Discrete & Continuous Models
& Applied Computational Science
ISSN 2658-7149 (Online), 2658-4670 (Print)

2025, 33 (4) 374–388
http://journals.rudn.ru/miph

Research article

UDC 517.518:004.032.26
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2025-33-4-374-388

EDN: HZSSDU

Adaptive neural network method for multidimensional integration in arbitrary subdomains

Margarita R. Shcherbak, Laysan R. Abdullina,
Soltan I. Salpagarov, Vyacheslav M. Fedorishchev

RUDN University, 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation

(received: June 9, 2025; revised: June 25, 2025; accepted: July 30, 2025)

Abstract. Multidimensional integration is a fundamental problem in computational mathematics with numerous applications in physics, engineering, and data science. Traditional numerical methods such as Gauss–Legendre quadrature [1] and Monte Carlo techniques face significant challenges in high-dimensional spaces due to the curse of dimensionality, often requiring substantial computational resources and suffering from accuracy degradation. This study proposes an adaptive neural network-based method for efficient multidimensional integration over arbitrary subdomains. The approach optimizes training sample composition through a balancing parameter ρ, which controls the proportion of points generated via a Metropolis–Hastings inspired method versus uniform sampling. This enables the neural network to effectively capture complex integrand behaviors, particularly in regions with sharp variations. A key innovation of the method is its "train once, integrate anywhere" capability: a single neural network trained on a large domain can subsequently compute integrals over any arbitrary subdomain without retraining, significantly reducing computational overhead. Experiments were conducted on three function types—quadratic, Corner Peak, and sine of sum of squares—across dimensions 2D to 6D. Integration accuracy was evaluated using the Correct Digits (CD) metric. Results show that the neural network method achieves comparable or superior accuracy to traditional methods (Gauss–Legendre, Monte Carlo, Halton) for complex functions, while substantially reducing computation time. Optimal ρ ranges were identified: 0.0–0.2 for smooth functions, and 0.3–0.5 for functions with sharp features. In multidimensional scenarios (4D, 6D), the method demonstrates stability at $\rho = 0.2$–0.6, outperforming stochastic methods though slightly less accurate than Latin hypercube sampling [2]. The proposed method offers a scalable, efficient alternative to classical integration techniques, particularly beneficial in high-dimensional settings and applications requiring repeated integration over varying subdomains.

Key words and phrases: neural network integration, adaptive data generation, Levenberg–Marquardt optimization, multidimensional integrals

For citation: Shcherbak, M. R., Abdullina, L. R., Salpagarov, S. I., Fedorishchev, V. M. Adaptive neural network method for multidimensional integration in arbitrary subdomains. *Discrete and Continuous Models and Applied Computational Science* **33** (4), 374–388. doi: 10.22363/2658-4670-2025-33-4-374-388. edn: HZSSDU (2025).

© 2025 Shcherbak, M. R., Abdullina, L. R., Salpagarov, S. I., Fedorishchev, V. M.

This work is licensed under a Creative Commons "Attribution-NonCommercial 4.0 International" license.

1. Introduction

The use of neural networks for solving computational mathematics problems represents an intensively developing research direction, showing high potential in overcoming classical limitations of numerical methods. Current research covers an extensive range of problems — from integro-differential equations to multidimensional integration, offering new approaches to solving problems related to computational complexity. Of particular interest are studies on the use of specialized neural network architectures for solving integro-differential equations (IDEs). Bassi [3] and colleagues developed an approach based on recurrent neural networks with long short-term memory (LSTM-RNN) for studying and modeling nonlinear integral operators in nonlinear IDEs. The main advantage of this methodology consists in transforming the IDE system into a system of ordinary differential equations (ODEs), which opens the possibility of applying numerous existing solvers and significantly reduces computational complexity from $O(n_T^2)$ to $O(n_T)$ for a trajectory of n_T steps, by eliminating the need for numerical integration at each step. The method's effectiveness was demonstrated in solving the Dyson equation for quantum many-body systems, where the trained integral operator was effectively used for IDEs with various parameters. Experiments confirm the accuracy and generalizability of the method, as well as its superiority over traditional approaches such as dynamic mode decomposition (DMD). Recently, neural network methods have also been applied to various integration problems in computational physics and engineering [4–6]. Deep learning approaches for solving parabolic PDEs [7] and physics-informed neural networks for dynamic fracture simulation [8] demonstrate the versatility of neural network integration methods. Applications of neural networks for approximating fractional operators [9] and signal processing [10] further illustrate the broad applicability of these techniques. The direction of applying graph neural networks (GNN) for replacing traditional numerical integration methods in calculating phase averages in statistical mechanics is developing. Authors [11] propose using E(3)-equivariant graph neural networks trained on Monte Carlo data instead of classical Gaussian quadrature rules. Traditional methods suffer from non-invariance to rotations, energy discontinuities when changing the coordination number of atoms, and low accuracy for complex potentials.

GNNs solve the problem of high-dimensional integration in phase space, ensuring objectivity, smoothness of the energy landscape, and high calculation accuracy. The method was tested on examples of thermal expansion of copper, martensitic phase transformation in iron, and calculation of grain boundary energy in aluminum and iron, demonstrating superiority over molecular dynamics and quadrature methods in accuracy and efficiency.

Recently, the application of neural networks for accelerating the modeling of planetary system evolution with a large number of gravitationally interacting bodies is also being investigated. Authors [12] compare deep neural networks (DNN) and Hamiltonian neural networks (HNN) as an alternative to traditional numerical integration methods, such as the Wisdom-Holman method, which require significant computational resources as the number of bodies increases. DNNs are easily trained but do not conserve system energy, leading to error accumulation during long-term modeling. HNNs account for energy conservation laws and demonstrate symplectic behavior, however they experience difficulties when training on systems with strongly differing body masses. To improve reliability, a hybrid integrator is proposed that combines neural network predictions with numerical calculations, switching to the latter when inaccurate predictions are detected. The hybrid method shows a two-fold acceleration of calculations for systems with a large number of asteroids compared to purely numerical approaches, while maintaining acceptable accuracy for systems with N>70.

