



Practical PINN Implementation for a Fractional-Order Damped Oscillator with CppAD-Computed Gradients [†]

Marina Shitikova ¹, Konstantin Modestov ²  and Yaroslav Tsvira ^{3,*} 

¹ Department of Higher Mathematics, Institute of Digital Technologies and Modeling in Construction, Moscow State University of Civil Engineering, 129337 Moscow, Russia; shitikovamv@mgsu.ru

² Department of General and Applied Physics, Institute of Digital Technologies and Modeling in Construction, Moscow State University of Civil Engineering, 129337 Moscow, Russia; modestovka@mgsu.ru

³ Department of Information Science and Applied Mathematics, Institute of Digital Technologies and Modeling in Construction, Moscow State University of Civil Engineering, 129337 Moscow, Russia

* Correspondence: tsvira.yaroslav@mail.ru

[†] Presented at the 1st International Online Conference on Fractal and Fractional, 13–15 April 2026; Available online: <https://sciforum.net/event/IOCF2026>.

Abstract

This work presents a practical C++23 implementation of a physics-informed neural network (PINN) for a fractional-order damped oscillator. A fully connected network outputs displacement and velocity, so the governing dynamics are enforced through a compact state-space residual involving first and second time derivatives. Integer-order derivatives are obtained via automatic differentiation, which removes finite-difference noise and preserves smooth, consistent gradients during training. The history-dependent fractional damping term is incorporated using the classical L1 discretization on a uniform time grid, which makes each residual evaluation depend on the entire predicted solution history and naturally captures memory effects. The training objective combines the squared residual norms at collocation points with a strongly weighted initial-condition penalty to control drift and stabilize early iterations. Gradients of the complete objective with respect to all network parameters are computed using reverse-mode automatic differentiation in CppAD (20260000.0) by constructing a scalar loss function of a flat parameter vector, enabling efficient gradient-based optimization. Parameters are updated with the Adam algorithm using bias correction and double-precision moment accumulation for numerical robustness. This implementation includes deterministic parameter packing, explicit size checks, and lightweight diagnostics of boundary values during training, improving reproducibility and debuggability. Overall, the code provides an end-to-end baseline for PINN-based simulation of fractional-order oscillatory systems and can be readily extended to include external forcing, alternative loss weight schedules, and parameter identification from measurement data.

Keywords: PINN; fractional differential; equations; optimization



Academic Editor: Haci
Mehmet Baskonus

Published: 23 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The idea of using neural networks as a class of approximating functions for solving differential equations arose significantly earlier than the modern term physics-informed neural networks (PINNs) [1–3]. Even in works related to multiquadrics and radial basis functions (RBFs), the solution to a differential equation was considered not only as the result of grid discretization but also as a functional approximation, the parameters of which

are selected based on the equation's residuals, initial and boundary conditions. One of the early directions in the field was the work of R. Hardy on multiquadrics and E. Kansa on the application of MQ/RBF approximations to computational fluid dynamics problems [4–7].

Further development of this approach involved a transition from pure collocation schemes to formulations, wherein differential equations, initial and boundary conditions are incorporated into a single error functional. This approach allows one to consider the problem of solving a differential equation as a problem of optimizing the parameters of a selected class of approximating functions. In this context, the work by E. Galperin, E. Kansa, and co-authors on global optimization of residual functionals is important, as well as the early neural network methods for solving ODEs and PDEs proposed by M. Dissanayake, N. Phan-Thien, I. Lagaris, A. Likas, D. Fotiadis, N. Mai-Duy, and T. Tran-Cong [8–13].

The modern formulation of PINN was systematized in the works of M. Raissi, P. Perdikaris, and G. E. Karniadakis, where a neural network is used as a global approximation of the solution, and physical information is incorporated into the loss function through the residual of the differential equation, initial and boundary conditions [14–16]. Unlike classical grid methods, PINN does not require the construction of a grid in the traditional sense, and training is performed using a set of collocation points. Automatic differentiation allows for the calculation of derivatives of the neural network output with respect to the input variables, making the method convenient for constructing residuals of differential equations [17–20].

