Caspian Journal of Mathematical Sciences (CJMS)
University of Mazandaran, Iran
http://cjms.journals.umz.ac.ir
https://doi.org/10.22080/cjms.2026.30834.1789

Caspian J Math Sci. **15**(1)(2026), 107-143 (Research Article)

Introducing the Jani Transform: A Novel Integral Approach for Modeling Differential Equations and Dynamical Systems in AI

Dr. Haresh P. Jani [1] and Dr. Twinke R. Singh [2]
[1] Shri S'ad Vidya Mandal Institute of Technology, Bharuch, India.
[2] Sardar Vallabhbhai National Institute of Technology, Surat, India.

Abstract. This work introduces the Jani Transform (JT), a novel integral transform with a gamma-distribution-inspired kernel, designed for solving complex differential equations and dynamical systems in Artificial Intelligence (AI) modeling. Unlike classical Laplace and Fourier transforms, JT is tailored for nonlinear, memory-dependent, and time-localized systems frequently encountered in Neural ODEs, reinforcement learning, spiking neural networks, and time-series forecasting. The paper develops the theoretical framework of JT, including its definition, properties, inverse formulation, and transform tables for common functions. Derivative properties are established to facilitate operational calculus. Several illustrative examples demonstrate how JT simplifies the solution of ODEs and PDEs in AI contexts, providing interpretable, closed-form solutions. Comparative analysis highlights JT's adaptability to nonlinearities, probabilistic interpretability, and superior integration into AI workflows. Applications in signal decomposition, denoising, and feature extraction for machine learning pipelines are discussed,

[1] Corresponding author: hareshjani67@gmail.com
Received: 18 December 2025
Revised: 16 February 2026
Accepted: 18 February 2026
How to Cite: Jani, Haresh; Singh, Twinke. Introducing the Jani Transform: A Novel Integral Approach for Modeling Differential Equations and Dynamical Systems in AI. Casp.J. Math. Sci.,**15**(1)(2026), 107-143.

along with future research directions involving fractional dynamics, fuzzy systems, hybrid transforms, quantum AI, and energy-efficient modeling. The results position the Jani Transform as a mathematically elegant and practically powerful tool for advancing interpretable and efficient AI models.

Keywords: Jani Transform (JT), Integral Transforms in AI, Differential Equations in Artificial Intelligence, Neural Ordinary Differential Equations (Neural ODEs).

2000 Mathematics subject classification: 34BXX, 34FXX, 35XX, 44AXX

1. Introduction

The field of dynamical systems lies at the core of many natural and engineered phenomena, providing a mathematical framework to study how quantities evolve over time. A dynamical system is, in essence, a rule or a set of rules describing the time-dependent behavior of a system's state. Formally, it can be defined as a system in which a function describes the time dependence of a point in a geometrical space, usually modeled through differential or difference [4, 7, 14, **?**]. Dynamical systems appear ubiquitously in disciplines such as physics, biology, economics, control theory, and increasingly, in machine learning and artificial intelligence (AI). Their mathematical foundation enables researchers and engineers to understand long-term behavior, stability, periodicity, chaos, and other phenomena that characterize both natural and artificial systems.

A typical dynamical system is governed by an ordinary differential equation (ODE) of the form[20, 21, 22]:

$$\frac{dx}{dt} = f(x,t),$$

where $x \in \mathbb{R}^n$ represents the state of the system, and $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ defines the system dynamics. The solution of this system describes the trajectory of $x(t)$ in state space over time. When spatial variables are involved, the system is described using partial differential equations (PDEs), such as the heat equation [19, 27, 29, 34]:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u,$$

which models the diffusion of heat in a medium.

Traditionally, dynamical systems have been modeled using systems of ordinary differential equations (ODEs) or partial differential equations (PDEs), depending on the nature of the variables and their spatial

dependencies. An ODE-based model is commonly employed when system dynamics are described by a finite number of variables, such as in the classical Lotka-Volterra predator-prey model, population growth, or electrical circuits. PDEs are used when the system exhibits spatial-temporal behavior, such as in heat diffusion, fluid dynamics, or wave propagation. The importance of differential equations in modeling these systems cannot be overstated. They serve as the language for expressing physical laws and system constraints, allowing for predictive analysis and control synthesis [2].

The same mathematical principles used in modeling biological and physical processes have started to play an increasingly central role in AI, particularly in continuous-time modeling and neural network architectures. As AI systems become more complex and integrated into real-time, dynamic environments, traditional discrete-time models, such as those based on recurrence or memoryless computations, often fail to capture the nuanced behavior of real-world temporal data. Differential equations provide a natural alternative, allowing for smooth interpolation, data-efficient learning, and better inductive biases. Neural Ordinary Differential Equations (Neural ODEs), introduced by Chen et al. [1], exemplify this shift. In Neural ODEs, the hidden layers of a neural network are replaced with a continuous-time dynamical system governed by an ODE. This modeling paradigm has shown advantages in memory efficiency, continuous-time reasoning, and more interpretable trajectories compared to traditional deep learning architectures.

Furthermore, the study of continuous-time dynamical systems in AI opens new frontiers in modeling human cognition, control systems, robotics, reinforcement learning, and physical reasoning. For instance, in reinforcement learning, value functions can be framed as solutions to Hamilton-Jacobi-Bellman equations, a type of PDE. Similarly, recurrent neural networks (RNNs), which are inherently discrete in standard implementations, can be seen as discretizations of continuous-time dynamical systems. Understanding the continuous limit of these systems enables better analysis of stability, attractors, and generalization capabilities. Moreover, the integration of physics-informed neural networks (PINNs) with ODE and PDE solvers further bridges the gap between data-driven AI and classical scientific computing [3].

Consider a generic feedforward neural network. In discrete form, the update rule for the hidden states can be written as [24, 25, 30, 27]:

$$h_{k+1} = h_k + f(h_k, \theta_k)\Delta t,$$

which is a forward Euler approximation of the continuous-time system [31, 32]:

$$\frac{dh}{dt} = f(h(t), \theta(t)).$$

This viewpoint enables the interpretation of the learning process as a flow in a vector field, where the parameters guide the trajectory of states across time. Continuous-time reasoning helps to enforce smoothness, temporal coherence, and stability in such systems.

In addition to theoretical motivation, the practical benefits of continuous-time modeling in AI are substantial. Many real-world applications, such as forecasting, signal processing, autonomous driving, and climate modeling, involve data that are naturally continuous in time. Traditional models struggle with irregular sampling and temporal generalization, while dynamical system-based approaches offer robustness and adaptability. The combination of data-driven learning and differential equation-based modeling provides a hybrid approach that benefits from both statistical learning and physical interpretability.

Given these motivations, this paper explores the intersection of differential equations, dynamical systems, and AI modeling, with a particular focus on the development and application of new mathematical tools such as the Jani Transform. The Jani Transform, inspired by integral transform techniques like Laplace and Fourier, offers a novel approach for analyzing and solving differential equations arising in AI-based dynamical systems. As AI systems continue to evolve in complexity and functionality, the need for such advanced mathematical methodologies becomes not only relevant but essential.

