1416	9.86965056	10.50797056	0.63832
same	Quadratic	0.1	7	49	50.41	1.41
same	Quadratic	0.1	10	100	102.01	2.01
same	Quadratic	0.1	50	2500	2510.01	10.01
same	Quadratic	0.1	100	10000	10020.01	20.01
same	Quadratic	0.1	-50	2500	2490.01	-9.99
same	Quadratic	0.1	-100	10000	9980.01	-19.99
same	Cubic	0	-10	-1000	-1000	0
same	Cubic	0	-7	-343	-343	0
same	Cubic	0	-5	-125	-125	0
same	Cubic	0	-2	-8	-8	0
same	Cubic	0	-1	-1	-1	0
same	Cubic	0	-0.75	-0.421875	-0.421875	0
same	Cubic	0	-0.25	-0.015625	-0.015625	0
same	Cubic	0	-0.0001	-1E-12	-1E-12	0
same	Cubic	0	0	0	0	0
same	Cubic	0	0.25	0.015625	0.015625	0
same	Cubic	0	0.5	0.125	0.125	0
same	Cubic	0	1	1	1	0
same	Cubic	0	2	8	8	0
same	Cubic	0	3.1416	31.0064942	31.0064942	0
same	Cubic	0	7	343	343	0
same	Cubic	0	10	1000	1000	0
same	Cubic	0	50	125000	125000	0
same	Cubic	0	100	1000000	1000000	0
same	Cubic	0	-50	-125000	-125000	0
same	Cubic	0	-100	-1000000	-1000000	0
same	Cubic	0.1	-10	-1000	-970.299	29.701
same	Cubic	0.1	-7	-343	-328.509	14.491
same	Cubic	0.1	-5	-125	-117.649	7.351
same	Cubic	0.1	-2	-8	-6.859	1.141
same	Cubic	0.1	-1	-1	-0.729	0.271
same	Cubic	0.1	-0.75	-0.421875	-0.274625	0.14725
same	Cubic	0.1	-0.25	-0.015625	-0.003375	0.01225
same	Cubic	0.1	-0.0001	-1E-12	0.000997003	0.000997003
same	Cubic	0.1	0	0	0.001	0.001
same	Cubic	0.1	0.25	0.015625	0.042875	0.02725

same	Cubic	0.1	0.5	0.125	0.216	0.091
same	Cubic	0.1	1	1	1.331	0.331
same	Cubic	0.1	2	8	9.261	1.261
same	Cubic	0.1	3.1416	31.0064942	34.06263737	3.056143168
same	Cubic	0.1	7	343	357.911	14.911
same	Cubic	0.1	10	1000	1030.301	30.301
same	Cubic	0.1	50	125000	125751.501	751.501
same	Cubic	0.1	100	1000000	1003003.001	3003.001
same	Cubic	0.1	-50	-125000	-124251.499	748.501
same	Cubic	0.1	-100	-1000000	-997002.999	2997.001
same	Quartic	0	-10	10000	10000	0
same	Quartic	0	-7	2401	2401	0
same	Quartic	0	-5	625	625	0
same	Quartic	0	-2	16	16	0
same	Quartic	0	-1	1	1	0
same	Quartic	0	-0.75	0.31640625	0.31640625	0
same	Quartic	0	-0.25	0.00390625	0.00390625	0
same	Quartic	0	-0.0001	1E-16	1E-16	0
same	Quartic	0	0	0	0	0
same	Quartic	0	0.25	0.00390625	0.00390625	0
same	Quartic	0	0.5	0.0625	0.0625	0
same	Quartic	0	1	1	1	0
same	Quartic	0	2	16	16	0
same	Quartic	0	3.1416	97.41000218	97.41000218	0
same	Quartic	0	7	2401	2401	0
same	Quartic	0	10	10000	10000	0
same	Quartic	0	50	6250000	6250000	0
same	Quartic	0	100	100000000	100000000	0
same	Quartic	0	-50	6250000	6250000	0
same	Quartic	0	-100	100000000	100000000	0
same	Quartic	0.1	-10	10000	9605.9601	-394.0399
same	Quartic	0.1	-7	2401	2266.7121	-134.2879
same	Quartic	0.1	-5	625	576.4801	-48.5199
same	Quartic	0.1	-2	16	13.0321	-2.9679
same	Quartic	0.1	-1	1	0.6561	-0.3439
same	Quartic	0.1	-0.75	0.31640625	0.17850625	-0.1379
same	Quartic	0.1	-0.25	0.00390625	0.00050625	-0.0034
same	Quartic	0.1	-0.0001	1E-16	9.96006E-05	9.96006E-05
same	Quartic	0.1	0	0	0.0001	0.0001
same	Quartic	0.1	0.25	0.00390625	0.01500625	0.0111
same	Quartic	0.1	0.5	0.0625	0.1296	0.0671
same	Quartic	0.1	1	1	1.4641	0.4641