The application of PINN to equations with fractional derivatives plays a special role in this work. Models with fractional derivatives naturally arise when describing systems with hereditary elastic properties. Y. A. Rossikhin, M. V. Shitikova, M. Caputo, I. Podlubny, F. Mainardi, and other authors have shown that fractional operators allow the description of dissipative processes, in which the current state of the system depends on the history of motion [21–32].

2. Materials and Methods

2.1. Mathematical Model

Let us consider the free vibrations of a one-degree-of-freedom mechanical system consisting of a mass, an elastic element, and a fractional damper. When the mass deviates from its equilibrium position, the spring generates a restoring force proportional to the displacement $x(t)$. The fractional damper describes the resistance to motion considering the inherent properties of the system, so the damping term is given not by the usual first derivative but by the fractional derivative $D_t^\nu x(t)$ (Figure 1).

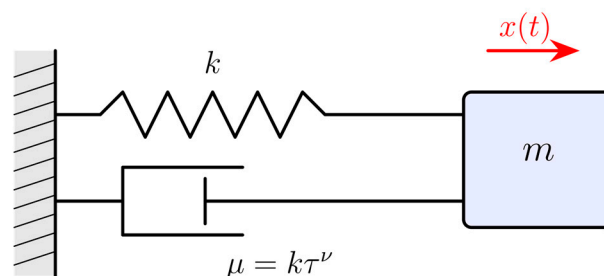


Figure 1. Mechanical model of a free fractionally damped oscillator.

Then the equation of the system's vibration could be written as

$$mx''(t) + \mu D_t^\nu x(t) + kx(t) = 0, \quad (1)$$

where D_t^ν is the fractional derivative with respect to time t of the order $\nu \in (0, 1)$ in the Gerasimov–Caputo sense, k is the spring's rigidity coefficient, m is the mass of the system, $\mu = k\tau^\nu$ is the viscous coefficient, and τ^ν is the relaxation time.

Dividing Equation (1) by the mass yields:

$$x''(t) + \zeta D_t^\nu x(t) + \omega_0^2 x(t) = 0, \quad (2)$$

where $\zeta = \frac{k\tau^\nu}{m}$ is the fractional damping coefficient, and $\omega_0^2 = \frac{k}{m}$ is the natural frequency of vibrations.

Equation (2) is subjected to the following initial conditions:

$$x(0) = 1, \quad x'(0) = 0. \quad (3)$$

For numerical modeling, Equation (1) is reduced to the following system:

$$\begin{cases} y_2(t) = y_1'(t), \\ y_2'(t) + \zeta D_t^\nu y_1(t) + \omega_0^2 y_1(t) = 0, \end{cases} \quad (4)$$

where

$$y_1(t) = x(t), \quad y_2(t) = x'(t). \quad (5)$$

2.2. Neural Network Architecture

To solve the problem, a multilayer perceptron (MLP) is used. This architecture was chosen because the problem statement requires approximating an unknown scalar function of a single variable. Therefore, the problem is reduced to mapping:

$$x \mapsto \hat{u}(x; \theta), \quad (6)$$

where x is the independent variable, $\hat{u}$ is the approximation of the desired solution, and θ is the set of network weights.

The network consists of an input layer, several hidden fully connected layers, and an output layer. Each neuron in the hidden layer computes a linear combination followed by a nonlinear activation function. Let $h_0 = x$. Then, we represent the computations in the multilayer perceptron as a sequence of linear and nonlinear transformations:

$$\begin{aligned} t_l &= h_{l-1}W_l + b_l, \quad l = 1, 2, \dots, L, \\ h_l &= F(t_l), \quad l = 1, 2, \dots, L-1, \\ \hat{u}(x; \theta) &= t_L = h_{L-1}W_L + b_L, \end{aligned} \quad (7)$$

where W_l is the weight matrix of the l -th layer, b_l is the bias vector of the l -th layer, F is the activation function, and $\theta = \{W_1, b_1, \dots, W_L, b_L\}$ is the full set of trainable parameters of the neural network.

Let us look at an example of one fully connected layer of a neural network (Figure 2).

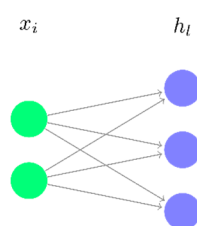


Figure 2. An example of one fully connected layer of a neural network.

