Adaptive Balancing Scaling Model (ABSM) for Single Computational Node

Parthib Ghosh

INDEPENDENT SECURITY RESEARCHER

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.

- *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:

$x_1 = \S(x_0); \S: R \to R$

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.*

$D \neq 0$

*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: $P(x) = ax^n+bx^{n-1}+cx^{n-2}+\ldots+C$*
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.*

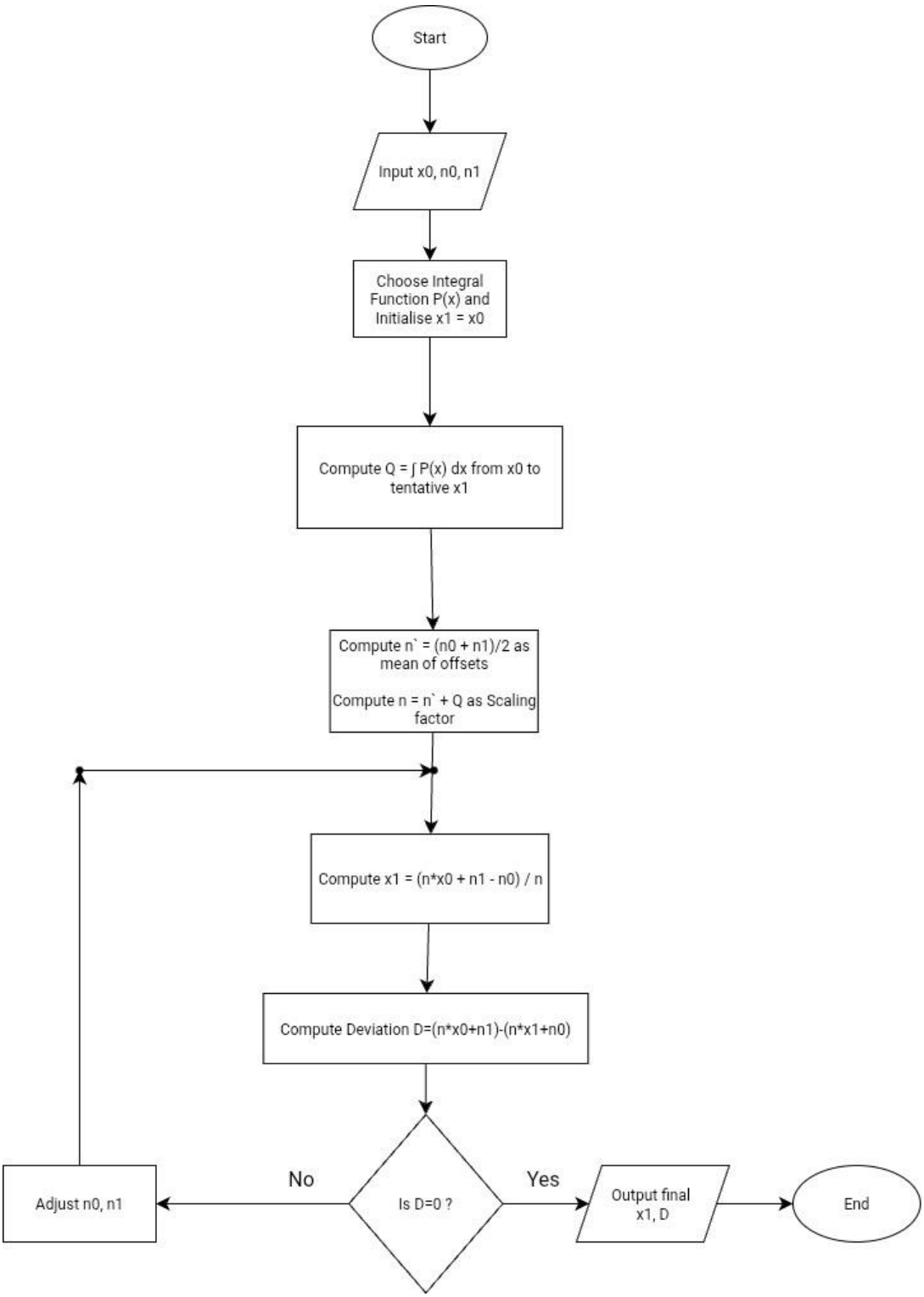
Algorithmic Framework

```
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
```

GitHub Repository link:

https://github.com/parthib2006/ABSM_Grudge_Machine

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
Yes
Adjust n0, n1
Output final x1, D
End

Pseudocode:

Algorithm ABSM_Grudge_Machine

Input: x0 ∈ $\mathbb{R}$

Output: x1 ∈ $\mathbb{R}$, D ∈ $\mathbb{R}$

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.”