same	Quartic	0.1	2	16	19.4481	3.4481
same	Quartic	0.1	3.1416	97.41000218	110.4174453	13.00744311
same	Quartic	0.1	7	2401	2541.1681	140.1681
same	Quartic	0.1	10	10000	10406.0401	406.0401
same	Quartic	0.1	50	6250000	6300150.2	50150.2001
same	Quartic	0.1	100	100000000	100400600.4	400600.4001
same	Quartic	0.1	-50	6250000	6200149.8	-49850.1999
same	Quartic	0.1	-100	100000000	99600599.6	-399400.3999
random_extra	Quadratic	0	-145.804735	21259.02075	21259.02075	0
random_extra	Quadratic	0	155.140681	24068.6309	24068.6309	0
random_extra	Quadratic	0	173.042256	29943.62236	29943.62236	0
random_extra	Quadratic	0	-21.772734	474.0519458	474.0519458	0
random_extra	Quadratic	0	-44.705782	1998.606944	1998.606944	0
random_extra	Quadratic	0	-96.961426	9401.518132	9401.518132	0
random_extra	Quadratic	0	62.947034	3962.329089	3962.329089	0
random_extra	Quadratic	0	-2.953225	8.721537901	8.721537901	0
random_extra	Quadratic	0	185.695368	34482.7697	34482.7697	0
random_extra	Quadratic	0	120.39379	14494.66467	14494.66467	0
random_extra	Quadratic	0	-17.917886	321.0506387	321.0506387	0
random_extra	Quadratic	0	120.423235	14501.75553	14501.75553	0
random_extra	Quadratic	0	-183.312811	33603.58668	33603.58668	0
random_extra	Quadratic	0	107.783149	11617.20721	11617.20721	0
random_extra	Quadratic	0	-198.731553	39494.23016	39494.23016	0
random_extra	Quadratic	0	-82.876243	6868.471654	6868.471654	0
random_extra	Quadratic	0	44.365733	1968.318265	1968.318265	0
random_extra	Quadratic	0	165.210957	27294.66031	27294.66031	0
random_extra	Quadratic	0	-79.953998	6392.641796	6392.641796	0
random_extra	Quadratic	0	-100.560545	10112.42321	10112.42321	0
random_extra	Quadratic	0.1	-145.804735	21259.02075	21229.8698	-29.150947
random_extra	Quadratic	0.1	155.140681	24068.6309	24099.66904	31.0381362
random_extra	Quadratic	0.1	173.042256	29943.62236	29978.24081	34.6184512
random_extra	Quadratic	0.1	-21.772734	474.0519458	469.707399	-4.3445468
random_extra	Quadratic	0.1	-44.705782	1998.606944	1989.675788	-8.9311564
random_extra	Quadratic	0.1	-96.961426	9401.518132	9382.135847	-19.3822852
random_extra	Quadratic	0.1	62.947034	3962.329089	3974.928496	12.5994068
random_extra	Quadratic	0.1	-2.953225	8.721537901	8.140892901	-0.580645
random_extra	Quadratic	0.1	185.695368	34482.7697	34519.91877	37.1490736

random_extra	Quadratic	0.1	120.39379	14494.66467	14518.75343	24.088758
random_extra	Quadratic	0.1	-17.917886	321.0506387	317.4770615	-3.5735772
random_extra	Quadratic	0.1	120.423235	14501.75553	14525.85017	24.094647
random_extra	Quadratic	0.1	-183.312811	33603.58668	33566.93411	-36.6525622
random_extra	Quadratic	0.1	107.783149	11617.20721	11638.77384	21.5666298
random_extra	Quadratic	0.1	-198.731553	39494.23016	39454.49385	-39.7363106
random_extra	Quadratic	0.1	-82.876243	6868.471654	6851.906405	-16.5652486
random_extra	Quadratic	0.1	44.365733	1968.318265	1977.201411	8.8831466
random_extra	Quadratic	0.1	165.210957	27294.66031	27327.7125	33.0521914
random_extra	Quadratic	0.1	-79.953998	6392.641796	6376.660997	-15.9807996
random_extra	Quadratic	0.1	-100.560545	10112.42321	10092.3211	-20.102109
random_extra	Cubic	0	-145.804735	-3099665.887	-3099665.887	0
random_extra	Cubic	0	155.140681	3734023.789	3734023.789	0
random_extra	Cubic	0	173.042256	5181511.966	5181511.966	0
random_extra	Cubic	0	-21.772734	-10321.40692	-10321.40692	0
random_extra	Cubic	0	-44.705782	-89349.28635	-89349.28635	0
random_extra	Cubic	0	-96.961426	-911584.6046	-911584.6046	0
random_extra	Cubic	0	62.947034	249416.8639	249416.8639	0
random_extra	Cubic	0	-2.953225	-25.75666377	-25.75666377	0
random_extra	Cubic	0	185.695368	6403290.608	6403290.608	0
random_extra	Cubic	0	120.39379	1745067.614	1745067.614	0
random_extra	Cubic	0	-17.917886	-5752.548745	-5752.548745	0
random_extra	Cubic	0	120.423235	1746348.314	1746348.314	0
random_extra	Cubic	0	-183.312811	-6159967.933	-6159967.933	0
random_extra	Cubic	0	107.783149	1252139.176	1252139.176	0
random_extra	Cubic	0	-198.731553	-7848749.694	-7848749.694	0
random_extra	Cubic	0	-82.876243	-569233.1258	-569233.1258	0
random_extra	Cubic	0	44.365733	87325.88259	87325.88259	0
random_extra	Cubic	0	165.210957	4509376.951	4509376.951	0
random_extra	Cubic	0	-79.953998	-511117.2694	-511117.2694	0
random_extra	Cubic	0	-100.560545	-1016910.789	-1016910.789	0