Let the vectors x, h, b be given as

$$\begin{aligned} x &= (x_1, x_2), \\ h &= (h_1, h_2, h_3), \\ b &= (b_1, b_2, b_3). \end{aligned} \quad (8)$$

Then the vector of output values $h = F(xW + b)$ is calculated as the nonlinear activation function F of the expression $xW + b$, where

$$W = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} \quad (9)$$

To calculate the first element of the vector h , we need:

$$\begin{aligned} h_1 &= F(x_1 \cdot w_{11} + x_2 \cdot w_{21} + b_1), \\ h_2 &= F(x_1 \cdot w_{12} + x_2 \cdot w_{22} + b_2), \\ h_3 &= F(x_1 \cdot w_{13} + x_2 \cdot w_{23} + b_3). \end{aligned} \quad (10)$$

Let us split the calculations of the h_l layer into two stages: the linear part t_l (without using the activation function F) and the nonlinear part h_l (Figure 3).

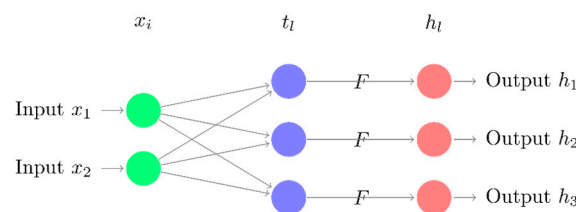


Figure 3. Fully connected neural network layer with separation of linearity and nonlinearity.

Then $t = xW + b$ is the linear part and $h = F(t)$ is the nonlinear part.

In physically informed neural network problems, an important requirement is the ability to calculate the derivatives of the desired function with respect to the input variable. Therefore, it is advisable to use smooth activation functions in hidden layers; in our case, we will use the hyperbolic tangent function denoted from hereafter as $\tanh$.

$$F(t_l) = \tanh(t) = \frac{e^{t_l} - e^{-t_l}}{e^{t_l} + e^{-t_l}}. \quad (11)$$

Taking the above facts into account, we obtain a topological model shown in Figure 4.

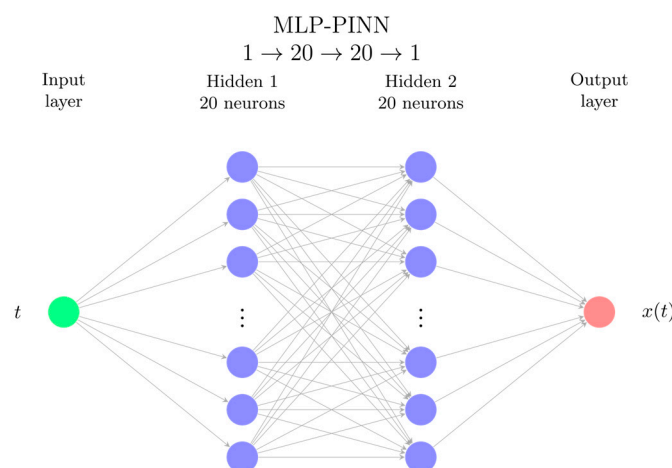


Figure 4. Topological model of a neural network (multilayer perceptron).

2.3. Physics-Informed Loss Function and Gradient Optimizer

Within the grid t_i , 8 collocation points have been considered.

Let us compose our residual functions:

$$\begin{aligned} r_1(t_i) &= y_1'(t_i) - y_2(t_i), \\ r_2(t_i) &= y_2'(t_i) + \zeta D_1^y y_1(t_i) + \omega_0^2 y_1(t_i). \end{aligned} \quad (12)$$

Having determined the residual, we need to compose the error function:

$$L(\theta) = L_{phys}(\theta) + \lambda_{ic} L_{ic}(\theta), \quad (13)$$

where λ_{ic} is a coefficient for adjusting the priority of initial conditions.

Let us examine the error function expression in more detail. Then the physical part of the losses:

$$L_{phys}(\theta) = \sum_{i=1}^N (r_1^2(t_i) + r_2^2(t_i)), \quad (14)$$

where t_i are collocation points, and $r_1^2(t_i)$ and $r_2^2(t_i)$ are the squares of the residual, in order that the errors do not cancel each other in signs.