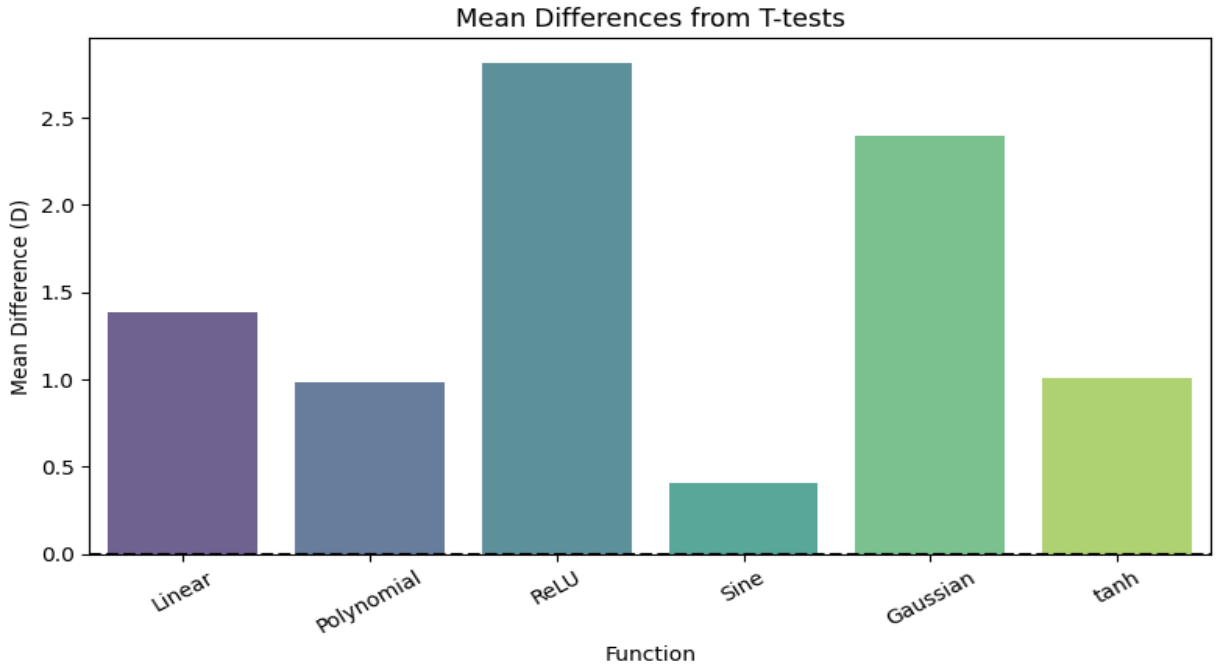
Time Complexity: O(n)

Space Complexity: O(1)