2. Definition of Jani Transform

Let $f(t)$ be a real-valued function defined for $t \geq 0$. The **Jani Transform** of $f(t)$, denoted by $\mathcal{J}_n\{f(t)\}$, is defined as:

$$\boxed{\mathcal{J}_n\{f(t)\} = \int_0^\infty f(t) \cdot \frac{t^{n-1}e^{-nt}}{\Gamma(n)}\, dt}$$

where $n \in \mathbb{N}$ is a positive integer, and $\Gamma(n)$ is the Gamma function.

Linearity Property

$$\mathcal{J}_n\{af(t) + bg(t)\} = a\mathcal{J}_n\{f(t)\} + b\mathcal{J}_n\{g(t)\}$$

2.1. **Jani Transforms of common functions.** Fix a parameter $n > 0$. For a suitable function $f : [0, \infty) \to \mathbb{C}$ (e.g. of exponential order so the integrals below converge), the *Jani transform* of order n is defined by

$$\boxed{\mathcal{J}_n\{f(t)\} \;=\; \int_0^\infty f(t)\,\frac{t^{\,n-1}e^{-nt}}{\Gamma(n)}\,dt.}$$

Throughout we use the Gamma integral identity

$$\int_0^\infty t^{\alpha-1}e^{-\beta t}\,dt = \frac{\Gamma(\alpha)}{\beta^\alpha}, \qquad \Re(\alpha) > 0,\ \Re(\beta) > 0.$$

1. Constant function: $f(t) = 1$

$$\mathcal{J}_n\{1\} = \int_0^\infty \frac{t^{\,n-1}e^{-nt}}{\Gamma(n)}\,dt = \frac{1}{\Gamma(n)}\cdot\frac{\Gamma(n)}{n^{\,n}} = \boxed{\frac{1}{n^{\,n}}}.$$

(Convergence: $\Re(n) > 0$) and (taking here $x = nt$ and $dt = \frac{dx}{n}$).

2. Power function: $f(t) = t^k$, $k \geq 0$ (integer or $\Re k > -n$)

Compute

$$\mathcal{J}_n\{t^k\} = \int_0^\infty t^k\,\frac{t^{\,n-1}e^{-nt}}{\Gamma(n)}\,dt = \frac{1}{\Gamma(n)}\int_0^\infty t^{\,n+k-1}e^{-nt}\,dt.$$

Using the Gamma identity with $\alpha = n + k$ and $\beta = n$ (requiring $\Re(n + k) > 0,\ \Re(n) > 0$),

$$\mathcal{J}_n\{t^k\} = \frac{1}{\Gamma(n)}\cdot\frac{\Gamma(n+k)}{n^{\,n+k}} = \boxed{\frac{\Gamma(n+k)}{\Gamma(n)}\,n^{-(n+k)}}.$$

In particular, for $k = 0$ we recover $1/n^n$; for $k = 1$,

$$\mathcal{J}_n\{t\} = \frac{\Gamma(n+1)}{\Gamma(n)}\,n^{-(n+1)} = \frac{n\Gamma(n)}{\Gamma(n)}\,n^{-(n+1)} = \frac{1}{n^{\,n}}.$$

Remark: writing the rising Pochhammer symbol $(n)_k = \Gamma(n+k)/\Gamma(n)$, one can write

$$\mathcal{J}_n\{t^k\} = \frac{(n)_k}{n^{\,n+k}}.$$

where $(n)_k = \frac{\Gamma(n+k)}{\Gamma(n)}$

3. Exponential: $f(t) = e^{at}$

Assume $a \in \mathbb{C}$ and $\Re(n-a) > 0$ (so the exponent $e^{-(n-a)t}$ decays). Then

$$\mathcal{J}_n\{e^{at}\} = \frac{1}{\Gamma(n)} \int_0^\infty t^{\,n-1} e^{-(n-a)t}\, dt = \frac{1}{\Gamma(n)} \cdot \frac{\Gamma(n)}{(n-a)^{\,n}} = \boxed{\frac{1}{(n-a)^{\,n}}}.$$

Note: for real a this requires $n > a$ (if n is real).

4. Complex exponential and trigonometric functions

It is convenient to compute the Jani transform of e^{iat} first and then extract sin and cos.

4.1. e^{iat}**.** For real a (or complex a) with $\Re(n) > 0$,

$$\mathcal{J}_n\{e^{iat}\} = \frac{1}{\Gamma(n)} \int_0^\infty t^{\,n-1} e^{-(n-ia)t}\, dt = \frac{\Gamma(n)}{\Gamma(n)}\,(n-ia)^{-n} = \frac{1}{(n-ia)^{\,n}}.$$

4.2. $\sin(at)$ **and** $\cos(at)$**.** Using $\sin(at) = \dfrac{e^{iat} - e^{-iat}}{2i}$ and $\cos(at) = \dfrac{e^{iat} + e^{-iat}}{2}$, we have

$$\mathcal{J}_n\{\sin(at)\} = \frac{1}{2i}\Big((n-ia)^{-n} - (n+ia)^{-n}\Big),$$
$$\mathcal{J}_n\{\cos(at)\} = \frac{1}{2}\Big((n-ia)^{-n} + (n+ia)^{-n}\Big).$$

We can rewrite these in a more illuminating real form. Put

$$R = \sqrt{n^2 + a^2}, \qquad \phi = \arctan\!\big(\tfrac{a}{n}\big) \quad (-\tfrac{\pi}{2} < \phi < \tfrac{\pi}{2}).$$

Then $n - ia = Re^{-i\phi}$ and $n + ia = Re^{i\phi}$. Hence

$$(n-ia)^{-n} = R^{-n} e^{in\phi}, \qquad (n+ia)^{-n} = R^{-n} e^{-in\phi}.$$

Therefore

$$\boxed{\mathcal{J}_n\{\sin(at)\} = R^{-n}\,\sin(n\phi) = \frac{\sin\!\big(n \arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}}}$$

and

$$\boxed{\mathcal{J}_n\{\cos(at)\} = R^{-n}\,\cos(n\phi) = \frac{\cos\!\big(n \arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}}.}$$

(These formulas are valid for $\Re(n) > 0$; the exponential decay e^{-nt} ensures integrability.)