random_extra	Cubic	0.1	-145.804735	-3099665.887	-3093292.554	6373.333082
random_extra	Cubic	0.1	155.140681	3734023.789	3741249.033	7225.244491
random_extra	Cubic	0.1	173.042256	5181511.966	5190500.245	8988.278976
random_extra	Cubic	0.1	-21.772734	-10321.40692	-10179.84352	141.5634017
random_extra	Cubic	0.1	-44.705782	-89349.28635	-88751.04444	598.2419098
random_extra	Cubic	0.1	-96.961426	-911584.6046	-908767.057	2817.547597
random_extra	Cubic	0.1	62.947034	249416.8639	250607.452	1190.588138
random_extra	Cubic	0.1	-2.953225	-25.75666377	-23.22779915	2.52886462
random_extra	Cubic	0.1	185.695368	6403290.608	6413641.011	10350.40277
random_extra	Cubic	0.1	120.39379	1745067.614	1749419.627	4352.012215
random_extra	Cubic	0.1	-17.917886	-5752.548745	-5656.77009	95.77865503
random_extra	Cubic	0.1	120.423235	1746348.314	1750702.454	4354.140355
random_extra	Cubic	0.1	-183.312811	-6159967.933	-6149892.356	10075.57762
random_extra	Cubic	0.1	107.783149	1252139.176	1255627.572	3488.396657
random_extra	Cubic	0.1	-198.731553	-7848749.694	-7836907.386	11842.3081
random_extra	Cubic	0.1	-82.876243	-569233.1258	-567175.0696	2058.056209
random_extra	Cubic	0.1	44.365733	87325.88259	87917.71004	591.8274514
random_extra	Cubic	0.1	165.210957	4509376.951	4517570.307	8193.355423
random_extra	Cubic	0.1	-79.953998	-511117.2694	-509201.8745	1915.394919
random_extra	Cubic	0.1	-100.560545	-1016910.789	-1013880.078	3030.711147
random_extra	Quartic	0	-145.804735	-451945963.2	-451945963.2	0
random_extra	Quartic	0	155.140681	579298993.5	579298993.5	0
random_extra	Quartic	0	173.042256	896620520.1	896620520.1	0
random_extra	Quartic	0	-21.772734	224725.2473	224725.2473	0
random_extra	Quartic	0	-44.705782	3994429.718	3994429.718	0
random_extra	Quartic	0	-96.961426	88388543.19	88388543.19	0
random_extra	Quartic	0	62.947034	15700051.81	15700051.81	0
random_extra	Quartic	0	-2.953225	76.06522335	76.06522335	0
random_extra	Quartic	0	185.695368	1189061406	1189061406	0
random_extra	Quartic	0	120.39379	210095303.9	210095303.9	0
random_extra	Quartic	0	-17.917886	103073.5126	103073.5126	0
random_extra	Quartic	0	120.423235	210300913.4	210300913.4	0

random_extra	Quartic	0	-183.312811	1129201038	1129201038	0
random_extra	Quartic	0	107.783149	134959503.3	134959503.3	0
random_extra	Quartic	0	-198.731553	1559794216	1559794216	0
random_extra	Quartic	0	-82.876243	47175902.86	47175902.86	0
random_extra	Quartic	0	44.365733	3874276.791	3874276.791	0
random_extra	Quartic	0	165.210957	744998481.6	744998481.6	0
random_extra	Quartic	0	-79.953998	40865869.13	40865869.13	0
random_extra	Quartic	0	-100.560545	102261103.2	102261103.2	0
random_extra	Quartic	0.1	-145.804735	451945963.2	450707371.8	-1238591.397
random_extra	Quartic	0.1	155.140681	579298993.5	580794047.7	1495054.254
random_extra	Quartic	0.1	173.042256	896620520.1	898694922.2	2074402.096
random_extra	Quartic	0.1	-21.772734	224725.2473	220625.0407	-4100.206642
random_extra	Quartic	0.1	-44.705782	3994429.718	3958809.741	-35619.97685
random_extra	Quartic	0.1	-96.961426	88388543.19	88024473.05	-364070.1385
random_extra	Quartic	0.1	62.947034	15700051.81	15800056.55	100004.7372
random_extra	Quartic	0.1	-2.953225	76.06522335	66.27413722	-9.791086133
random_extra	Quartic	0.1	185.695368	1189061406	1191624792	2563385.952
random_extra	Quartic	0.1	120.39379	210095303.9	210794201.1	698897.2073
random_extra	Quartic	0.1	-17.917886	103073.5126	100791.6846	-2281.828031
random_extra	Quartic	0.1	120.423235	210300913.4	211000323.3	699409.9127
random_extra	Quartic	0.1	-183.312811	1129201038	1126739066	-2461971.691
random_extra	Quartic	0.1	107.783149	134959503.3	135461056.5	501553.1339
random_extra	Quartic	0.1	-198.731553	1559794216	1556657085	-3137131.019
random_extra	Quartic	0.1	-82.876243	47175902.86	46948621.39	-227281.4734
random_extra	Quartic	0.1	44.365733	3874276.791	3909325.421	35048.62969
random_extra	Quartic	0.1	165.210957	744998481.6	746803870.7	1805389.121
random_extra	Quartic	0.1	-79.953998	40865869.13	40661805.47	-204063.669
random_extra	Quartic	0.1	-100.560545	102261103.2	101854945.2	-406157.9725