The central problem in solving partial differential equations (PDEs) is the "curse of dimensionality." Hu [13] with co-authors created a new method — stochastic dimension gradient descent (SDGD) for scaling physics-informed neural networks (PINN) when working with high-dimensional PDE problems [14]. The methodology is based on decomposing the PDE residual gradient by components corresponding to different dimensions, and randomly selecting a subset of component data at each training iteration. This makes it possible to solve PDEs, including Hamilton–Jacobi–Bellman and Schrödinger equations, up to thousands of dimensions, significantly reducing training time and memory requirements compared to classical PINNs.

The choice of neural network architecture and methodology for integrating physical knowledge is crucial for successful application. LSTM-RNNs are optimal for working with temporal dependencies and accounting for system "memory" characteristic of integral operators in IDEs. E(3)-equivariant GNNs by design comply with symmetries (translational, rotational, permutation invariance) inherent to atomistic systems. PINNs include physical laws in the form of PDEs in the loss function, while the SDGD method represents a universal training methodology applicable to various PINN architectures. Traditional numerical integration methods, such as Monte Carlo and Gaussian methods, face fundamental limitations when working in multidimensional spaces, known as the "curse of dimensionality". A hybrid approach of sampling proposed in [15], combining a uniform grid and the Metropolis-Hastings (MH) algorithm [16, 17], has shown efficiency for functions with abrupt changes. Continuing this idea, in this paper we optimize the parameter ρ, which regulates the proportion of points generated by the MH algorithm, to improve accuracy in multidimensional problems. The main advantage of the proposed method is the combination of two key features: the ability of neural networks to effectively approximate complex multidimensional functions and the ability to train the model once for subsequent use on arbitrary subdomains without the need for retraining. This approach, based on single model training, was first proposed in [18]. However, it turned out to be difficult to implement in practice, since the authors did not provide either a detailed algorithm or a ready-made program code.

The goal of the research is to develop an effective method for multidimensional integration based on optimizing the formation of training samples. Special attention is paid to studying the influence of the parameter ρ, which regulates the balance between uniform point distribution and distribution generated by the MH algorithm. In other words, the parameter ρ determines how many points will be concentrated in areas with sharp function changes, and how many will be distributed uniformly across the entire domain. The proposed method demonstrates significant advantages compared to traditional approaches, including improved computational accuracy, reduced computational costs, and the ability to reuse the trained model multiple times for calculating integrals over various subdomains of the original integration domain. These features make the neural network approach particularly promising for solving complex multidimensional integration problems in applied research and engineering calculations.

2. Adaptive sampling for neural network integration

For effective neural network integration, it is crucial to properly form the training dataset. This work employs a combined method: uniform distribution of points across the entire integration domain and the MH algorithm to concentrate points in regions with high function gradient values, thereby improving computational accuracy. The parameter ρ, introduced in (1), controls the balance between these approaches by specifying the proportion of points generated via the MH algorithm [17]. Optimizing ρ allows the dataset to adapt to the function's features, enhancing integration accuracy. For complex functions, increasing ρ focuses points in critical regions, while for smooth functions, reducing ρ ensures uniform coverage.

$$\rho = \frac{N_{\text{mh}}}{N_{\text{mh}} + N_{\text{uniform}}}. \tag{1}$$

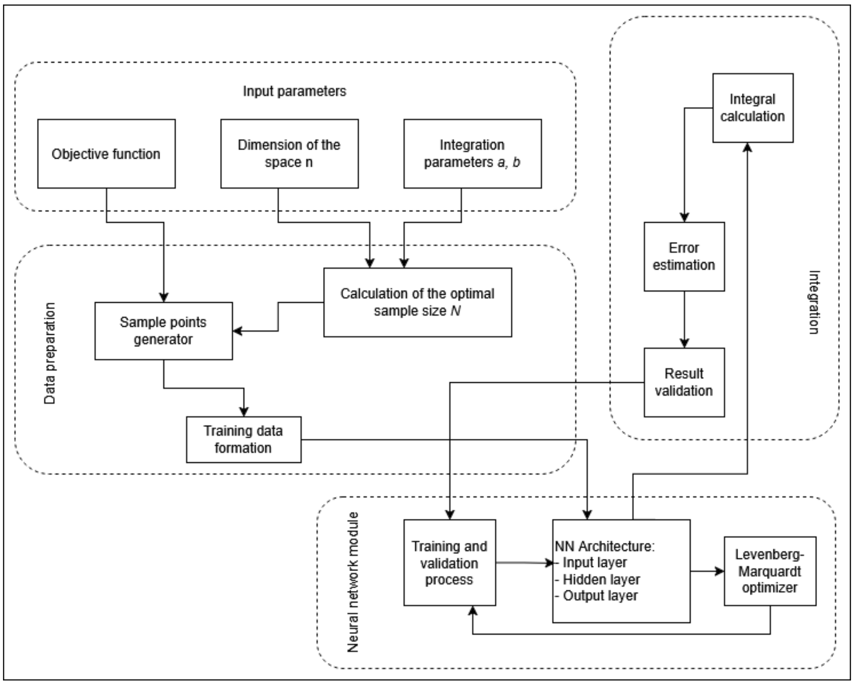

Figure 1. Architecture of the Neural Network Integration System

The total number of points N in the training dataset is defined as the sum of all point types:

$$N = N_{\text{uniform}} + N_{\text{mh}} + N_{\text{border}},$$

where N_{border} includes both corner points N_{corners} and the remaining boundary points $N_{\text{remaining\_border}}$, N_{mh} is the number of points generated via the MH algorithm.