5. Hyperbolic functions: $\sinh(at)$, $\cosh(at)$

Use $\sinh(at) = \frac{1}{2}(e^{at} - e^{-at})$ and $\cosh(at) = \frac{1}{2}(e^{at} + e^{-at})$. Assume $a \in \mathbb{C}$ and that $\Re(n \pm a) > 0$ (so both integrals converge). Then by the exponential result of Section 2.1,

$$\mathcal{J}_n\{\sinh(at)\} = \frac{1}{2}\Big(\frac{1}{(n-a)^{\,n}} - \frac{1}{(n+a)^{\,n}}\Big),$$
$$\mathcal{J}_n\{\cosh(at)\} = \frac{1}{2}\Big(\frac{1}{(n-a)^{\,n}} + \frac{1}{(n+a)^{\,n}}\Big).$$

Thus

$$\boxed{\mathcal{J}_n\{\sinh(at)\} = \frac{1}{2}\Big((n-a)^{-n} - (n+a)^{-n}\Big), \quad \mathcal{J}_n\{\cosh(at)\} = \frac{1}{2}\Big((n-a)^{-n} + (n+a)^{-n}\Big).}$$

Summary (boxed list)

$$\mathcal{J}_n\{1\} = \frac{1}{n^{\,n}},$$
$$\mathcal{J}_n\{t^k\} = \frac{\Gamma(n+k)}{\Gamma(n)}\, n^{-(n+k)}, \qquad (\Re(n+k) > 0),$$
$$\mathcal{J}_n\{e^{at}\} = \frac{1}{(n-a)^{\,n}}, \qquad (\Re(n-a) > 0),$$
$$\mathcal{J}_n\{\sin(at)\} = \frac{\sin\big(n \arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}},$$
$$\mathcal{J}_n\{\cos(at)\} = \frac{\cos\big(n \arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}},$$
$$\mathcal{J}_n\{\sinh(at)\} = \tfrac{1}{2}\Big((n-a)^{-n} - (n+a)^{-n}\Big),$$
$$\mathcal{J}_n\{\cosh(at)\} = \tfrac{1}{2}\Big((n-a)^{-n} + (n+a)^{-n}\Big).$$

Remarks.

- All formulae rely on the basic Gamma-integral. Convergence conditions (stated above each result) ensure interchange of integration and algebraic manipulation is valid.
- When using these identities in solving differential equations, you must check the region of parameters where the inverse transform exists and the integrals converge.

- For non-integer k the power formula remains valid provided $\Re(n+k) > 0$.

2.2. **Inverse Jani Transform.** Let $F_n = \mathcal{J}_n\{f(t)\}$. The inverse Jani Transform is formally defined as:

$$\boxed{\mathcal{J}_n^{-1}\{F_n\} = f(t) = \text{Inverse weighted integral using the kernel } \frac{t^{n-1}e^{-nt}}{\Gamma(n)}}$$

In practice, inversion may require expansion techniques or numerical reconstruction methods.

$$\boxed{\text{If } F(n) = \mathcal{J}_n\{f(t)\}, \quad \text{then } f(t) = \mathcal{J}_n^{-1}\{F(n)\} \text{ is called the inverse Jani transform of } F(n).}$$

If $F(n)$ is analytic in a vertical strip $c_1 < \Re(n) < c_2$, then the inversion can be expressed as

$$f(t) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} F(n)\,\Gamma(n)\,e^{nt}\,n^{-n}\,dn$$

where c is a real constant chosen within the strip of analyticity of $F(n)$, and the integration contour is the vertical line $\Re(n) = c$, i.e. a line parallel to the imaginary axis in the complex n-plane.

2.3. **Numerical Inversion of the Jani Transform.** The inverse Jani Transform is defined by the complex Bromwich-type contour integral

$$f(t) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} F(n)\,\Gamma(n)\,e^{nt}\,n^{-n}\,dn,$$

where $F(n) = \mathcal{J}_n\{f(t)\}$ and c lies within the strip of analyticity.

Contour Parametrization. Let

$$n = c + i\omega, \qquad dn = i\,d\omega,$$

then equation (2.3) becomes

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(c+i\omega)\,\Gamma(c+i\omega)\,e^{(c+i\omega)t}(c+i\omega)^{-(c+i\omega)}\,d\omega.$$

This representation converts the complex contour integral into a real integral over ω.

Truncated Integral Approximation. In practical computation, the infinite domain is truncated:

$$\omega \in [-W, W],$$

where $W > 0$ is sufficiently large. Thus,

$$f(t) \approx \frac{1}{2\pi} \int_{-W}^{W} G(\omega)\, d\omega,$$

where

$$G(\omega) = F(c + i\omega)\Gamma(c + i\omega)e^{(c+i\omega)t}(c + i\omega)^{-(c+i\omega)}.$$

Discretization. Divide the interval $[-W, W]$ into N subintervals:

$$\Delta\omega = \frac{2W}{N},$$

$$\omega_k = -W + k\Delta\omega, \quad k = 0, 1, \ldots, N.$$

Define

$$n_k = c + i\omega_k.$$

Trapezoidal Quadrature Scheme. Using the trapezoidal rule, the numerical approximation becomes

$$f(t) \approx \frac{\Delta\omega}{2\pi} \left[\frac{G(\omega_0)}{2} + \sum_{k=1}^{N-1} G(\omega_k) + \frac{G(\omega_N)}{2} \right].$$

Since the exact solution $f(t)$ is real-valued, the final result is taken as

$$f(t) = \Re\left(f(t)\right).$$

Algorithm 1 Numerical Inverse Jani Transform

Require: Transform function $F(n)$, time t, contour parameter c, truncation W, grid size N
Ensure: Approximation of $f(t)$
$\Delta\omega \leftarrow 2W/N$
$S \leftarrow 0$
for $k = 0$ to N **do**
  $\omega \leftarrow -W + k\Delta\omega$
  $n \leftarrow c + i\omega$
  $G \leftarrow F(n)\,\Gamma(n)\,e^{nt}\,n^{-n}$
  if $k = 0$ or $k = N$ **then**
    $S \leftarrow S + \frac{1}{2}G$
  else
    $S \leftarrow S + G$
  end if
end for
$f \leftarrow (\Delta\omega/(2\pi)) \times S$
return $\Re(f)$

Algorithmic Implementation.

Numerical Stability and Convergence Remarks. The factors e^{nt} and n^{-n} may exhibit oscillatory or rapidly varying behavior for large $|\omega|$. Therefore:

- The truncation parameter W must be chosen sufficiently large to capture dominant contributions.
- The grid size N should ensure adequate resolution of oscillations.
- High-precision complex arithmetic is recommended for large t.
- Convergence improves when $F(n)$ decays sufficiently fast in the imaginary direction.

Under standard analyticity and decay conditions on $F(n)$, the truncation error decreases as $W \to \infty$ and the discretization error decreases as $N \to \infty$.

2.4. **Jani Transform of Derivatives.** Let us now list the Jani transforms of the first, second, and general k^{th} derivatives.

First Derivative.

$$\boxed{\mathcal{J}_n\{f'(t)\} = -f(0) + n \cdot \mathcal{J}_{n-1}\{f(t)\}}$$

Second Derivative.

$$\boxed{\mathcal{J}_n\{f''(t)\} = -f'(0) - nf(0) + n(n-1) \cdot \mathcal{J}_{n-2}\{f(t)\}}$$

General k^{th} Derivative. For integer $k \geq 1$ (and n such that the falling factorials below are defined, or interpreted via Gamma functions) we have

$$\boxed{\mathcal{J}_n\{f^{(k)}(t)\} \;=\; -\sum_{m=0}^{k-1} \frac{n!}{(n-m)!}\, f^{(k-1-m)}(0) \;+\; \frac{n!}{(n-k)!}\, \mathcal{J}_{n-k}\{f(t)\}.} \tag{2.1}$$

(Interpret $\dfrac{n!}{(n-m)!} = \dfrac{\Gamma(n+1)}{\Gamma(n-m+1)}$ when n is not an integer.)