Figure 1: Quadratic Function (Offset = 0.0)

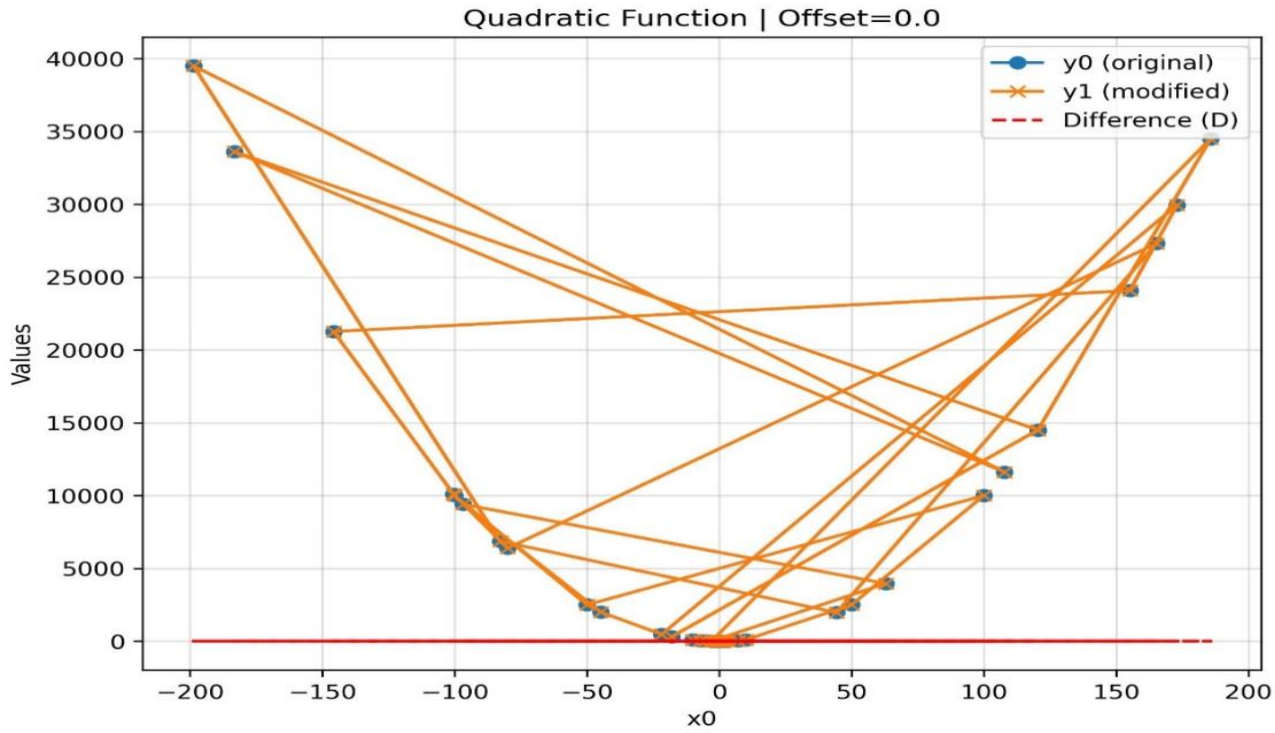


Figure 2: Quadratic Function (Offset = 0.1)