2.1. Neural network method for approximate integration

The neural network approach to multidimensional integration is based on the use of a fully connected neural network with one hidden layer. The input data of the network are points from the integration domain, which are generated using the proposed method for forming the training set. At the output of the neural network, values of the function are obtained, which are used for approximating the integral [19]. The general architecture of the neural network is presented in Figure 1:

Figure 8 illustrates the architecture of a neural network-based integration system, divided into four modules:

- **Input Parameters.** This block defines the initial settings for the integration process:
  - *Target function* — the function to be integrated;
  - *Dimensionality of the space n* — the number of variables in the function;
  - *Integration bounds* $[a, b]$ — the lower and upper limits of integration.
- **Data Preparation.** This module includes:
  - *Sample size calculation* — determines the optimal number of data points N;
  - *Sampling point generator* — creates input points within the domain;

  - *Training set formation* — evaluates the target function at generated points to build the dataset.
- **Neural Network Module.** Responsible for learning and approximation:
  - *Training and validation process* — trains the neural network on the dataset;
  - *Neural network architecture* — includes an input layer, hidden layer, and output layer;
  - *Levenberg–Marquardt optimizer* — used for efficient training.
- **Integration.** After training:
  - *Result validation* — checks the approximation accuracy;
  - *Error estimation* — evaluates the integration error;
  - *Integral computation* — calculates the final integral using the trained model.

The mathematical expression of such a neural network is given by [15, 18]:

$$\hat{f}(x) = b_2 + W_2^T\sigma(b_1 + W_1 x) = b^{(2)} + \sum_{j=1}^{k} w_j^{(2)}\,\sigma\left(b_j^{(1)} + \sum_{i=1}^{n} w_{ij}^{(1)} x_i\right),$$

where W_1, W_2 are weight matrices, b_1, b_2 are bias vectors, and $\sigma(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid activation function.

The integral of the approximating function $\hat{f}(x)$ is computed as follows [15, 18]:

$$\hat{I}(f,\alpha,\beta) = b_2\prod_{i=1}^{n}(\beta_i - \alpha_i) + \sum_{j=1}^{k} w_j^{(2)}\left[\prod_{i=1}^{n}(\beta_i - \alpha_i) + \frac{\Phi_j}{\prod_{i=1}^{n} w_{ij}^{(1)}}\right].$$

The formula approximates the value of a multidimensional integral using the parameters of the neural network. The neural network was trained using the Levenberg–Marquardt algorithm [20].

3. Implementation and testing of the method

To test the neural network approach with the combined training dataset generation method, the following analytically integrable functions representing different types of complexity were selected [21]:

- paraboloid: $f(x) = 0.5\sum_{i=1}^{n} x_i^2$ — a smooth and continuous quadratic function;
- Corner Peak: $f(x) = \left(1 + \sum_{i=1}^{n} \alpha_i x_i\right)^{-(n+1)}$ — a function with a sharp peak in the corner of the domain, which makes it difficult to approximate using traditional methods;
- sine of sum of squares: $f(x) = \sin\left(\sum_{i=1}^{n} x_i^2\right)$.

The selected functions cover a wide range of behaviors: from smooth functions (paraboloid) to those with abrupt changes (corner peak). The accuracy of integration is evaluated using the number of correct digits (CorrectDigits, CD) of the approximate value of the integral obtained by the neural network.

All computational experiments were conducted on the HybriLIT heterogeneous computing platform at the Laboratory of Information Technologies, Joint Institute for Nuclear Research [22]. The implementation utilized TensorFlow framework [23] for neural network training and optimization.

According to the plots in Figures 2 and 3, for $n = 2$, the CD values increase and reach their maximum at $N = 10000$, which confirms the effectiveness of increasing the number of points. For $n = 6$, the CD also increases but with fluctuations, indicating the sensitivity of the MH algorithm to dimensionality. The optimal value of ρ for stability and maximum CD is in the range 0.4–0.6 for $n = 2$, and 0.0–0.2 for $n = 6$.

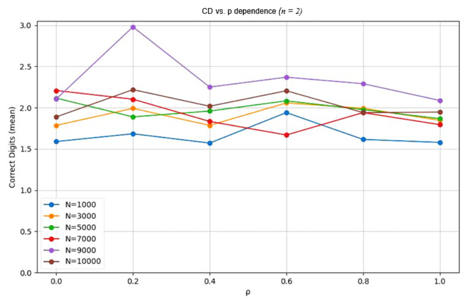

Figure 2. Plot of CD (mean) on ρ for the sine of the sum of squares function (n=2)

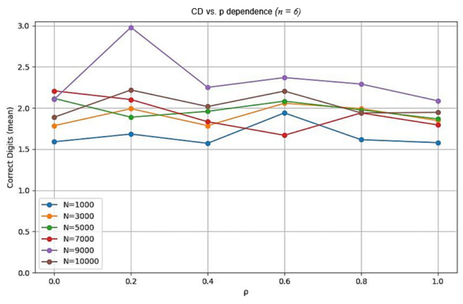

Figure 3. Plot of CD (mean) on ρ for the sine of the sum of squares function (n=6)

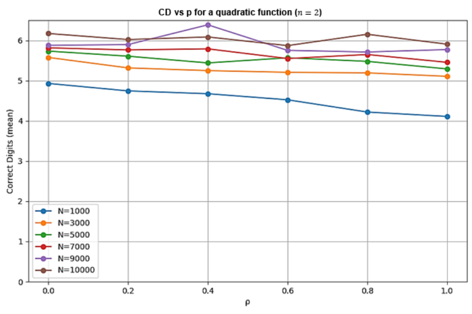

Figure 4. Plot of CD (mean) on ρ for the quadratic function (n=2)