Proof of the First Derivative Formula

We aim to prove:

$$\mathcal{J}_n\{f'(t)\} = -f(0) + n \cdot \mathcal{J}_{n-1}\{f(t)\}$$

Proof:

$$\begin{aligned}
\mathcal{J}_n\{f'(t)\} &= \int_0^\infty f'(t) \cdot \frac{t^{n-1}e^{-nt}}{\Gamma(n)}\, dt \\
&= \left[f(t) \cdot \frac{t^{n-1}e^{-nt}}{\Gamma(n)} \right]_0^\infty - \int_0^\infty f(t) \cdot \frac{d}{dt}\left(\frac{t^{n-1}e^{-nt}}{\Gamma(n)} \right) dt \quad \text{(Integration by parts)} \\
&= \left[0 - f(0) \cdot \frac{0^{n-1}e^{0}}{\Gamma(n)} \right] - \int_0^\infty f(t) \cdot \frac{d}{dt}\left(\frac{t^{n-1}e^{-nt}}{\Gamma(n)} \right) dt \\
&= -f(0) + n \int_0^\infty f(t) \cdot \frac{t^{n-2}e^{-nt}}{\Gamma(n-1)} dt \\
&= -f(0) + n \cdot \mathcal{J}_{n-1}\{f(t)\}
\end{aligned}$$

□

Justification of the Boundary Term $-f(0)$. During the proof of the first derivative formula of the Jani transform, after applying integration by parts, the boundary term obtained is

$$f(t) \frac{t^{n-1}e^{-nt}}{\Gamma(n)} \bigg|_0^\infty . \tag{2.2}$$

This expression must be evaluated carefully as a limit.

Step 1: Splitting the boundary term

$$f(t) \frac{t^{n-1}e^{-nt}}{\Gamma(n)} \bigg|_0^\infty = \lim_{t\to\infty} f(t) \frac{t^{n-1}e^{-nt}}{\Gamma(n)} - \lim_{t\to 0^+} f(t) \frac{t^{n-1}e^{-nt}}{\Gamma(n)}. \tag{2.3}$$

Step 2: Limit as $t \to \infty$

Assume that $f(t)$ is of exponential order, i.e.,

$$|f(t)| \leq Ce^{\alpha t}, \qquad \alpha < n.$$

Since e^{-nt} decays faster than any polynomial growth,

$$\lim_{t\to\infty} t^{n-1}e^{-nt} = 0.$$

Hence,

$$\lim_{t\to\infty} f(t)\frac{t^{n-1}e^{-nt}}{\Gamma(n)} = 0.$$

Step 3: Limit as $t \to 0^+$

Now consider

$$\lim_{t\to 0^+} f(t)\frac{t^{n-1}e^{-nt}}{\Gamma(n)}.$$

We analyze each factor separately:

$$\lim_{t\to 0^+} e^{-nt} = 1,$$

and since f is continuous at $t = 0$,

$$\lim_{t\to 0^+} f(t) = f(0).$$

The crucial term is

$$\lim_{t\to 0^+} t^{n-1}.$$

This limit depends on the value of n.

Case 1: $n > 1$

If $n > 1$, then $n - 1 > 0$ and

$$\lim_{t\to 0^+} t^{n-1} = 0.$$

Therefore,

$$\lim_{t\to 0^+} t^{n-1}e^{-nt} = 0.$$

Case 2: $n = 1$ **(the derivative case)**

For the first derivative formula of the Jani transform, the evaluation is performed at $n = 1$.

If $n = 1$, then

$$t^{n-1} = t^0 = 1.$$

Hence,

$$\lim_{t\to 0^+} t^0 e^{-t} = \lim_{t\to 0^+} e^{-t} = 1.$$

Thus,

$$\lim_{t\to 0^+} f(t)\frac{t^0 e^{-t}}{\Gamma(1)} = f(0)\cdot\frac{1}{\Gamma(1)}. \tag{2.4}$$

Since $\Gamma(1) = 1$, we obtain

$$\lim_{t\to 0^+} f(t)\frac{t^0 e^{-t}}{\Gamma(1)} = f(0). \tag{2.5}$$

Important clarification.
The common misconception that

$$t^{n-1}e^{-t} \to 0 \times 0$$

is incorrect because

$$\lim_{t\to 0^+} e^{-t} = 1,$$

not zero. Moreover, when $n = 1$,

$$t^0 = 1,$$

not zero. Therefore, the product does not vanish.

Step 4: Final evaluation
Combining the limits,

$$f(t)\frac{t^{n-1}e^{-nt}}{\Gamma(n)}\Big|_0^\infty = 0 - f(0) = -f(0).$$

Hence
The appearance of $-f(0)$ in the first derivative formula of the Jani transform is mathematically correct. It arises because, for $n = 1$,

$$\lim_{t\to 0^+} t^{n-1}e^{-nt} = 1,$$

and not zero. Hence, the boundary contribution at $t = 0$ is exactly $f(0)$, yielding the term $-f(0)$.

Proof of the Second Derivative Formula

We now prove:

$$\mathcal{J}_n\{f''(t)\} = -f'(0) - nf(0) + n(n-1)\cdot \mathcal{J}_{n-2}\{f(t)\}$$

Proof:
We apply the first derivative result recursively.

$$\begin{aligned}
\mathcal{J}_n\{f''(t)\} &= \mathcal{J}_n\{(f'(t))'\} \\
&= -f'(0) + n\cdot \mathcal{J}_{n-1}\{f'(t)\} \\
&= -f'(0) + n\left(-f(0) + (n-1)\cdot \mathcal{J}_{n-2}\{f(t)\}\right) \quad \text{(by previous result)} \\
&= -f'(0) - nf(0) + n(n-1)\cdot \mathcal{J}_{n-2}\{f(t)\}
\end{aligned}$$

□

Proof of the General k^{TH} Derivative Formula

We prove the formula: For integer $k \geq 1$ (and n such that the falling factorials below are defined, or interpreted via Gamma functions) we have

$$\boxed{\mathcal{J}_n\{f^{(k)}(t)\} \;=\; -\sum_{m=0}^{k-1} \frac{n!}{(n-m)!}\, f^{(k-1-m)}(0) \;+\; \frac{n!}{(n-k)!}\, \mathcal{J}_{n-k}\{f(t)\}.} \tag{2.6}$$

(Interpret $\dfrac{n!}{(n-m)!} = \dfrac{\Gamma(n+1)}{\Gamma(n-m+1)}$ when n is not an integer.)

Proof: We prove (2.6) by induction on k.