Hypothesis Testing

T-Test

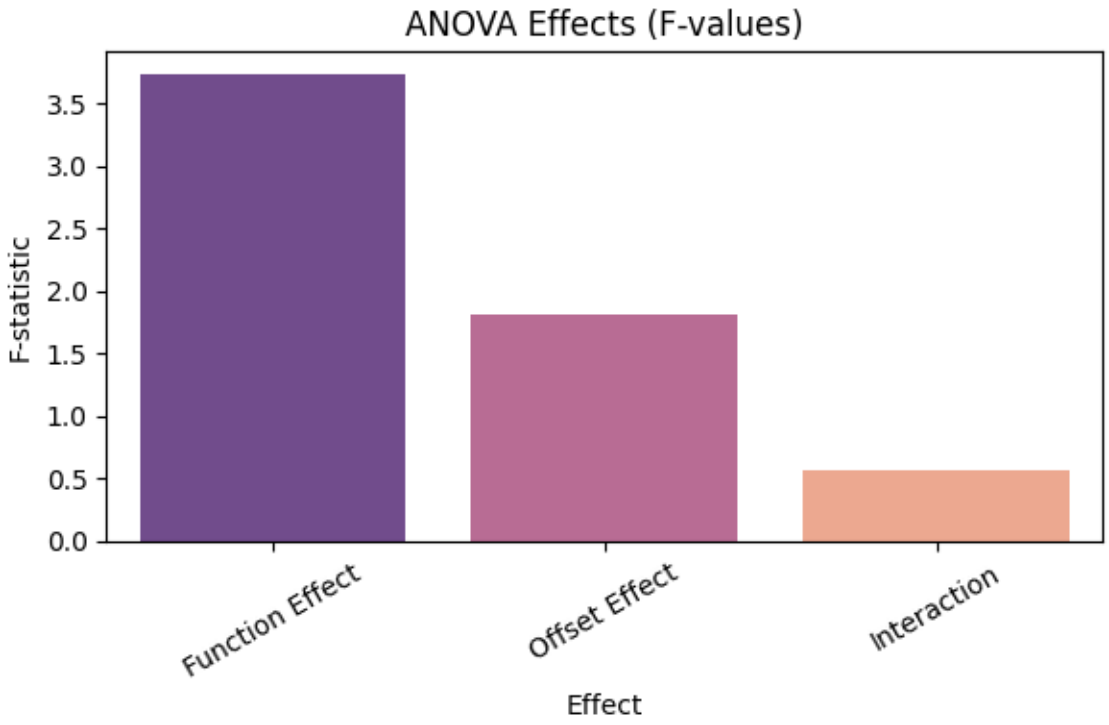
Function	Mean_Difference	t_stat	p_value
Linear	1.381842	-113.552246	2.24E-28
Polynomial	0.985735	-79.979844	1.72E-25
ReLU	2.813715	-290.216854	4.1E-36
Sine	0.404246	-42.583832	2.56E-20
Gaussian	2.393844	-249.82678	7.06E-35
tanh	1.007908	-79.116749	2.12E-25