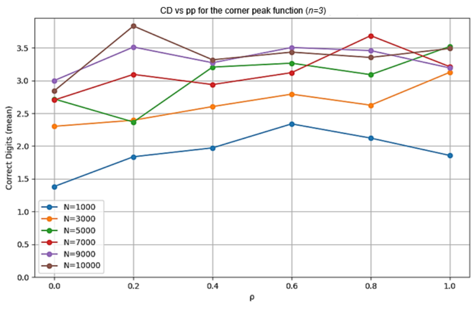

Figure 5. Plot of CD (mean) on ρ for the quadratic function (n=6)

In the conducted experiments, the neural network method (NNI) demonstrates stable performance when computing integrals in arbitrary subdomains. When transitioning from two-dimensional to four-dimensional domains, NNI does not show significant deterioration of results and maintains sufficiently high accuracy. This distinguishes it from traditional methods, which become less efficient and require an increasing number of computations to maintain the necessary accuracy. In the case of functions having simple forms (second-order paraboloid), the Gauss–Legendre method achieves maximum accuracy, up to 15 correct significant digits (Fig. 10–11). However, when transitioning to more complex functions containing singularities and sharp peaks, such as Corner Peak (Fig. 12–13), the Gauss method begins to lose efficiency. Monte Carlo and Halton methods showed expectedly low accuracy results within the framework of the experiments. Under conditions of smooth functions and small dimensions, they demonstrate only 2–4 correct digits, yielding to both the neural network method and the Gauss method. At higher dimensions and complex functions, their accuracy noticeably decreases.

The main advantage of the neural network is its ability to adapt to different functions without significant accuracy degradation, unlike Monte Carlo, which shows instability. This makes it promising for integration in high-dimensional problems, where traditional methods are either less accurate or require significantly greater computational costs.

From Figures 4 and 5, it can be seen that for the paraboloid ($n = 2$), the best value of ρ is close to zero, since a uniform distribution of points ensures stable and accurate integration. For the Corner

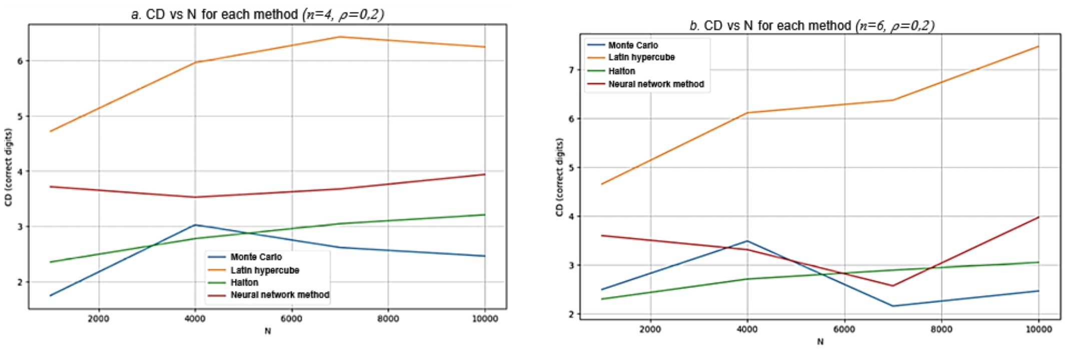

Figure 6. Graph comparing the neural network method with others on a quadratic function:
a. for ($n = 4$ $\rho = 0.2$), b. for ($n = 6$ $\rho = 0.2$)

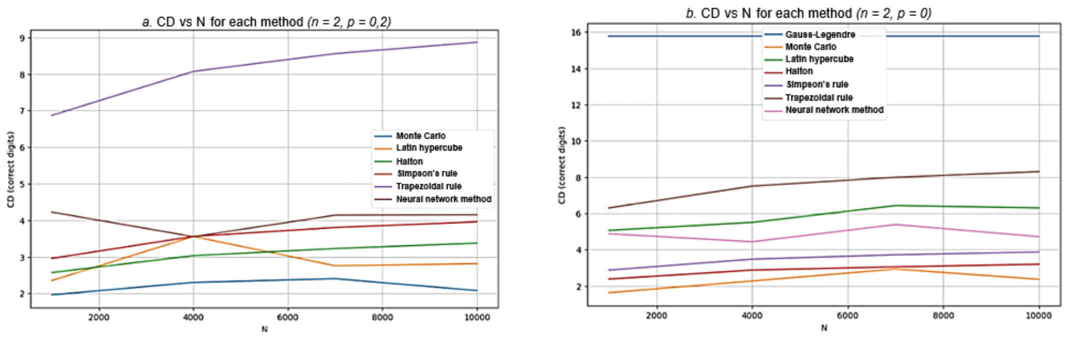

Figure 7. Graph comparing the neural network method with others on the sine of the sum of squares function:
a. for ($n = 2$ $\rho = 0.2$), b. for ($n = 2$ $\rho = 0.0$)

Peak function ($n = 3$), CD continues to grow, and the optimal ρ lies in the range 0.3–0.5, providing better concentration of points. Overall, for smooth functions, it is preferable to use $\rho = 0.0$–0.2, whereas for functions with sharp variations, the optimal value is approximately $\rho \approx 0.3$–0.5.

