Application of Neural Network Approach to Numerical Integration

Gregory A. Shipunov[1,*], Oksana I. Streltsova[1,2] and Yury L. Kalinovsky[1,2]

[1]Dubna State University, 19 Universitetskaya St, Dubna, 141980, Russian Federation
[2]Joint Institute for Nuclear Research, 6 Joliot-Curie St, Dubna, 141980, Russian Federation

Abstract

This paper is dedicated to the description of the application of neural network approach to numerical integration of functions of one and multiple variables. The essence of the approach is to train a neural network model to approximate the integral function and than use the parameters of the model to numerically calculate the value of the integral using the formulae based on those parameters. The usage of the approach will reduce the amount of calculations (and time) required to get a numerical integration result when the number of integral function's variables is big. Where the common numerical methods become too complex, the numerical approach allows calculations to be less demanding of the computational time and resources. This approach is being tested within the framework of a physics problem of modeling of the particles formation and their properties in the NICA experiment. In this experiment the key problem is to calculate integrals of functions of multiple variables. Currently the author of this paper is developing the framework for integration of functions of two variables. The main goal of the project though is to develop a Python library for numerical integration based on neural network approach.

Keywords

neural networks, numerical integration, meson, NICA, python library

1. Introduction

The problem of numerical integration appears in both theoretical and applied calculations and potentially raises two obstacles. The first one is the complexity of the analytical form of the anti-derivative of the integral function (which is required to get the value of the definite integral). Sometimes it is just hard to produce the anti-derivative of the function and sometimes there are none. In general case there is no general algorithm to get a given function's anti-derivative. Methods of numerical integration exist (e.g. Simpson's method) to solve this problem by providing a formulae to get the value of an integral with some error without knowing the anti-derivative. But as the dimensions of the integral function's grow, the complexity of those methods also grows as well as the error bound. In this case the neural network approach to the numerical integration may help reduce the complexity as well as increase accuracy of the calculation.

Currently, there were several works done in the field of application of neural networks to the numerical integration problem. We here will highlight the paper "Using neural networks for fast numerical integration and optimization" by Lloyd et al.[1]. This work was served as an inspiration for the development of the neural network approach. Another source was the physics problem of modeling of the particles formation and their properties in the NICA experiment which involves solving the equations containing double and triple integrals. The development of this neural network approach to the numerical integration has the main goal of

Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems 2025 (ITTMM 2025), Moscow, April 07–11, 2025

*Corresponding author.
shgregory3@gmail.com (G. A. Shipunov); strel@jinr.ru (O. I. Streltsova); kalinov@jinr.ru (Y. L. Kalinovsky)
0009-0007-7819-641X (G. A. Shipunov); 0000-0003-4522-6735 (O. I. Streltsova); 0000-0002-7596-5531 (Y. L. Kalinovsky)

the development of a software library for the Python programming language with functionality of numerical integration based on neural network approach.

This paper contains several parts. Firstly, we will give the description of one of the possible neural network approaches to the numerical integration as well as highlight other potential approaches. Secondly, the physics problem of modeling of the of the particles formation and their properties in the NICA experiment will be depicted. Thirdly, we will give the observation on the results already reached in the development of the neural network approach to the numerical integration. Lastly, the insight on the future of this project will be given.

2. Neural Network Approach to Numerical Integration

The problem we investigate in this paper is the application of neural network approach to the calculation of approximate value of integral of a function, i.e. numerical integral with usage of neural network approach.

Let a continuous real function $f(\mathbf{x})$ be defined as $f : \mathbb{R}^n \to \mathbb{R}$. Let Ω be a compact subset of $\mathbb{R}^n$ and let G be a bounded convex subset of Ω. Let, also, $\mathbf{x}$ be a vector of n dimensions in Ω. Than

$$I(f) = \int_G f(\mathbf{x}) d\mathbf{x}, \tag{1}$$

is a definite integral of f across set G. Therefore, the problem is to get the $I(f)$ value calculated with use of neural network approach.

The work of Lloyd et al.[1] proposes, that function $f(\mathbf{x})$ can be approximated using an MLP (multilayer perceptron) neural network. Let this network contain of 3 layers with sizes n, k and 1 accordingly. n is defined above as a number of dimensions of the function $f(\mathbf{x})$, k is the hidden-layer size which is set to minimize the approximation error and 1 neuron of the output layer accumulates the approximated $f(\mathbf{x})$ value. Because this value is not equal to the value $f(\mathbf{x})$ with given $\mathbf{x}$, let us denote it as $\hat{f}(\mathbf{x})$ and further on we will refer to the MLP network as $\hat{f}(\mathbf{x})$. Figure 1 depicts the structure of the $\hat{f}(\mathbf{x})$ network.