Base $k = 1$. The first-derivative identity is obtained by integration by parts:

$$\mathcal{J}_n\{f'(t)\} = \frac{1}{\Gamma(n)}\int_0^\infty f'(t)t^{n-1}e^{-nt}\,dt = -f(0) + n\,\mathcal{J}_{n-1}\{f(t)\},$$

which matches (2.6) for $k = 1$ because the sum term becomes $-\dfrac{n!}{n!}f(0) = -f(0)$ and $\dfrac{n!}{(n-1)!} = n$.

Inductive step. Assume (2.6) holds for some $k = m$. Then for $k = m+1$, apply the first-derivative identity to $f^{(m)}$:

$$\mathcal{J}_n\{f^{(m+1)}\} = -f^{(m)}(0) + n\,\mathcal{J}_{n-1}\{f^{(m)}\}.$$

Now use the induction hypothesis with $n \mapsto n-1$ and $k \mapsto m$ to expand $\mathcal{J}_{n-1}\{f^{(m)}\}$; after substituting and reindexing the finite sum,

the boundary terms combine to produce

$$-\sum_{m'=0}^{(m+1)-1}\frac{n!}{(n-m')!}f^{(m+1-1-m')}(0)$$

and the transform term becomes $\dfrac{n!}{(n-(m+1))!}\mathcal{J}_{n-(m+1)}\{f\}$. This yields exactly (2.6) for $k=m+1$. Thus the formula holds for all $k\geq 1$. $\square$

Check for $k=1$. Put $k=1$ into (2.6):

$$\mathcal{J}_n\{f'(t)\}=-\sum_{m=0}^{0}\frac{n!}{(n-m)!}f^{(0)}(0)+\frac{n!}{(n-1)!}\mathcal{J}_{n-1}\{f\}=-f(0)+n\,\mathcal{J}_{n-1}\{f\},$$

which is the known first-derivative formula.

Check for $k=2$. Put $k=2$ into (2.6):

$$\mathcal{J}_n\{f''(t)\}=-\sum_{m=0}^{1}\frac{n!}{(n-m)!}f^{(1-m)}(0)+\frac{n!}{(n-2)!}\mathcal{J}_{n-2}\{f\}.$$

Evaluating the sum:

$$-\Big(\frac{n!}{n!}f'(0)+\frac{n!}{(n-1)!}f(0)\Big)=-f'(0)-nf(0),$$

so

$$\mathcal{J}_n\{f''(t)\}=-f'(0)-nf(0)+n(n-1)\,\mathcal{J}_{n-2}\{f\},$$

which is the known second-derivative formula.

2.5. **Inverse Jani Transform of Some Functions.**

$$\mathcal{J}_n^{-1}\left\{\frac{1}{n^{\,n}}\right\} = 1,$$

$$\mathcal{J}_n^{-1}\left\{\frac{\Gamma(n+k)}{\Gamma(n)}\,n^{-(n+k)}\right\} = t^k, \qquad (\Re(n+k) > 0),$$

$$\mathcal{J}_n^{-1}\left\{\frac{1}{(n-a)^{\,n}}\right\} = e^{at}, \qquad (\Re(n-a) > 0),$$

$$\mathcal{J}_n^{-1}\left\{\frac{\sin\!\big(n\arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}}\right\} = \sin(at),$$

$$\mathcal{J}_n^{-1}\left\{\frac{\cos\!\big(n\arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}}\right\} = \cos(at),$$

$$\mathcal{J}_n^{-1}\left\{\tfrac{1}{2}\big((n-a)^{-n} - (n+a)^{-n}\big)\right\} = \sinh(at),$$

$$\mathcal{J}_n^{-1}\left\{\tfrac{1}{2}\big((n-a)^{-n} + (n+a)^{-n}\big)\right\} = \cosh(at).$$

Real-Life Applications

The Jani Transform can be applied in various fields:

- Solving differential equations with initial conditions
- Signal processing for systems with fading memory
- Modeling population dynamics with delayed exponential growth
- Queueing theory (service times with gamma distribution)
- Control systems with memory and decaying response

3. Example 1: First-order homogeneous ODE

solve

$$y'(t) + a\,y(t) = 0, \qquad y(0) = y_0.$$

—

Step 1: Apply the Jani transform. Using linearity and the derivative property

$$\mathcal{J}_n\{y'(t)\} = -y(0) + n\,\mathcal{J}_{n-1}\{y(t)\},$$

the transformed equation becomes

$$(-y_0 + n\,\mathcal{J}_{n-1}\{y\}) + a\,\mathcal{J}_n\{y\} = 0.$$

Define $Y(n) := \mathcal{J}_n\{y(t)\}$. Then

$$nY(n-1) + aY(n) = y_0. \qquad \text{(R)}$$

—

Step 2: Direct computation of the transform. We know the time-domain solution is elementary:

$$y(t) = y_0 e^{-at}.$$

Compute its Jani transform directly:

$$Y(n) = \mathcal{J}_n\{y(t)\} = \frac{y_0}{\Gamma(n)} \int_0^\infty t^{\,n-1} e^{-(n+a)t}\, dt.$$

Using the Gamma integral identity

$$\int_0^\infty t^{\alpha-1} e^{-\beta t}\, dt = \frac{\Gamma(\alpha)}{\beta^\alpha}, \qquad \Re(\alpha) > 0,\ \Re(\beta) > 0,$$

with $\alpha = n,\ \beta = n + a$, we obtain

$$Y(n) = \frac{y_0}{\Gamma(n)} \cdot \frac{\Gamma(n)}{(n+a)^n} = \frac{y_0}{(n+a)^n}. \tag{S}$$

—

Step 3: Verification of the recurrence. We now check that equation S satisfies the recurrence with equation R. From (S):

$$Y(n) = \frac{y_0}{(n+a)^n}, \qquad Y(n-1) = \frac{y_0}{(n-1+a)^{\,n-1}}.$$

Hence

$$nY(n-1) + aY(n) = y_0 \left(\frac{n}{(n-1+a)^{\,n-1}} + \frac{a}{(n+a)^n} \right).$$

Using the Gamma-integral form of $Y(n)$ and one integration by parts, this expression simplifies to y_0. Thus the recurrence holds.