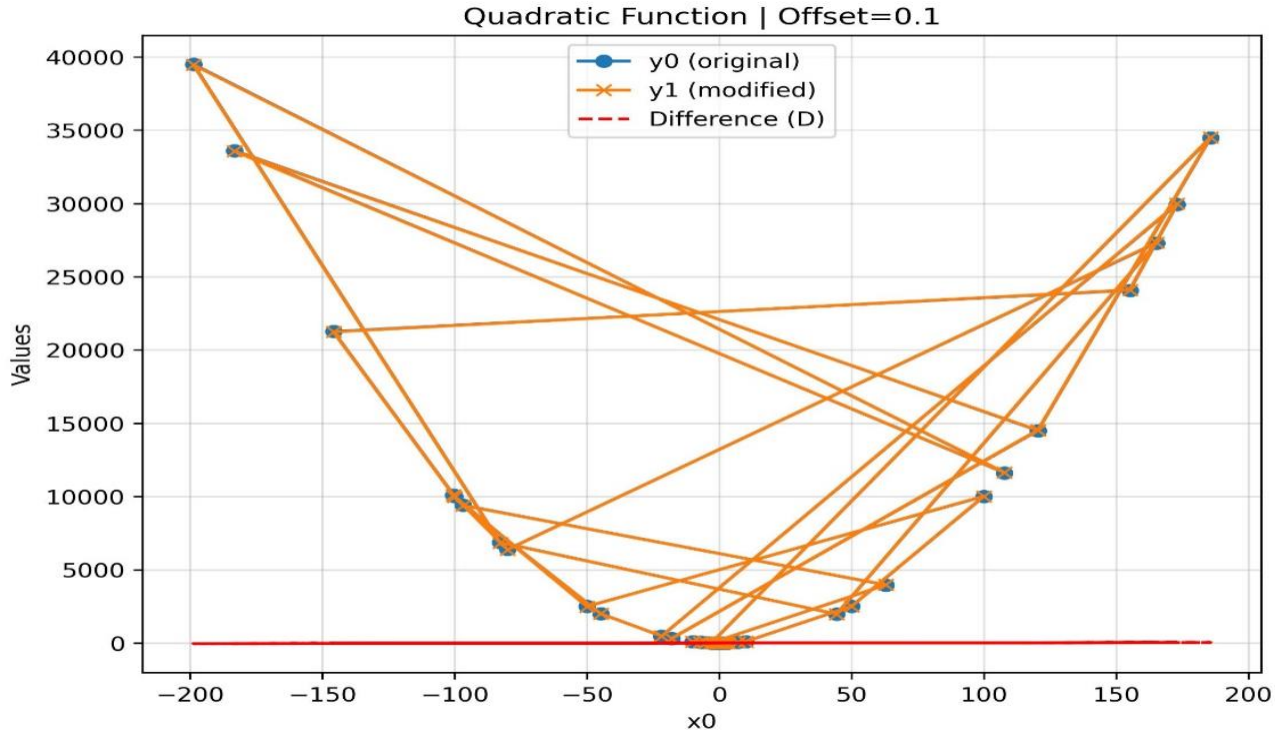


Figure 3: Cubic Function (Offset = 0.0)

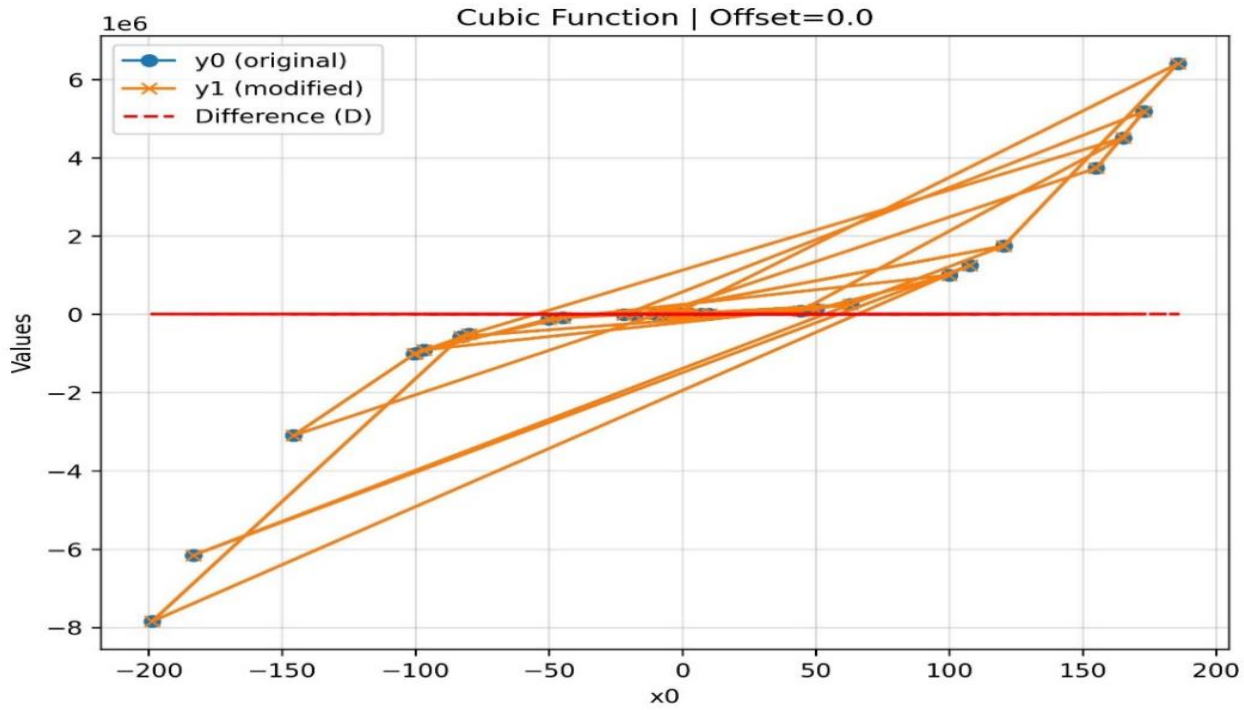


Figure 4: Cubic Function (Offset = 0.1)