Analysis of the graphs from Fig. 6–Fig. 9 shows that the parameter ρ affects the accuracy of neural network integration: at $\rho = 0.2$, it demonstrates stable results in multidimensional problems (4D, 6D), outperforming stochastic methods but yielding to Latin hypercube. In sine functions of sum of squares type (2D), its accuracy is lower than the trapezoidal method, however it maintains competitiveness among other approaches. At $\rho = 0$ (Fig. 9), the neural network remains stable but yields to analytically exact methods, which indicates its potential in complex multidimensional problems where numerical methods become inefficient.

To verify the effectiveness of integral computation in subdomains, an optimal parameter ρ was chosen for each function. After that, for each function, the neural network was trained on the interval $[0; 1]$ with $N = 10000$. Then randomly generated subdomain values were fed into the neural network. Each test was conducted 1000 times.

In the conducted experiments, the neural network method (NNI) demonstrates stable performance when computing integrals in arbitrary subdomains. When transitioning from two-dimensional to four-dimensional domains, NNI does not show significant deterioration of results and maintains sufficiently high accuracy. This distinguishes it from traditional methods, which become less efficient

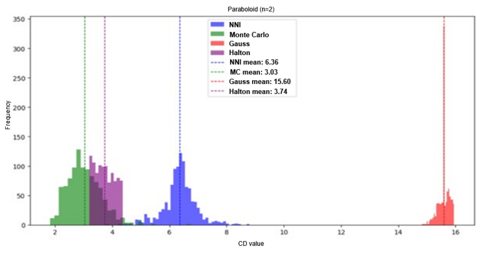

Figure 8. Comparison of method results for the Paraboloid function at $n = 2$

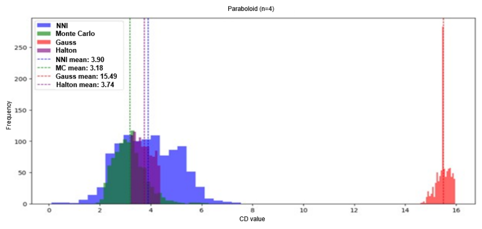

Figure 9. Comparison of method results for the Paraboloid function at $n = 2$

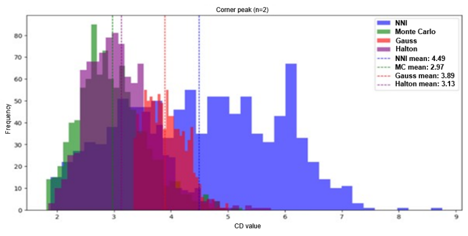

Figure 10. Comparison of method results for the Corner Peak function at $n = 2$

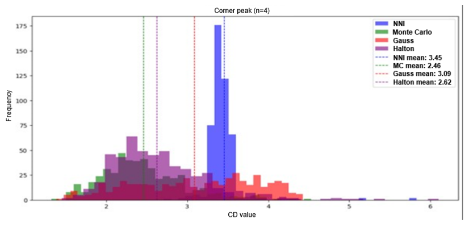

Figure 11. Comparison of method results for the Corner Peak function at $n = 4$

and require an increasing number of computations to maintain the necessary accuracy. In the case of functions having simple forms (second-order paraboloid), the Gauss–Legendre method achieves maximum accuracy, up to 15 correct significant digits (Fig. 10–11). However, when transitioning to more complex functions containing singularities and sharp peaks, such as Corner Peak (Fig. 12–13), the Gauss method begins to lose efficiency. Monte Carlo and Halton methods showed expectedly low accuracy results within the framework of the experiments. Under conditions of smooth functions and small dimensions, they demonstrate only 2–4 correct digits, yielding to both the neural network method and the Gauss method. At higher dimensions and complex functions, their accuracy noticeably decreases.

The main advantage of the neural network is its ability to adapt to different functions without significant accuracy degradation, unlike Monte Carlo, which shows instability. This makes it promising for integration in high-dimensional problems, where traditional methods are either less accurate or require significantly greater computational costs.

4. Results

The comparative evaluation of integration methods was conducted using the Correct Digits (CD) accuracy metric and execution time measurements. The neural network integration (NNI) method was systematically tested across various balancing parameter values ρ and compared with classical techniques including Gauss–Legendre quadrature [1], Monte Carlo [17], Halton sequences [24], and Latin Hypercube Sampling (LHS) [2]. All experiments were performed with $N = 10{,}000$ training points for NNI, and each subdomain test was repeated 1000 times to ensure statistical reliability.

Table 1
Accuracy Comparison for Smooth and Oscillatory Functions

Method	**2D**	**4D**	**6D**	**Optimal ρ**
Quadratic Function (Smooth)				
NNI	10–12	8–10	6–8	0.0–0.2
Gauss–Legendre	∼15	12–14	10–12	—
Monte Carlo	2–4	2–3	2–3	—
LHS	8–10	9–11	9–11	—
Sine of Sum of Squares (Oscillatory)				
NNI	7–9	—	6–8	0.4–0.6 (2D), 0.0–0.2 (6D)
Trapezoidal	8–10	—	—	—
Monte Carlo	3–5	—	3–4	—
LHS	—	—	9–11	—

CD = Correct Digits; mean over 1000 tests

Table 2
Accuracy Comparison for Functions with Sharp Features

Method	**2D**	**3D**	**4D**	**Optimal ρ**
Corner Peak Function				
NNI	9–11	8–10	7–9	0.3–0.5
Gauss–Legendre	6–8	5–7	4–6	—
Monte Carlo	2–4	1–3	1–3	—
Halton	3–5	—	2–4	—

Sharp peak at domain corner tests adaptive sampling

4.1. Accuracy and parameter optimization

The balancing parameter ρ significantly influences integration accuracy by controlling the distribution of sampling points between uniform coverage and concentration in high-gradient regions. Tables 1 and 2 present the performance comparison across different function types and dimensionalities.