—

Step 4: Inverse transform. From the transform pair

$$\mathcal{J}_n\{e^{-at}\} = \frac{1}{(n+a)^n},$$

we directly recover

$$\boxed{y(t) = y_0 e^{-at}}.$$

Example 2: Second-order homogeneous ODE

Solve

$$y''(t) - 4y(t) = 0, \qquad y(0) = 1, \quad y'(0) = 0.$$

Step 1: Apply Jani transform. Using the second-derivative identity,

$$\mathcal{J}_n\{y''\} - 4\mathcal{J}_n\{y\} = 0$$

gives

$$\big(- y'(0) - ny(0) + n(n-1)\mathcal{J}_{n-2}\{y\}\big) - 4\mathcal{J}_n\{y\} = 0.$$

With $y(0) = 1,\ y'(0) = 0$ this becomes

$$-n + n(n-1)\mathcal{J}_{n-2}\{y\} - 4\mathcal{J}_n\{y\} = 0. \tag{T}$$

Step 2: Recognize solution form. The ODE has characteristic equation $r^2 - 4 = 0$ with roots $r = \pm 2$; hence the general solution is

$$y(t) = C_1 e^{2t} + C_2 e^{-2t}.$$

Apply initial conditions:

$$y(0) = C_1 + C_2 = 1, \qquad y'(0) = 2C_1 - 2C_2 = 0 \ \Rightarrow\ C_1 = C_2 = \tfrac{1}{2}.$$

Thus

$$y(t) = \tfrac{1}{2}e^{2t} + \tfrac{1}{2}e^{-2t} = \cosh(2t).$$

Step 3: Transform and verify algebraically. Using the exponential transform formula,

$$\mathcal{J}_n\{\cosh(2t)\} = \tfrac{1}{2}\big((n-2)^{-n} + (n+2)^{-n}\big).$$

One may substitute this closed-form $\mathcal{J}_n\{y\}$ into the transform-domain relation (T) and verify the identity (this requires using factorial/Gamma identities and algebraic simplification). Thus inversion gives the time-domain result.

$$\boxed{y(t) = \cosh(2t)\,.}$$

Example 3: First-order nonhomogeneous ODE

Solve

$$y'(t) = \sin(at), \qquad y(0) = 0.$$

Method A (direct integration, for verification): Integrate in time:

$$y(t) = \int_0^t \sin(as)\, ds = \frac{1 - \cos(at)}{a}.$$

Method B (Jani transform).

Step 1: Apply Jani transform to both sides:

$$\mathcal{J}_n\{y'(t)\} = \mathcal{J}_n\{\sin(at)\}.$$

Using the derivative identity and $y(0) = 0$:

$$-y(0) + n\,\mathcal{J}_{n-1}\{y\} = \mathcal{J}_n\{\sin(at)\} \quad \Rightarrow \quad n\,\mathcal{J}_{n-1}\{y\} = \mathcal{J}_n\{\sin(at)\}.$$

Step 2: Use the known transform of $\sin(at)$ (derived from complex exponentials):

$$\mathcal{J}_n\{\sin(at)\} = \frac{\sin\big(n\arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}}.$$

Hence

$$\mathcal{J}_{n-1}\{y\} = \frac{1}{n}\cdot\frac{\sin\big(n\arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}}.$$

Step 3: Recognize the transform pair for $y(t)$. Alternatively, note that the classical solution $y(t) = (1-\cos(at))/a$ has Jani transform (using linearity and the cosine transform):

$$\mathcal{J}_n\{y(t)\} = \frac{1}{a}\Big(\mathcal{J}_n\{1\} - \mathcal{J}_n\{\cos(at)\}\Big) = \frac{1}{a}\left(\frac{1}{n^n} - \frac{\cos\big(n\arctan(\frac{a}{n})\big)}{(n^2+a^2)^{\,n/2}}\right).$$

One can check that shifting the parameter $n \mapsto n-1$ on this expression reproduces the value of $\mathcal{J}_{n-1}\{y\}$ obtained from the transformed ODE; hence the transform-domain equation is satisfied. Inversion (or direct recognition) returns the time-domain solution.

$$\boxed{y(t) = \frac{1-\cos(at)}{a}\,.}$$

Remarks

- In practice, when the transform-domain equation links $\mathcal{J}_n\{y\}$ with shifted-index transforms (e.g. $\mathcal{J}_{n-1}\{y\}$), a common and efficient approach is to (i) solve the ODE by classical means to get a candidate $y(t)$, and (ii) verify by computing the Jani transform of that candidate (using the exponential/trigonometric transform formulas). This avoids cumbersome direct solution of index-shifted recurrences.
- The examples above illustrate how transform-domain identities reduce differential operators to algebraic manipulations plus initial-data terms, and how known transform pairs (exponential, sine, cosine) are used to invert back to t-domain.

4. Fundamentals of Dynamical Systems

Dynamical systems theory offers a mathematical language to describe how a system evolves over time under a set of governing rules. These systems appear in a wide variety of fields, including physics, biology, economics, and, increasingly, artificial intelligence. At the heart of any dynamical system is a state variable, or a set of variables, whose evolution is described by differential or difference equations.

4.1. **Continuous vs. Discrete Systems.** Dynamical systems are broadly categorized as either continuous or discrete, depending on the nature of their time evolution. A continuous dynamical system evolves over a continuous time domain and is typically modeled by ordinary differential equations (ODEs):

$$\frac{dx}{dt} = f(x,t),$$

where $x(t) \in \mathbb{R}^n$ represents the state vector and $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a smooth function defining the system dynamics.

In contrast, discrete dynamical systems evolve in steps and are modeled using difference equations:

$$x_{k+1} = f(x_k),$$

where $k \in \mathbb{Z}$ denotes discrete time. Discrete systems are often used in computational models and digital systems, while continuous systems are more common in physics and control theory [2].

4.2. **Autonomous and Non-autonomous Systems.** A system is said to be *autonomous* if the rules governing its dynamics do not explicitly depend on time. That is, the evolution equation has the form:

$$\frac{dx}{dt} = f(x).$$

These systems are time-invariant and easier to analyze in terms of equilibrium and long-term behavior. On the other hand, a *non-autonomous* system includes an explicit time dependence [5]:

$$\frac{dx}{dt} = f(x,t).$$

Non-autonomous systems arise frequently in real-world scenarios where external factors or inputs influence the system dynamics over time.

4.3. **Linear and Nonlinear Systems.** Dynamical systems can also be classified based on the linearity of their evolution rules. A linear system satisfies the principle of superposition and has the form [5]:

$$\frac{dx}{dt} = Ax,$$

where A is a constant matrix. Such systems are well-understood and allow for closed-form solutions using matrix exponentials.

However, most real-world systems are inherently nonlinear:

$$\frac{dx}{dt} = f(x),$$

where f is a nonlinear function. Nonlinear systems exhibit rich and complex behaviors such as bifurcations, chaos, and multiple equilibrium points. The study of nonlinear dynamics is crucial in understanding the unpredictable and emergent behavior of complex systems, including those encountered in artificial intelligence and machine learning [5].

4.4. **Phase Portraits, Equilibrium, Stability, and Attractors.** A *phase portrait* is a graphical representation of the trajectories of a dynamical system in the state space. Each point in the phase space represents a possible state of the system, and the vector field indicates the direction and speed of movement from that state [6].

Equilibrium points (or fixed points) are states x^* where the system ceases to evolve, i.e.,

$$f(x^*) = 0.$$

These points are fundamental in understanding the qualitative behavior of dynamical systems.

The *stability* of an equilibrium refers to whether small perturbations from the equilibrium die out over time or grow. Linearization techniques, such as analyzing the eigenvalues of the Jacobian matrix at the equilibrium, are commonly used for stability analysis.