Initial conditions should be added by a separate penalty:

$$L_{ic}(\theta) = (y_1(0) - 1)^2 + (y_2(0) - 0)^2, \quad (15)$$

in order that the neural network results in the tending of $L_{ic}(\theta)$ to 0.

After compiling the error function, we need to compose the optimizer itself and to solve the optimization problem $\min_{\theta} \mathcal{L}(\theta)$. We will use the adaptive Adam method [33], as it is best suited for solving physics problems. The calculation at each step is

$$g_t = \nabla_{\theta} \mathcal{L}(\theta_t). \quad (16)$$

Next, the exponential means are updated. Let us consider the first moment:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad (17)$$

where m_t is a smoothed gradient, and $\beta_1 \in (0, 1)$ specifies the “memory” of past gradients.

Then the second moment:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) (g_t \odot g_t), \quad (18)$$

where v_t is a smoothed gradient estimate, and $\beta_2 \in (0, 1)$ specifies the “memory” of past gradients.

Initialization of values at the first step is the following:

$$m_0 = 0, \quad v_0 = 0. \quad (19)$$

Since conditions (19) require the equality to zero, then estimates ((17) and (18)) are systematically underestimated at the early steps. Therefore, corrected estimates (bias correction) are introduced as

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \widehat{v}_t = \frac{v_t}{1 - \beta_2^t}. \quad (20)$$

Adam’s final update considering (20) results in the following relationship:

$$\theta_{t+1} = \theta_t - \eta \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t + \epsilon}}. \quad (21)$$

For neutral network learning the following hyperparameters are adopted:

$$\eta = 5 \cdot 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}, epochs = 80000, \lambda_{ic} = 10000. \quad (22)$$

The derivatives of the neural network output are calculated using automatic differentiation. It is worth noting that the fractional derivative could not be obtained using automatic differentiation. It is approximated by an L1 scheme on a uniform time grid. The gradient of the overall loss function over all trainable parameters is then calculated using inverse automatic differentiation in CppAD [34].

The hyperparameters (22) have been selected empirically to ensure that convergence is stable. The penalty weight λ_{ic} in (22) has been taken in such a way as to enforce the initial conditions, in order to minimize initial errors during the calculation of the fractional derivative.

3. Results

In this work, a PINN has been developed to solve the mechanical model of a free fractionally damped oscillator. The numerical results of the solution of Equation (2) according to the approach discussed above are compared with those obtained in [35] using the inverse Laplace transform (ILT) and Caputo L1 (L1) numerical scheme, and are illustrated in Table 1 and Figure 5.

Table 1. Numerical results.

No	t	PINN	ILT	L1	$\delta, \%$
1	0.0	0.999985	1.000000	1.000000	0.001502
2	2.0	0.123111	0.124178	0.124175	0.866901
3	4.0	0.192444	0.191305	0.191312	0.591452
4	6.0	0.183427	0.185167	0.185162	0.948685
5	8.0	0.113223	0.114302	0.114305	0.952644
6	10.0	0.139373	0.127773	0.127773	8.322746
7	12.0	0.114579	0.102692	0.102692	10.373831
8	14.0	0.100518	0.097514	0.097514	2.989291
9	16.0	0.090006	0.089410	0.089410	0.661748
10	18.0	0.083058	0.082952	0.082952	0.127838
11	20.0	0.078154	0.078136	0.078136	0.022814
12	22.0	0.073534	0.073530	0.073531	0.003917
13	24.0	0.069800	0.069800	0.069800	0.000645