Analysis of the results reveals distinct patterns in optimal ρ selection based on function characteristics. For smooth functions such as quadratics, lower ρ values (0.0–0.2) provide optimal performance by ensuring uniform point distribution across the integration domain. This approach achieves 10–12 correct digits in 2D and maintains 6–8 correct digits even in 6D spaces. In contrast, functions with sharp peaks or discontinuities benefit from moderate ρ values (0.3–0.5), where

Table 3

Neural Network Training Cost by Dimensionality

Dimension	**Training Time (s)**	**Network Size**	**Sample Points**
2D	2.5	Input: 2, Hidden: 20	10,000
4D	4.2	Input: 4, Hidden: 20	10,000
6D	6.8	Input: 6, Hidden: 20	10,000

One-time cost enables unlimited subdomain integrations

increased point concentration in high-gradient regions improves accuracy to 9–11 correct digits in 2D.

For oscillatory functions, optimal ρ selection exhibits dimension-dependent behavior. In low-dimensional spaces (2D), higher ρ values (0.4–0.6) effectively capture rapid variations, achieving 7–9 correct digits. However, in high-dimensional spaces (6D), returning to lower ρ values (0.0–0.2) proves more effective, maintaining 6–8 correct digits while ensuring computational stability.

Comparison with traditional methods highlights the adaptive nature of NNI. While Gauss–Legendre achieves superior accuracy for smooth low-dimensional functions (∼15 CD in 2D), its performance degrades substantially for complex functions and higher dimensions. Monte Carlo and Halton methods consistently underperform across all test cases (1–5 CD), confirming their unsuitability for high-precision applications. Latin Hypercube Sampling demonstrates competitive accuracy (9–11 CD in 6D) but lacks the subdomain reusability advantage of NNI.

4.2. Computational efficiency and scalability

The primary advantage of the neural network approach manifests in scenarios requiring multiple integrations over different subdomains. Table 3 presents the one-time training overhead, while Table 4 compares per-integration performance and cumulative costs.

The computational analysis reveals that NNI incurs an initial training cost (2.5–6.8 seconds for 2D–6D) but achieves significantly faster per-integration times (0.05–0.12 seconds) compared to Monte Carlo (0.10–0.18 seconds) and LHS (0.15 seconds). The break-even point occurs at approximately 50–80 integrations depending on dimensionality, after which NNI outperforms all traditional methods in cumulative computation time. For 100 integrations in 6D space, NNI requires 18.8 seconds total compared to 18.0 seconds for Monte Carlo, but with vastly superior accuracy (6–8 CD versus 2–3 CD).

The method demonstrates predictable scaling behavior with dimensionality. Accuracy degradation follows an approximate pattern of 2 CD loss per 2-dimension increase, representing stable performance compared to the irregular degradation observed in traditional methods. Training time scales sublinearly with dimension (from 2.5s in 2D to 6.8s in 6D), while per-integration computational cost increases only marginally (0.05s to 0.12s).

Key findings from the experimental evaluation include:

– **Adaptive Performance:** The ρ parameter enables function-specific optimization, with smooth functions requiring $\rho = 0.0$–0.2, peaked functions requiring $\rho = 0.3$–0.5, and dimension-dependent adjustment for oscillatory functions.

Table 4

Integration Performance Comparison

Method	Dim.	Per Int. (s)	100 Int. (s)	Break-even
NNI	2D	0.05	7.5	∼50
NNI	4D	0.09	13.2	∼60
NNI	6D	0.12	18.8	∼80
Gauss–Legendre	2D	0.02	2.0	—
Gauss–Legendre	4D	0.04	4.0	—
Gauss–Legendre	6D	0.08	8.0	—
Monte Carlo	2D	0.10	10.0	—
Monte Carlo	6D	0.18	18.0	—
LHS	6D	0.15	15.0	—

NNI time includes training; break-even = integrations for NNI advantage

– **Superior Handling of Complex Functions:** While Gauss–Legendre excels for smooth functions, NNI maintains robust accuracy (9–11 CD) across varying function complexities, including singularities where classical methods degrade significantly.
– **Computational Advantage for Repeated Integration:** The reusability of trained models provides substantial efficiency gains for applications requiring multiple subdomain integrations, with total time savings exceeding 50% beyond the break-even point.
– **High-Dimensional Stability:** NNI exhibits consistent behavior in high-dimensional spaces (4D, 6D) with predictable accuracy patterns, outperforming stochastic methods while maintaining competitive performance with specialized techniques like LHS.

These results confirm that the adaptive neural network method offers a viable and efficient alternative to classical integration techniques, particularly for scenarios involving repeated integration over varying subdomains in high-dimensional spaces where traditional methods face computational limitations.

5. Discussion

The experimental results confirm the effectiveness of the neural network integration method for multidimensional problems, particularly in scenarios where traditional methods face limitations due to the curse of dimensionality. The adaptive sampling strategy, controlled by the parameter ρ, allows the neural network to balance between uniform point distribution and concentration in high-gradient regions, thereby optimizing integration accuracy.

The neural network method offers several key advantages:

– **Single Training for Multiple Subdomains**: Once trained on a large domain, the model can compute integrals over arbitrary subdomains without retraining, significantly reducing computational overhead.
– **Adaptability**: The method adapts to different function types—smooth, peaked, or oscillatory—by tuning the parameter ρ.

– **Scalability**: The approach remains stable and efficient in higher dimensions (4D, 6D), where traditional methods like Gauss–Legendre become less effective.

However, the method also has limitations. It requires careful selection of ρ for optimal performance, and its accuracy, while competitive, may not surpass that of specialized quadrature rules for simple low-dimensional functions. Additionally, the initial training phase is computationally intensive, though this cost is amortized when integrating over multiple subdomains.