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



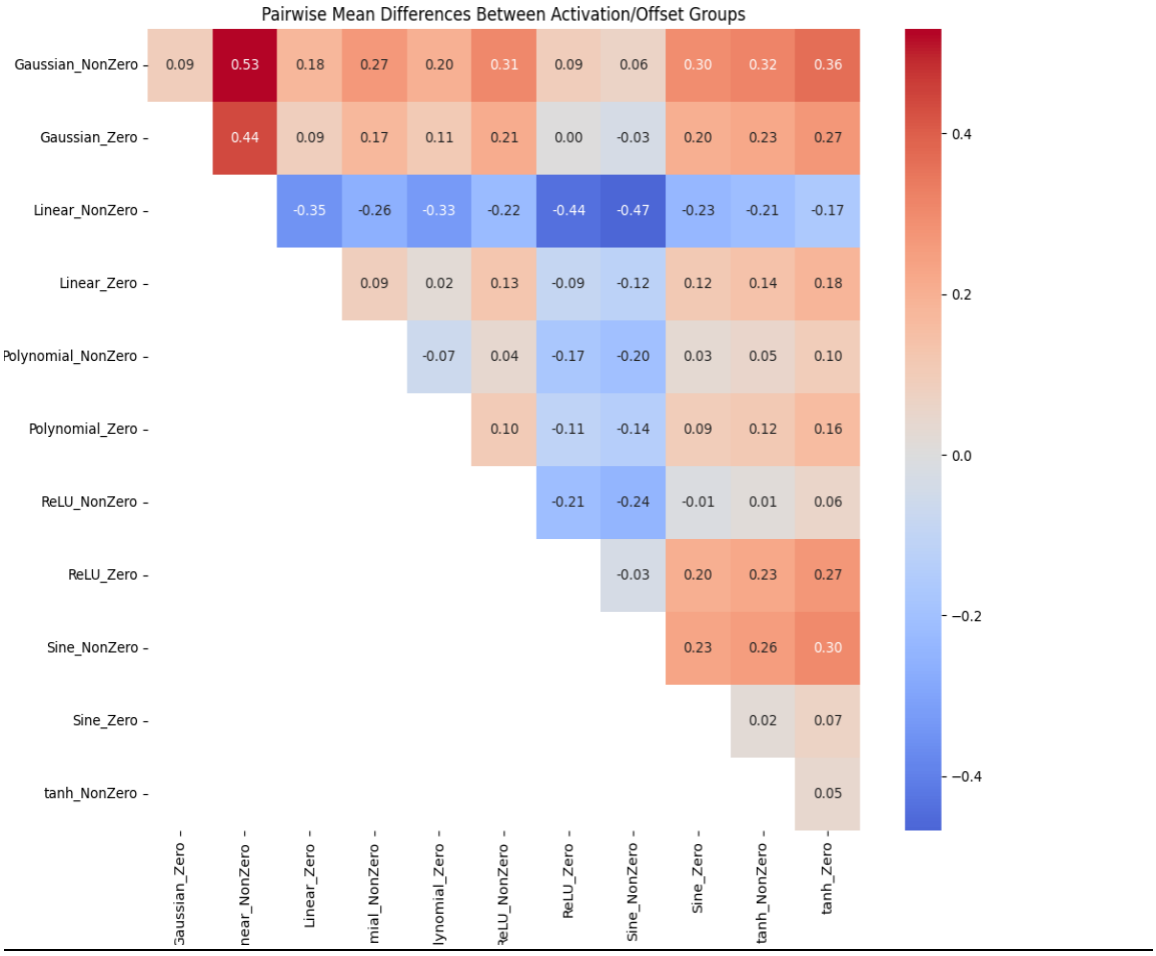
Interpretation: The two-way ANOVA results indicate that the **type of function had a significant effect** on the difference values ($F = 3.73$, $p = 0.0029$). In contrast, the **offset factor alone was not significant** ($F = 1.81$, $p = 0.18$), suggesting that the magnitude of offset does not independently 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

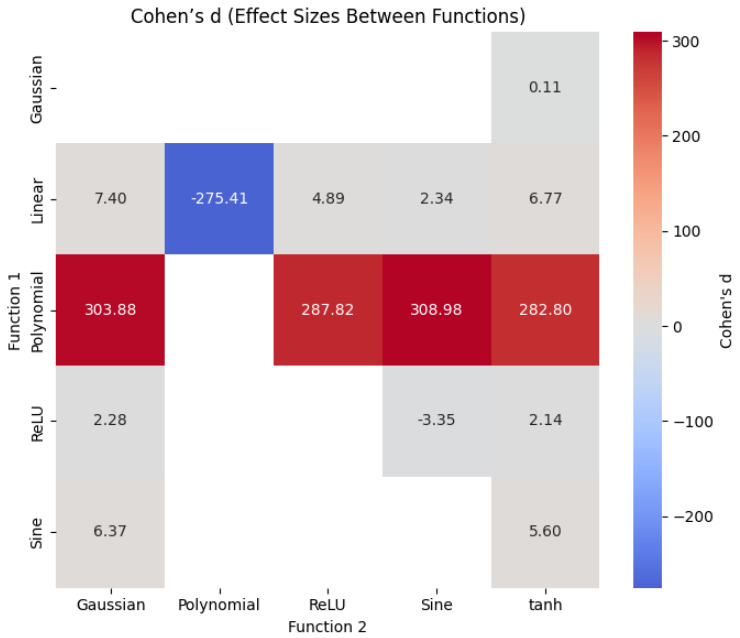
Sine_NonZero	tanh_NonZero	0.2563	0	0.2362	0.2763	TRUE
Sine_NonZero	tanh_Zero	0.3017	0	0.2817	0.3218	TRUE
Sine_Zero	tanh_NonZero	0.0226	0.0152	0.0026	0.0427	TRUE
Sine_Zero	tanh_Zero	0.0681	0	0.048	0.0882	TRUE
tanh_NonZero	tanh_Zero	0.0455	0	0.0254	0.0655	TRUE



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.

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