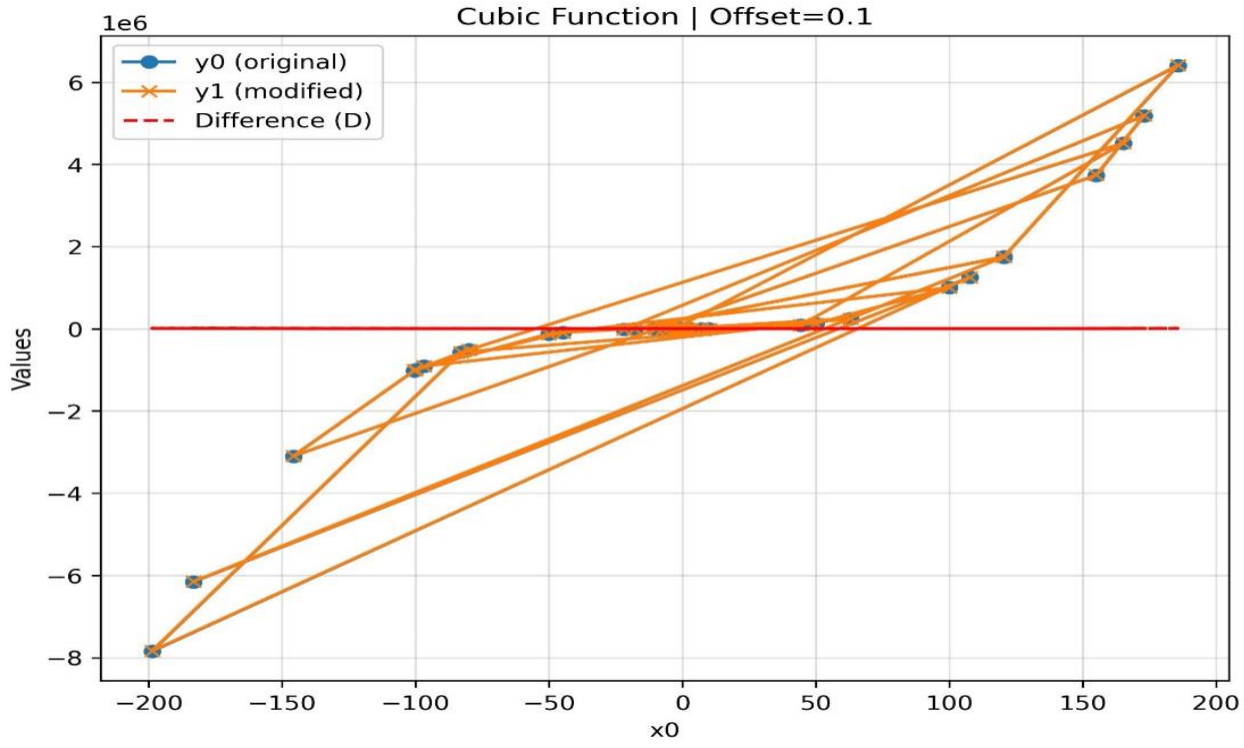


Figure 5: Quartic Function (Offset = 0.0)