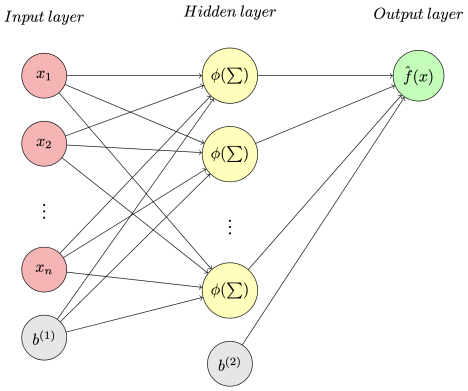


Figure 1: The MLP structure used in the neural network approach

MLP network can have activation functions applied to each neuron's value before its value is propagated forward. In our case both input and output layer has *linear activation function* applied and the hidden layer has *logistic sigmoid function* applied. Last has the following definition:

$$\phi(z) = \frac{1}{1 + \exp(-z)}. \tag{2}$$

With $\phi(z)$ defined, network structure's mathematical form is:

$$\hat{f}(x) = b^{(2)} + \sum_{j=1}^{k} w_j^{(2)} \phi(b_j^{(1)} + \sum_{i=1}^{n} w_{ji}^{(1)} x_i). \tag{3}$$

The trained neural network $\hat{f}(\mathbf{x})$ can be used to get the approximate value of integral (1):

$$\hat{I}(f) = \int_G \hat{f}(\mathbf{x}) d\mathbf{x}. \tag{4}$$

The integration error is defined mainly by k (this is proved in works [1] and [2]). To get the value of $\hat{I}(f)$ let us apply following substitution to the (4):

$$-Li_0(-\exp(z)) = \frac{1}{1+\exp(-z)} = \phi(z), \tag{5}$$

where $Li_0(u(z))$ is a Jonquière's function or the polylogarithm of order 0. This substitution is proved to be numerically accurate in [1].

Let number of dimensions $n = 1$. With substitution (5) equation (3) can be integrated over given boundaries $[\alpha, \beta]$ to produce the following numerical integration formulae for 1-dimensional case:

$$\hat{I}(f) = \int_{\alpha}^{\beta} dx \left(b^{(2)} + \sum_{j=1}^{k} w_j^{(2)} \phi(b_j^{(1)} + w_{1j}^{(1)} x) \right) = b^{(2)}(\beta - \alpha) + \sum_{j=1}^{k} w_j^{(2)} \left((\beta - \alpha) + \frac{\Phi_j}{w_{1j}^{(1)}} \right), \tag{6}$$

$$\Phi_j = Li_1(-\exp[-b_j^{(1)} - w_{1j}^{(1)} \alpha]) - Li_1(-\exp[-b_j^{(1)} - w_{1j}^{(1)} \beta]). \tag{7}$$

Formulae (6-7) can be extrapolated to higher dimensions as it is shown in [1]. The described neural network approach to numerical integration concludes to following steps:

1. generate the dataset based on $f(\mathbf{x})$ and use it to train the neural network model (3) to approximate given $f(\mathbf{x})$;
2. extract parameters of the trained network (3) and use them with (6-7) to calculate the numerical integral.

3. Approach testing

The 6 families of functions for testing the numerical integration were originally proposed in 1984 by Alan Genz[3] (their definitions can be found in the related works). In our work this functions were used to evaluate the accuracy of the neural network approach. The approach was tested on the 1- and 2-dimensional functions. The resulting integrand values were compared to Python *scipy.integrate.quad* method results for the same functions and thus the MAE (mean absolute error) values were produced. Logarithmic-scaled values of MAE for each integrand family is shown on Figure 2. Each of the function was integrated 10 times using neural network approach and *quad* function. The Product Peak and Discontinuous functions were hard to integrate and show the worst MAE results. This is mostly due to this functions being very hard to integrate numerically in general, especially the last one, because (as its name states) it is discontinuous and it is not proved that discontinuous functions can be correctly approximated using MLPs.