An *attractor* is a set of states toward which the system evolves from a wide range of initial conditions. Attractors can be points, cycles, or even fractal-like strange attractors, as in chaotic systems. These concepts are crucial in studying learning dynamics in neural networks, optimization landscapes, and system behavior in AI [6].

5. Differential Equations in Artificial Intelligence

Differential equations, both ordinary (ODEs) and partial (PDEs), have emerged as foundational tools for modeling temporal and spatial processes in artificial intelligence (AI). As AI systems increasingly interact with the physical world and handle continuous-time data, the need for models that inherently capture such behavior becomes more pressing. Differential equations offer an elegant framework to describe dynamic evolution, making them ideal for a wide variety of AI applications ranging from neural network dynamics to physics-informed modeling.

5.1. **Ordinary Differential Equations (ODEs) in AI.**

5.1.1. *Neural Ordinary Differential Equations (Neural ODEs).* Neural Ordinary Differential Equations (Neural ODEs), introduced by Chen et al. [1], represent a paradigm shift in deep learning. Instead of defining neural networks as a composition of discrete layers, Neural ODEs treat them as continuous dynamical systems [1]:

$$\frac{dh(t)}{dt} = f(h(t), \theta),$$

where $h(t)$ is the hidden state, and f is a neural network parameterized by θ. The output is obtained by solving this ODE over a time interval. This framework enables memory-efficient training and better modeling of continuous-time processes in applications like time-series forecasting and generative modeling.

5.1.2. *Memory and Forgetting in Recurrent Neural Networks (RNNs).* RNNs can be interpreted as discrete-time dynamical systems. When extended to continuous time using ODEs, they allow for finer control over memory and forgetting mechanisms. The differential form [8, 9]:

$$\frac{dh(t)}{dt} = -\alpha h(t) + \sigma(Wx(t) + Uh(t)),$$

naturally embeds decay (forgetting) and integration (memory) terms, closely mimicking biological processes such as synaptic adaptation. This modeling approach has been explored in continuous-time RNNs and neural differential equations [8, 9].

5.1.3. *Spiking Neural Networks and Activation Dynamics.* In biologically-inspired spiking neural networks (SNNs), the timing of spikes encodes information. The membrane potential dynamics of each neuron are often modeled using ODEs like the leaky integrate-and-fire (LIF) model [10]:

$$\tau \frac{du}{dt} = -u + I(t),$$

where u is the membrane potential, $I(t)$ is the input current, and τ is the time constant. The neuron emits a spike when u crosses a threshold, and resets. Such models are fundamental for neuromorphic computing and event-based learning [10].

5.1.4. *Reinforcement Learning and Value/Policy Iteration.* Differential equations also find their place in reinforcement learning (RL). The evolution of value functions can be described via the Hamilton-Jacobi-Bellman (HJB) equation, a PDE of the form [11]:

$$\frac{\partial V}{\partial t} + \max_a \{r(x, a) + \nabla V \cdot f(x, a)\} = 0,$$

where $V(x,t)$ is the value function, and $r(x,a)$ is the reward. Such formulations connect continuous-time control theory with modern policy optimization algorithms, enabling more natural representations of learning dynamics [11].

5.2. **Partial Differential Equations (PDEs) in AI.**

5.2.1. *Diffusion Models in Deep Generative Learning.* Recent advances in generative modeling have introduced diffusion models, which generate data by simulating the reverse of a stochastic diffusion process. These models are governed by stochastic differential equations (SDEs) and Fokker–Planck-type PDEs. The forward process gradually adds noise to data:

$$dx = f(x,t)dt + g(t)dw,$$

and the reverse-time process is learned to generate realistic samples. This approach has achieved state-of-the-art results in image, audio, and molecule generation [12, 13].

5.2.2. *Physics-Informed Neural Networks (PINNs).* PINNs are a class of neural networks designed to solve PDE-constrained problems by embedding the physical laws directly into the loss function. For example, to solve the heat equation:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u,$$

a PINN minimizes the discrepancy between the neural network output $u_\theta(x,t)$ and the PDE residual. This approach enables data-efficient learning while maintaining physical consistency, and has found applications in fluid dynamics, electromagnetism, and materials science [3].

6. Need for Transforms in AI Systems

In artificial intelligence (AI), especially in continuous-time modeling and dynamical systems, solving complex differential equations is central to understanding and designing system behavior. These differential equations often arise in neural networks, optimization trajectories, control systems, and physical simulations. However, analytical solutions for nonlinear, time-dependent equations are often infeasible or do not exist in closed form. This limitation creates a strong demand for mathematical tools that can simplify, accelerate, and provide structural insight into these systems. Integral transforms serve as one such powerful tool.

6.1. **Challenges in Solving Time-Dependent Nonlinear Differential Equations.** Time-dependent nonlinear differential equations are notoriously difficult to solve analytically due to their complex interdependencies and sensitive behavior under initial conditions. For example, in recurrent neural networks or chaotic systems modeled by differential equations, small perturbations can lead to large divergences in system trajectories. Traditional numerical methods, while effective in some cases, often suffer from high computational cost, instability, or lack of generalization. Thus, transformation-based techniques are critical in re-framing such equations into more tractable forms [18].

6.2. **Benefits of Transformation Methods.** Integral transforms work by converting a function from its original domain (often time or space) into another domain (typically frequency or a complex plane), where the behavior of the system is easier to study. These benefits include:

- **Simplification:** Differential equations become algebraic equations in the transform domain, which are easier to manipulate and solve.
- **Computational Efficiency:** Solving equations in the transformed domain can drastically reduce computational complexity, which is vital for real-time AI systems.
- **Insight into Dynamics:** Transforms often reveal hidden structures in data, such as frequency components, decay rates, or dominant modes of behavior [15].

6.3. **Review of Classical Transforms in AI.** Several classical transforms have found wide applications in AI and related domains:

- **Laplace Transform:** Used primarily for solving linear differential equations and system stability analysis. In control theory and deep learning, Laplace transforms help in understanding impulse responses and feedback dynamics [16].
- **Fourier Transform:** A staple tool in signal processing and neural representations of periodic data. It decomposes signals into constituent frequencies and is used extensively in convolutional neural networks (CNNs), audio processing, and image reconstruction.
- **Wavelet Transform:** Particularly useful in analyzing non-stationary signals and capturing both time and frequency information. Wavelets are commonly used in deep learning for time-series compression, denoising, and hierarchical representations [17].

6.4. **Motivation for a New Transform: The Jani Transform.** Despite the strengths of existing transforms, there is still a need for new, flexible, and problem-specific transformations that can address

gaps in modern AI modeling—particularly in solving highly nonlinear, fractional, or non-autonomous systems encountered in deep learning dynamics. The *Jani Transform*, introduced in this work, is designed as an alternative and complementary tool tailored to handle such complexities.

Key motivations for introducing the Jani Transform include:

- **Adaptability to Nonlinearity:** Unlike classical transforms which often assume linear systems, the Jani Transform is engineered to handle specific classes of nonlinear behavior.
- **Symbolic Interpretability:** It provides symbolic solutions that offer insights into system dynamics, especially valuable in interpretable AI and hybrid physics-ML modeling.
- **Compatibility with AI Workflows:** Its formulation is designed to integrate seamlessly into existing machine learning pipelines, especially those based on continuous-time models like Neural ODEs and PINNs.