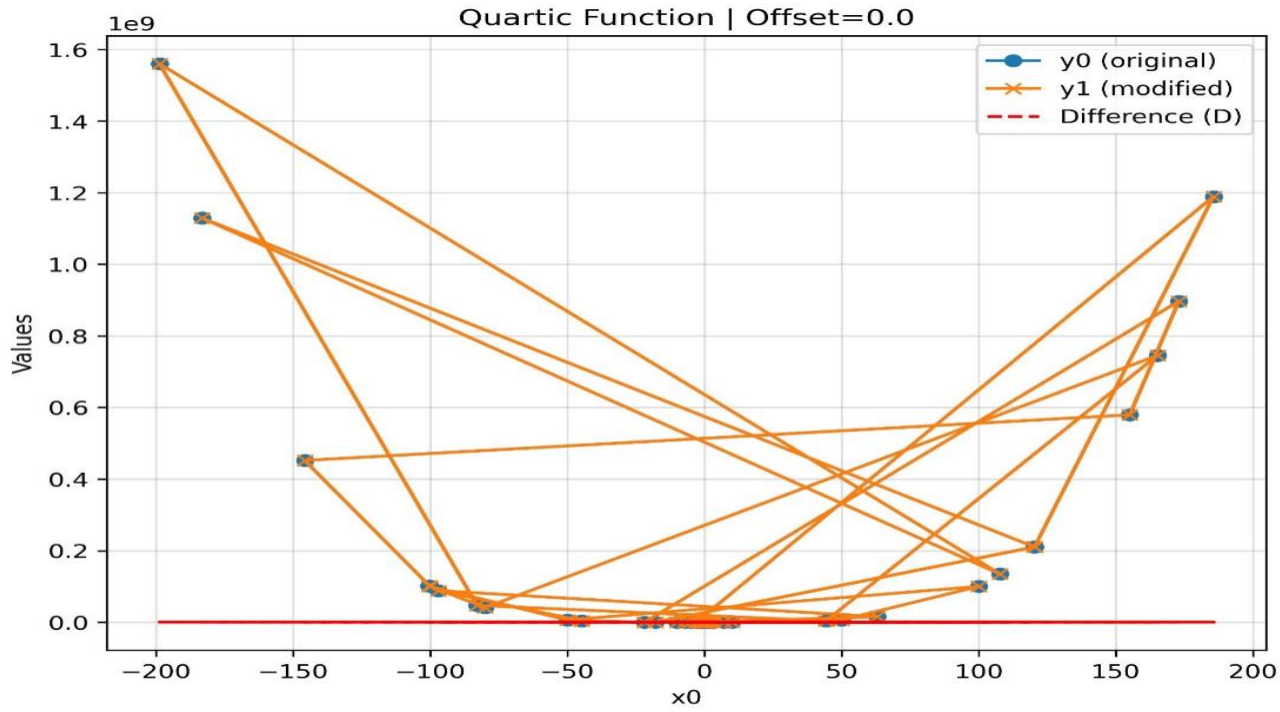
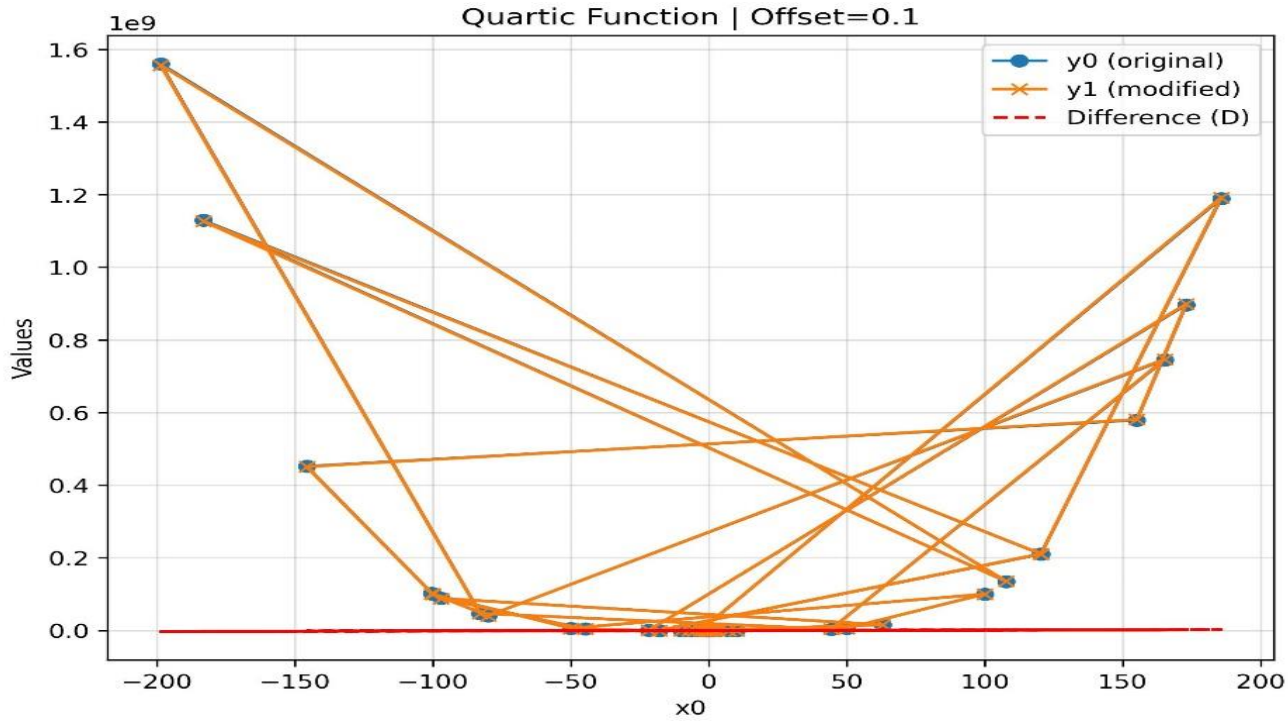


Figure 6: Quartic Function (Offset = 0.1)



Interpretation: The analysis of the polynomial (D) values shows that when the offset is 0, the computed and expected values match perfectly for all polynomial functions, resulting in (D = 0). Introducing a small offset of 0.1 causes deviations, with (D) increasing in magnitude proportionally to both the offset and the input value. Higher-degree polynomials (Cubic, Quartic) exhibit larger deviations compared to Quadratic, reflecting greater sensitivity to small input perturbations. The pattern of (D) is symmetric for positive and negative inputs, indicating consistent behaviour across the domain. Overall, the (D) values quantify the impact of small input offsets, highlighting the increasing error amplification with polynomial degree and input magnitude.