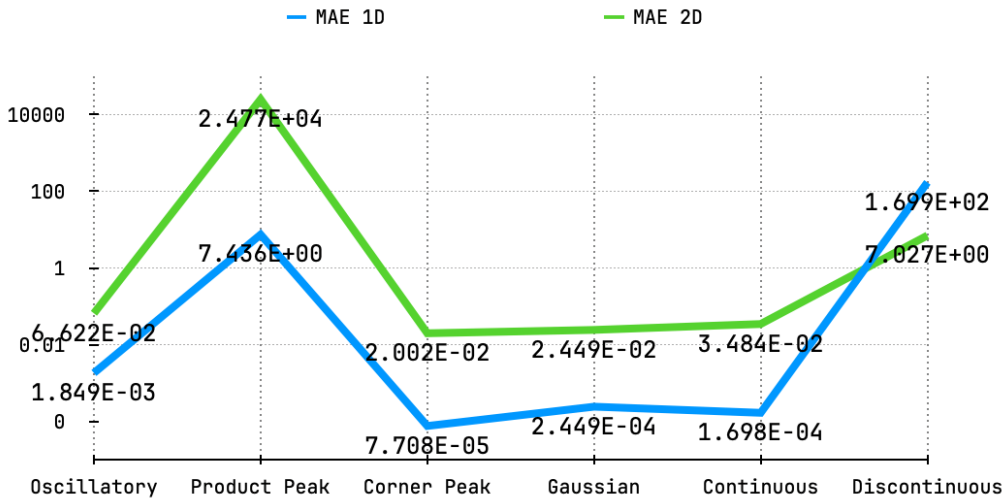


Figure 2: MAE between neural network numerical integration values and *quad* numerical integration values (logarithmic scale)

4. Modeling of the formation of particles and their properties for the NICA experiment

The main task is to build a model for the unified description of both light and heavy mesons formed in heavy ion collisions.

This will be a nonlocal interaction model [4, 5]. The model concludes to the fact that meson (quark-antiquark pair) formation proceeds with a gluon exchange. But the gluon exchange cannot be described as a diagram sum, because perturbation theory cannot be applied here.

Mesons can be described as $q\bar{q}$ bound states by the Bethe-Salpeter equation

$$\Gamma(q,P) = -\frac{4}{3}\int \frac{d^4p}{(2\pi)^4} D(q-p)\gamma_\alpha S_1 \Gamma(q,P) S_2 \gamma_\alpha \,. \tag{8}$$

The vertex function $\Gamma(p,P)$ depends on the relative momentum p, and the total momentum of the bound state P. $S_i(p)$ are the dressed quark propagator in the Euclidean space

$$S_i(p_i) = \frac{1}{i(p_i\cdot\gamma)+m_i} \tag{9}$$

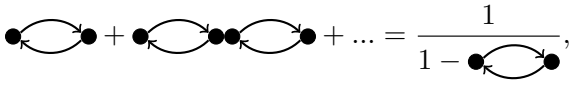
The momenta $p_i = p + q_i$, $q_i = b_i P$, $i = 1,2$ with $b_1 = -m_1/(m_1+m_2)$, $b_2 = m_2/(m_1+m_2)$, m_i are the constituent quark masses.

Equation (8) contains the interaction kernel $D(q-p)$ which describes the effective gluon interaction within a meson. We consider the rank -1 separable model

$$D(q-p) = D_0 F(q^2) F(p^2)\,, \tag{10}$$

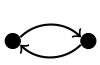
where D_0 is the coupling constant and the function $F(p^2)$ is related to scalar part of Bethe - Salpeter vertex function. We employ $F(q^2)$ in the Gaussian form $F(p^2) = e^{-p^2/\lambda^2}$ with the parameter λ which characterizes the finite size of the meson.

This separable Anzatz of the interaction kernel (10) allows to write the meson observables in the term of the polarization operator (the buble diagram):



(11)

where



$$\rightarrow \int_0^\infty \frac{dp}{2\pi^4} F(p^2) \frac{1}{u_1 u_2} = \int_0^\infty \frac{dp}{2\pi^4} F(p^2) \frac{1}{[(p+q_1)^2 + m_1^2][(p+q_2)^2 + m_2^2]}. \quad (12)$$

To calculate these integrals we use the equations:

$$\frac{1}{(p+q_i)^2 + m_i^2} = \int_0^\infty dt\{\exp(-t[(p+q_i)^2 + m_i^2])\}, \quad (13)$$

$$F(p^2) = \int_0^\infty ds\{\exp(-sp^2)F(s)\}. \quad (14)$$

The case of more than two mesons is described using triple and quadruple integrals.

The amplitudes of particular decays and the squared amplitudes of particular decays are calculated using FORM (software for analytical calculation). But the key problem is to calculate integrals of functions of multiple variables.

In the framework of this problem there is a particular iterated integral:

$$\int_0^1 d\alpha\{\alpha^a(1-\alpha)^b\} \int_0^\infty dt\{\frac{t^m}{(1+t)^n} F[z_0]\} \equiv I(a,b,m,n;F[z_0]), \quad (15)$$

$$F[z_0] = \exp[-2z_0],$$

$$z_0 = tD + \frac{t}{1+t}R^2,$$

$$D = \alpha_1(b_1^2 P^2 + m_1^2) + \alpha_2(b_2^2 P^2 + m_2^2),$$

$$R^2 = (\alpha_1^2 b_1^2 + \alpha_2^2 b_2^2 + 2\alpha_1\alpha_2 b_1 b_2)P^2,$$

$$b_1 = -\frac{m_1}{m_1+m_2}, b_2 = \frac{m_2}{m_1+m_2},$$

$$\alpha_1 = \alpha, \alpha_2 = 1-\alpha.$$

This integral is used in multiple calculations in the problem and its value describes the form of the mesons. As it can be seen in (15) it is denoted as $I(a,b,m,n;F[z_0])$, and with different combination of parameters it is used multiple times in the calculations.