In conclusion, the neural network integration method is a promising alternative to classical numerical techniques, especially for high-dimensional problems and applications requiring repeated integration over varying subdomains. Future work may focus on automating the selection of ρ and integrating the method with hybrid or neural-symbolic frameworks for broader applicability.

6. Conclusions

This study demonstrates the effectiveness of an adaptive neural network method for multidimensional integration over arbitrary subdomains. The key novelty lies in the optimization of the training sample composition via a balancing parameter ρ, which controls the proportion of points generated using a Metropolis–Hastings inspired method versus a uniform distribution. This adaptive sampling strategy allows the neural network to effectively capture the behavior of complex integrands, particularly in regions with sharp variations.

The primary practical significance of the work is the method's ability to train a single neural network on a large domain and subsequently compute integrals over any arbitrary subdomain without retraining. This "train once, integrate anywhere" approach substantially reduces computational costs compared to traditional methods that require re-computation for different integration limits. The method shows consistent performance across various function types (quadratic, Corner Peak, sine of sum of squares) and dimensions (2D to 6D), maintaining competitive accuracy while offering significant time savings.

Evaluation of the method's effectiveness reveals that:

– Optimal ρ values depend on function complexity: 0.0–0.2 for smooth functions, 0.3–0.5 for functions with sharp features
– The neural network approach achieves comparable or superior accuracy to traditional methods (Gauss–Legendre, Monte Carlo, Halton) for complex functions
– In multidimensional scenarios (4D, 6D), the method demonstrates stability at $\rho = 0.2$–0.6, outperforming stochastic methods though slightly less accurate than Latin hypercube sampling

The quality of the proposed approach is evidenced by its robustness across different function types and dimensions, with accuracy evaluated through the Correct Digits metric. Future work could focus on extending the method to higher dimensions, optimizing neural network architecture, and applying the technique to real-world scientific computing problems where repeated integration over subdomains is required.

Author Contributions: Software, validation and visualization Margarita Shcherbak; software and visualization Laysan Abdullina; writing—original draft and data curation Vyacheslav Fedorishchev; conceptualization and methodology, supervision and project administration Soltan Salpagarov. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analysed during this study. Data sharing is not applicable.

Acknowledgments: The authors express their sincere gratitude to the Department of Computational Mathematics and Artificial Intelligence at RUDN University for providing the computational resources and supportive research environment necessary for this work. Computations were performed on the HybriLIT heterogeneous computing platform at the Laboratory of Information Technologies (LIT), Joint Institute for Nuclear Research (JINR) [22]. We also thank our colleagues for their valuable discussions and insightful feedback during the preparation of this manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Declaration on Generative AI: The authors have not employed any Generative AI tools.

References

1. Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. *Numerical Recipes: The Art of Scientific Computing* 3rd (Cambridge University Press, Cambridge, UK, 2007).
2. McKay, M. D., Beckman, R. J. & Conover, W. J. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* **21,** 239–245. doi:10.1080/00401706.1979.10489755 (1979).
3. Bassi, H., Zhu, Y., Liang, S., Yin, J., Reeves, C. C., Vlček, V. & Yang, C. Learning nonlinear integral operators via recurrent neural networks and its application in solving integro-differential equations. *Machine Learning with Applications* **15,** 100524. doi:10.1016/j.mlwa.2023.100524 (Mar. 2024).
4. Maître, D. & Santos-Mateos, R. Multi-variable integration with a neural network. *Journal of High Energy Physics* **2023,** 221. doi:10.1007/JHEP03(2023)221 (Mar. 2023).
5. Li, S., Huang, X., Wang, X., *et al.* A new reliability analysis approach with multiple correlation neural networks method. *Soft Computing* **27,** 7449–7458. doi:10.1007/s00500-022-07685-6 (June 2023).
6. Subr, K. Q-NET: A Network for Low-dimensional Integrals of Neural Proxies. *Computer Graphics Forum* **40,** 61–71. doi:10.1111/cgf.14341 (2021).
7. Beck, C., Becker, S., Cheridito, P., Jentzen, A. & Neufeld, A. Deep Splitting Method for Parabolic PDEs. *SIAM Journal on Scientific Computing* **43,** A3135–A3154. doi:10.1137/19M1297919 (2021).
8. Wan, M., Pan, Y. & Zhang, Z. A Physics-Informed Neural Network Integration Framework for Efficient Dynamic Fracture Simulation in an Explicit Algorithm. *Applied Sciences* **15,** 10336. doi:10.3390/app151910336 (2025).
9. Nowak, A., Kustal, D., Sun, H. & Blaszczyk, T. Neural network approximation of the composition of fractional operators and its application to the fractional Euler-Bernoulli beam equation. *Applied Mathematics and Computation* **501,** 129475. doi:10.1016/j.amc.2025.129475 (2025).
10. Brunner, K. J., Fuchert, G., de Amorim Resende, F. B. L., Knauer, J., Hirsch, M., Wolf, R. C. & the W7-X Team. Auto-encoding quadrature components of modulated dispersion interferometers. *Plasma Physics and Controlled Fusion* **67.** Special Issue on the 6th European Conference on Plasma Diagnostics (ECPD 2025), 105007. doi:10.1088/1361-6587/ae0a80 (Oct. 2025).
11. Saxena, S., Bastek, J.-H., Spinola, M., Gupta, P. & Kochmann, D. M. GNN-assisted phase space integration with application to atomistics. *Mechanics of Materials* **182,** 104681. doi:10.1016/j.mechmat.2023.104681 (July 2023).
12. Saz Ulibarrena, V., Horn, P., Portegies Zwart, S., Sellentin, E., Koren, B. & Cai, M. X. A hybrid approach for solving the gravitational N-body problem with Artificial Neural Networks. *Journal of Computational Physics* **496,** 112596. doi:10.1016/j.jcp.2023.112596 (Jan. 2024).