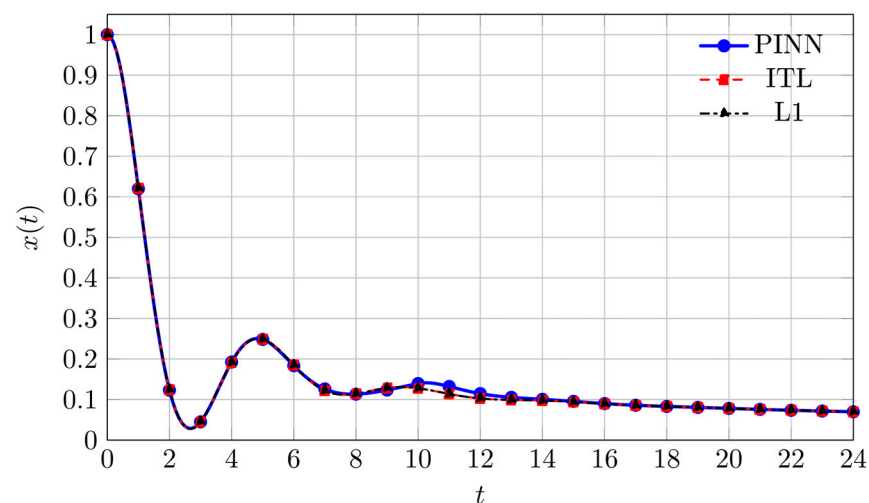


Figure 5. PINN solution graph.

4. Discussion

The obtained results show good agreement between the PINN solution and the reference solution obtained by the inverse Laplace transform. This indicates that the proposed loss function correctly enforces the governing equation and the initial conditions.

In the implementation, integer-order derivatives with respect to time are computed by automatic differentiation, while the fractional derivative is approximated by the L1 scheme. The gradient of the total loss with respect to the neural network parameters is computed by reverse-mode automatic differentiation in CppAD and then used in the Adam optimizer.

The increased relative errors at $t = 10.0$ and $t = 12.0$ are mainly associated with the long-time behavior of the system. Since the reference solution has a smaller magnitude at these points, moderate absolute deviations lead to larger relative errors. Moreover, the L1 approximation of the fractional derivative is history-dependent, so approximation errors may accumulate over time. This effect can be reduced by increasing the number of collocation points, refining the time step, and extending the training process.

The present study is limited to a fractionally damped oscillator. Further work may include forced oscillations, parameter identification, and comparison with other numerical schemes.

5. Conclusions

A C++ implementation of a physics-informed neural network for a fractionally damped oscillator has been presented. The fractional damping term has been evaluated using the L1 scheme, while CppAD has been used to compute gradients of the complete loss function with respect to all trainable parameters.

The comparison with the inverse Laplace transform demonstrates that the proposed PINN could approximate the solution with sufficient accuracy. The developed implementation could be used as a baseline for further studies of fractional-order oscillatory systems.

Author Contributions: Conceptualization, M.S., K.M. and Y.T.; methodology, Y.T.; software, Y.T.; verification, Y.T.; investigation, Y.T.; writing—original draft preparation, Y.T.; writing—review and editing, M.S.; visualization, Y.T.; supervision, K.M.; project administration, M.S.; funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been supported by the Russian Science Foundation, Project No. 25-11-00140.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.* **1989**, *2*, 303–314. [[CrossRef](#)]
2. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [[CrossRef](#)]
3. Park, J.; Sandberg, I.W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [[CrossRef](#)] [[PubMed](#)]
4. Hardy, R.L. Multiquadric equations of topography and other irregular surfaces. *J. Geophys. Res.* **1971**, *76*, 1905–1915. [[CrossRef](#)]
5. Hardy, R. Theory and applications of the multiquadric-biharmonic method 20 years of discovery 1968–1988. *Comput. Math. Appl.* **1990**, *19*, 163–208. [[CrossRef](#)]