5. Application of Neural Network Approach

The iterated integral (15) was firstly interpreted as a double integral:

$$\int_0^1 \alpha^a(1-\alpha)^b d\alpha \int_0^\infty \frac{t^m}{(1+t)^n} F[z_0]dt = \int_0^1 \int_0^\infty dt d\alpha \frac{t^m}{(1+t)^n} F[z_0]\alpha^a(1-\alpha)^b, \quad (16)$$

to fit the (1). Than the integrand from (16) was used to train the neural network models to approximate it with different parameters a, b, m, n. These parameters define the shape of the function, as well as the size of the volume under the surface defined by them. 8 values were calculated using neural networks approach and than compared to values calculated previously using *qdag* function of FORTRAN programming language. The resulting absolute errors of neural network approach integral values relative to *qdag* results are shown on Figure 3.

It can be observed, that absolute error reduces as the numerical integral values decrease which is mostly connected to the scaling, required to restore the correct value of the integral after the integrand's dataset was scaled to normalize data for neural network model training. The process of scaling the data and restoring the numerical integral value should have further improvement.

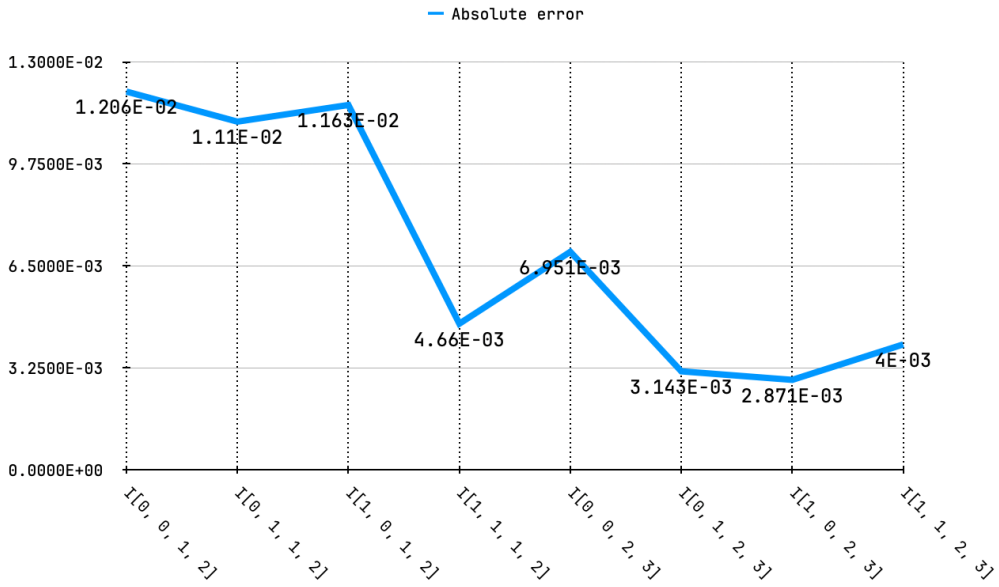


Figure 3: Absolute errors between numerical integral values calculated using neural network approach and FORTRAN function *qdag*

6. Conclusion

The neural network approach to numerical integration is being developed for the physics task of building a model for the unified description of both light and heavy mesons. Currently, the approach is capable of numerical integration of 2-dimensional functions.

The approach was tested using Alan Genz's package for testing the numerical integration. Than it was applied to iterated integral from the physics task.

Further development of the approach will include implementation of functionality to integrate functions of higher dimensions and optimization of the approach to produce higher accuracy of integration.

7. Acknowledgments

Authors of this work express their acknowledgments to Sara Shadmehri and Tatevik Bezhanyan from Meshcheryakov Laboratory of Information Technologies, JINR for their advices both in the field of neural networks and English language.

References

[1] S. Lloyd, R. A. Irani, M. Ahmadi, Using neural networks for fast numerical integration and optimization, IEEE Access 8 (2020) 84519–84531.

[2] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (1989) 303–314.

[3] A. Genz, Testing multidimensional integration routines, in: Proc. of international conference on Tools, methods and languages for scientific and engineering computation, 1984, pp. 81–94.

[4] S. Schmidt, D. Blaschke, Y. L. Kalinovsky, Scalar-pseudoscalar meson masses in nonlocal effective qcd at finite temperature, Physical Review C 50 (1994) 435.

[5] P. Costa, M. C. Ruivo, Y. L. Kalinovsky, Pseudoscalar neutral mesons in hot and dense matter, Physics Letters B 560 (2003) 171–177.