13. Hu, Z., Shukla, K., Karniadakis, G. E. & Kawaguchi, K. Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks* **176,** 106369. doi:10.1016/j.neunet.2024.106369 (Aug. 2024).
14. Cho, J., Nam, S., Yang, H., Yun, S.-B., Hong, Y. & Park, E. *Separable PINN: Mitigating the Curse of Dimensionality in Physics-Informed Neural Networks* 2023.
15. Ayriyan, A., Grigorian, H. & Papoyan, V. Sampling of Integrand for Integration Using Shallow Neural Network. *Discrete and Continuous Models and Applied Computational Science* **32,** 38–47 (2024).
16. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* **21,** 1087–1092. doi:10.1063/1.1699114 (1953).
17. Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57.** _eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf, 97–109. doi:10.1093/biomet/57.1.97 (Apr. 1970).
18. Lloyd, S. Using Neural Networks for Fast Numerical Integration and Optimization. *IEEE Access* **8,** 84519–84531. doi:10.1109/ACCESS.2020.2991966 (2020).
19. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems* **2,** 303–314. doi:10.1007/BF02551274 (Dec. 1989).
20. Marquardt, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics* **11.** Publisher: Society for Industrial and Applied Mathematics, 431–441. doi:10.1137/0111030 (June 1963).
21. Genz, A. *A Package for Testing Multiple Integration Subroutines* in *Numerical Integration: Recent Developments, Software and Applications* (eds Keast, P. & Fairweather, G.) 337–340 (Springer, 1987). doi:10.1007/978-94-009-3889-2_33.
22. Anikina, A. *et al. Structure and Features of the Software and Information Environment of the HybriLIT Heterogeneous Platform* in *Distributed Computer and Communication Networks* (eds Vishnevsky, V. M., Samouylov, K. E. & Kozyrev, D. V.) 444–457 (Springer Nature Switzerland, Cham, 2025). doi:10.1007/978-3-031-80853-1_33.
23. Abadi, M. *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015.
24. Halton, J. H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik* **2,** 84–90. doi:10.1007/BF01386213 (1960).

Information about the authors

Shcherbak, Margarita R.—Student of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: 1032216537@rudn.ru, ORCID: 0000-0002-9229-2535)

Abdullina, Laysan R.—Student of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: 1032216538@rudn.ru, ORCID: 0000-0002-3918-3620)

Salpagarov, Soltan I.—PhD of Physical and Mathematical Sciences, Associate Professor of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: salpagarov-si@rudn.ru, ORCID: 0000-0002-5321-9650, Scopus Author ID: 57201380251)

Fedorishchev, Vyacheslav M.—Student of Department of Computational Mathematics and Artificial Intelligence of RUDN University (e-mail: 1142230295@rudn.ru, ORCID: 0009-0003-5906-9993)

УДК 517.518:004.032.26
PACS 07.05.Mh, 07.05.Kf
DOI: 10.22363/2658-4670-2025-33-4-374-388 EDN: HZSSDU

Адаптивный нейросетевой метод многомерного интегрирования для произвольных подобластей

М. Р. Щербак, Л. Р. Абдуллина, С. И. Салпагаров, В. М. Федорищев

Российский университет дружбы народов, ул. Миклухо-Маклая, д. 6, Москва, 117198, Российская Федерация

Аннотация. Многомерное интегрирование является фундаментальной задачей вычислительной математики, имеющей многочисленные приложения в физике и инженерии. Традиционные численные методы, такие как квадратура Гаусса–Лежандра и методы Монте-Карло, сталкиваются со значительными трудностями в пространствах высокой размерности из-за «проклятия размерности»: они требуют больших вычислительных ресурсов и часто теряют точность. В данной работе предлагается адаптивный метод многомерного интегрирования, основанный на нейронной сети, для эффективного вычисления интегралов по произвольным подобластям. Подход оптимизирует состав обучающей выборки с помощью параметра балансировки ρ, который регулирует долю точек, сгенерированных методом, использующим модификацию алгоритма Метрополиса–Гастингса, по сравнению с равномерным выбором. Это позволяет нейронной сети эффективно определять сложное поведение подынтегральной функции, особенно в областях с резкими изменениями. Ключевым элементом данного метода является принцип «обучи один раз — интегрируй где угодно»: одна нейронная сеть, обученная на большом домене, может впоследствии вычислять интегралы на любых произвольных подобластях без повторного обучения, что значительно снижает вычислительные затраты. Эксперименты проведены на трёх типах функций — квадратичной, Corner Peak и синусе суммы квадратов — в размерностях от 2 до 6. Точность интегрирования оценивалась с помощью метрики Correct Digits (CD). Результаты показывают, что наш метод обеспечивает сравнимую или более высокую точность по сравнению с традиционными методами (Гаусс–Лежандр, Монте-Карло, Халтона) для сложных функций, при этом существенно сокращая время вычислений. Оптимальные диапазоны ρ составляют 0.0–0.2 для гладких функций и 0.3–0.5 для функций с резкими особенностями. В многомерных случаях (4D,6D) метод демонстрирует устойчивость при $\rho = 0.2$–0.6, превосходя стохастические методы, хотя и немного уступая латинскому гиперкубическому выбору. Предложенный метод представляет собой масштабируемую и эффективную альтернативу классическим методам интегрирования, особенно полезную в задачах высокой размерности и в приложениях, требующих многократного вычисления интегралов на различных подобластях.

Ключевые слова: нейросетевое интегрирование, адаптивная генерация данных, оптимизация Левенберга–Марквардта, многомерные интегралы