6. Kansa, E. Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—I surface approximations and partial derivative estimates. *Comput. Math. Appl.* **1990**, *19*, 127–145. [CrossRef]
7. Kansa, E. Multiquadrics—A scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations. *Comput. Math. Appl.* **1990**, *19*, 147–161. [CrossRef]
8. Galperin, E.A.; Zheng, Q. Solution and control of PDE via global optimization methods. *Comput. Math. Appl.* **1993**, *25*, 103–118. [CrossRef]
9. Galperin, E.A.; Pan, Z.; Zheng, Q. Application of global optimization to implicit solution of partial differential equations. *Comput. Math. Appl.* **1993**, *25*, 119–124. [CrossRef]
10. Galperin, E.; Kansa, E. Application of global optimization and radial basis functions to numerical solutions of weakly singular volterra integral equations. *Comput. Math. Appl.* **2002**, *43*, 491–499. [CrossRef]
11. Dissanayake, M.W.M.G.; Phan-Thien, N. Neural-network-based approximations for solving partial differential equations. *Commun. Numer. Methods Eng.* **1994**, *10*, 195–201. [CrossRef]
12. Lagaris, I.; Likas, A.; Fotiadis, D. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **1998**, *9*, 987–1000. [CrossRef] [PubMed]
13. Mai-Duy, N.; Tran-Cong, T. Numerical solution of differential equations using multiquadric radial basis function networks. *Neural Netw.* **2001**, *14*, 185–199. [CrossRef] [PubMed]
14. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning. Part I: Data-Driven Solutions of Nonlinear Partial Differential Equations. *arXiv* **2017**. [CrossRef]
15. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics Informed Deep Learning. Part II: Data-Driven Discovery of Nonlinear Partial Differential Equations. *arXiv* **2017**. [CrossRef]
16. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
17. Wengert, R.E. A simple automatic derivative evaluation program. *Commun. ACM* **1964**, *7*, 463–464. [CrossRef]
18. Linnainmaa, S. Taylor expansion of the accumulated rounding error. *BIT Numer. Math.* **1976**, *16*, 146–160. [CrossRef]
19. Speelpenning, B. Compiling Fast Partial Derivatives of Functions Given by Algorithms. Ph.D. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1980. Available online: <https://www.osti.gov/servlets/purl/5254402> (accessed on 25 May 2026).
20. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic Differentiation in Machine Learning: A Survey. *J. Mach. Learn. Res.* **2018**, *18*, 1–43.
21. Shitikova, M.V. Fractional Operator Viscoelastic Models in Dynamic Problems of Mechanics of Solids: A Review. *Mech. Solids* **2022**, *57*, 1–33. [CrossRef]
22. Samko, S.G.; Kilbas, A.A.; Marichev, O.I. *Fractional Integrals and Derivatives: Theory and Applications*; Gordon and Breach: Yverdon, Switzerland, 1993.
23. Oldham, K.B.; Spanier, J. *The Fractional Calculus: Theory and Applications of Differentiation and Integration to Arbitrary Order*; Academic Press: New York, NY, USA, 1974.
24. Podlubny, I. *Fractional Differential Equations*; Academic Press: San Diego, CA, USA, 1999.
25. Gemant, A. A Method for Analyzing Experimental Results Obtained from Elasto-Viscous Bodies. *Physics* **1936**, *7*, 311–317. [CrossRef]
26. Blair, G.W.S. Analytical and Integrative Aspects of the Stress-Strain-Time Problem. *J. Sci. Instrum.* **1944**, *21*, 80–84. [CrossRef]
27. Gerasimov, A.N. A Generalization of Linear Laws of Deformation and Its Application to Problems of Internal Friction. *Prikl. Mat. Mekh.* **1948**, *12*, 251–260. (In Russian). Available online: <https://pmm.ipmnet.ru/ru/get/1948/12-3/251-260> (accessed on 18 June 2026).
28. Rabotnov, Y.N. Equilibrium of an elastic medium with after-effect. *Fract. Calc. Appl. Anal.* **2014**, *17*, 684–696. [CrossRef]
29. Shermergor, D.T. On the Use of Fractional Differentiation Operators for the Description of Elastic-Aftereffect Properties of Materials. *J. Appl. Mech. Tech. Phys.* **1966**, *7*, 85–87. [CrossRef]
30. Caputo, M. Linear Models of Dissipation whose Q is almost Frequency Independent—II. *Geophys. J. Int.* **1967**, *13*, 529–539. [CrossRef]
31. Caputo, M.; Mainardi, F. Linear models of dissipation in anelastic solids. *La Riv. del Nuovo Cimento* **1971**, *1*, 161–198. [CrossRef]
32. Rossikhin, Y.A.; Shitikova, M.V. Applications of Fractional Calculus to Dynamic Problems of Linear and Nonlinear Hereditary Mechanics of Solids. *Appl. Mech. Rev.* **1997**, *50*, 15–67. [CrossRef]
33. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**. [CrossRef]

34. Bell, B.M. CppAD: A Package for Differentiation of C++ Algorithms, Version 20260000.0. GitHub Repository. 2026. Available online: <https://github.com/coin-or/CppAD> (accessed on 25 May 2026).
35. Rossikhin, Y.A.; Shitikova, M.V. Application of fractional derivatives to the analysis of damped vibrations of viscoelastic single mass systems. *Acta Mech.* **1997**, *120*, 109–125. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.