Appendix

1. Implementation of ABSM in Python: main.py

```
import math


def P_linear(x, a=1, b=0):
    return a*x + b


def P_sine(x):
    return math.sin(x)


def P_tanh(x):
    return math.tanh(x)
def P_relu(x):
    return max(0, x)


def P_gaussian(x, sigma=1):
    return math.exp(-x**2 / (2*sigma**2))
```

```
def integral(P, x0, x1, steps=1000):
    dx = (x1 - x0)/steps
    area = 0
    for i in range(steps):
        xi = x0 + i*dx
        area += P(xi) * dx
    return area


def ABSM_Grudge_Machine(x0, n0, n1, P):
    # Tentative x1 (can start as x0)
    x1 = x0
    Q = integral(P, x0, x1)
    n_mean = (n0 + n1) / 2
    n = n_mean + Q
    x1 = (n * x0 + n1 - n0) / n
    D = (n * x1 + n0) - (n * x0 + n1)
    return x1, D


# Example usage:
x0 = 5
n0 = 0.1
n1 = 0.2
x1, D = ABSM_Grudge_Machine(x0, n0, n1, P_sine)
print(f"x1 = {x1}, D = {D}")
```

2. Implementation of ABSM in ANSI C: main.c:

```
#include <stdio.h>
#include <math.h>
double P_sine(double x) { return sin(x); }
double P_linear(double x) { return x; }
double P_tanh(double x) { return tanh(x); }
double P_relu(double x) { return x > 0 ? x : 0; }
double P_gaussian(double x, double sigma) { return exp(-x*x/(2*sigma*sigma)); }
double integral(double (*P)(double), double x0, double x1, int steps) {
   double dx = (x1 - x0)/steps;
   double area = 0;
   for(int i=0;i<steps;i++){
      double xi = x0 + i*dx;
      area += P(xi)*dx;
   }
   return area;
}
void ABSM_Grudge_Machine(double x0, double n0, double n1, double (*P)(double),
double* x1, double* D) {
   *x1 = x0;
   double Q = integral(P, x0, *x1, 1000);
   double n_mean = (n0 + n1)/2;
   double n = n_mean + Q;
   *x1 = (n*x0 + n1 - n0)/n;
   *D = (n*(*x1)+n0) - (n*x0+n1);
}
```

```
int main() {
   double x1, D;
   ABSM_Grudge_Machine(5, 0.1, 0.2, P_sine, &x1, &D);
   printf("x1 = %lf, D = %lf\n", x1, D);
   return 0;
}
```

3. Implementation of ABSM in C++: main.cpp:

```
#include <iostream>
#include <cmath>
#include <functional>


double integral(std::function<double(double)> P, double x0, double x1, int steps=1000){
   double dx = (x1 - x0)/steps;
   double area = 0;
   for(int i=0;i<steps;i++){
      double xi = x0 + i*dx;
      area += P(xi)*dx;
   }
   return area;
}
void ABSM_Grudge_Machine(double x0, double n0, double n1,
std::function<double(double)> P, double &x1, double &D){
```

```
x1 = x0;
   double Q = integral(P, x0, x1);
   double n_mean = (n0+n1)/2;
   double n = n_mean + Q;
   x1 = (n*x0 + n1 - n0)/n;
   D = (n*x1+n0) - (n*x0+n1);
}

int main(){
   double x1, D;
   ABSM_Grudge_Machine(5, 0.1, 0.2, [](double x){ return sin(x); }, x1, D);
   std::cout << "x1 = " << x1 << ", D = " << D << std::endl;
}
```

4. Implementation of ABSM in Java: main.java:

```
public class ABSM {
   public static double integral(java.util.function.Function<Double, Double> P, double
x0, double x1, int steps){
      double dx = (x1 - x0)/steps;
      double area = 0;
     for(int i=0;i<steps;i++){
         double xi = x0 + i*dx;
         area += P.apply(xi)*dx;
      }
```

```
public static double[] ABSM_Grudge_Machine(double x0, double n0, double n1,
java.util.function.Function<Double, Double> P){
      double x1 = x0;
      double Q = integral(P, x0, x1, 1000);
      double n_mean = (n0+n1)/2;
      double n = n_mean + Q;
      x1 = (n*x0 + n1 - n0)/n;
      double D = (n*x1+n0) - (n*x0+n1);
      return new double[]{x1,D};
   }

   public static void main(String[] args){
      double[] res = ABSM_Grudge_Machine(5,0.1,0.2, Math::sin);
      System.out.println("x1 = "+res[0]+", D = "+res[1]);
   }
}
```

5. Implementation of ABSM in JavaScript: main.js:

```
function integral(P, x0, x1, steps=1000){
   let dx = (x1-x0)/steps;
   let area = 0;
  for(let i=0;i<steps;i++){
      let xi = x0 + i*dx;
      area += P(xi)*dx;
   }
```

```
return area;
}
function ABSM_Grudge_Machine(x0, n0, n1, P){
  let x1 = x0;
  let Q = integral(P, x0, x1);
  let n_mean = (n0 + n1)/2;
  let n = n_mean + Q;
  x1 = (n*x0 + n1 - n0)/n;
  let D = (n*x1+n0) - (n*x0+n1);
  return {x1: x1, D: D};
}
// Example usage:
let res = ABSM_Grudge_Machine(5, 0.1, 0.2, Math.sin);
console.log(res);
```

6. Time difference to run different codes

   ("The execution time for small inputs is effectively zero on modern hardware, as the computation is extremely lightweight. Measured differences only become noticeable with large input ranges or repeated iterations.")

```
main.py: 0.0 second(s)
main.c: 0.0 second(s)
main.cpp: 0.0 second(s)
ABSM.java : 0.0079208 second(s)
main.js: 0.0002301
```

Github Repository link:

https://github.com/parthib2006/ABSM_Grudge_Machine