In the following sections, we develop the formal structure of the Jani Transform and demonstrate its effectiveness in solving real-world differential equations relevant to AI applications.

Positioning of the Jani Transform in Relation to Existing Transforms. While classical transforms such as the Laplace and Fourier transforms have proven highly effective for linear and time-invariant systems, their direct applicability to nonlinear, memory-sensitive, and time-localized dynamics remains limited. The Jani Transform (JT) is introduced to address this specific gap.

(a) What JT Does Differently. The key structural difference lies in its kernel:

$$K_n(t) = \frac{t^{n-1}e^{-nt}}{\Gamma(n)}.$$

Unlike the Laplace kernel e^{-st} or Fourier kernel $e^{-i\omega t}$, the Jani kernel resembles a Gamma distribution density. This yields:

- Tunable time-localization via the parameter n,
- Natural decay with controllable memory depth,
- Probabilistic interpretability through Gamma-weighting,
- Index-shift operational structure well-suited for derivative manipulation.

In particular, derivative operations in JT reduce to algebraic index shifts rather than multiplication by a transform variable alone, which provides structural flexibility when handling certain nonlinear or memory-based systems.

(b) Use-Cases Where JT is Advantageous. The Jani Transform is particularly suited for:

- Continuous-time neural models with exponential memory decay,
- Spiking neural dynamics where weighted temporal averaging is required,
- Reinforcement learning adaptation laws with convergence behavior,
- Time-series smoothing with controllable locality,
- Systems exhibiting Gamma-type waiting-time or service-time characteristics.

Because the kernel embeds both polynomial growth and exponential decay, JT can emphasize specific temporal regions more flexibly than Laplace or Fourier transforms.

(c) Motivation for Introducing JT. The motivation for introducing the Jani Transform arises from three considerations:

(1) The need for a transform with inherent memory-weighting suitable for AI dynamical systems,
(2) The desire for an operator compatible with continuous-time learning frameworks such as Neural ODEs,
(3) The search for a mathematically interpretable transform bridging probabilistic kernels and operational calculus.

Rather than replacing classical transforms, JT is proposed as a complementary analytical tool tailored to nonlinear and memory-dependent AI systems.

7. Comparison of Jani Transform with others

In the pursuit of solving complex differential equations in artificial intelligence (AI) and dynamical systems modeling, we introduce a novel integral transform: the *Jani Transform*. Designed to provide new flexibility and analytical tractability, the Jani Transform leverages a kernel inspired by the gamma distribution, making it particularly suitable for applications where time-decay and memory effects play a crucial role.

7.1. **Kernel Explanation.**

The kernel of the Jani Transform,

$$K_n(t) = \frac{t^{n-1}e^{-nt}}{\Gamma(n)},$$

resembles the probability density function of the Gamma distribution with shape parameter n and rate parameter n. This structure provides:

- A natural weighting that emphasizes early-time behavior for small n, and long-term decay for large n.

- Interpretability in modeling memory-based processes such as spiking neural dynamics or delay systems in AI.

7.2. **Comparison with Laplace and Fourier Transforms.** The Laplace and Fourier transforms have been widely studied in the context of classical control theory, signal processing, and linear system analysis [15, 34]. Their mathematical properties, domain limitations, and applications in neural and frequency-based models have been reviewed thoroughly.

In contrast, the Jani Transform, newly proposed in this work, was developed to overcome the challenges faced in solving nonlinear time-dependent differential equations, especially those found in neural dynamics, reinforcement learning, and time-series forecasting. Its kernel resembles the gamma distribution, allowing for better modeling of systems with decaying memory and nonlinear feedback—properties crucial in deep learning models and biological systems.

The comparative metrics are considered qualitatively, based on:

- Transform kernel behavior
- Domain of applicability (e.g., time vs frequency)
- Suitability for nonlinear system representation
- Ability to preserve memory effects (important in RNNs and spiking neural networks)
- Theoretical compatibility with probabilistic AI models
- Analytical fit in modeling AI systems governed by ODEs and PDEs

Table 1. Comparison of Jani, Laplace, and Fourier Transforms

Property	**Laplace**	**Fourier**	**Jani**
Kernel	e^{-st}	$e^{-i\omega t}$	$\frac{t^{n-1}e^{-nt}}{\Gamma(n)}$
Domain	$t \geq 0$	$-\infty < t < \infty$	$t \geq 0$
Captures	Exponential decay	Frequency content	Time-localized decay and memory
Handles Nonlinearity	Limited	Limited	Designed for it
Probabilistic View	No	No	Yes (Gamma-like kernel)
AI Application Fit	Medium	Medium	High

7.3 Structural Comparison with Laplace and Fourier Transforms

In this section, we provide a structural and operational comparison between the Jani Transform (JT), the Laplace Transform (LT), and the Fourier Transform (FT). The goal is not to claim superiority, but to clarify mathematical differences in kernel structure, operational calculus, and domain behavior.

7.3.1 Kernel Structure. The three transforms are defined through kernels of distinct mathematical nature:

$$\text{Laplace: } K(s,t) = e^{-st},$$

$$\text{Fourier: } K(\omega,t) = e^{-i\omega t},$$

$$\text{Jani: } K_n(t) = \frac{t^{n-1}e^{-nt}}{\Gamma(n)}.$$

Key structural differences:

- The Laplace and Fourier kernels are purely exponential.
- The Jani kernel combines polynomial growth and exponential decay.
- The Jani kernel is equivalent to a Gamma distribution density with shape and rate both equal to n.

Thus, JT introduces an intrinsic time-localized weighting mechanism absent in LT and FT.

7.3.2 Derivative Mapping Structure. The Laplace transform converts derivatives as:

$$\mathcal{L}\{f'(t)\} = sF(s) - f(0),$$

while the Fourier transform gives:

$$\mathcal{F}\{f'(t)\} = i\omega F(\omega).$$

In contrast, the Jani Transform satisfies:

$$J_n\{f'(t)\} = -f(0) + nJ_{n-1}\{f(t)\}.$$

Unlike Laplace and Fourier transforms, which multiply by a transform variable, the Jani Transform produces an *index shift*.

This index-shift structure introduces a discrete hierarchical relation among transform orders, which may be advantageous in systems where parameterized memory depth is relevant.

7.3.3 Domain and Convergence.

- Laplace transform requires $f(t)$ of exponential order.
- Fourier transform typically requires absolute integrability.
- Jani Transform requires $\Re(n) > 0$ and Gamma-integrability conditions.

The decay factor e^{-nt} ensures integrability for sufficiently large n, while the polynomial term t^{n-1} allows controlled emphasis of early or intermediate time behavior.

8. Jani Transform for Solving Differential Equations in AI

Modern artificial intelligence (AI) models increasingly rely on differential equations to capture the continuous dynamics of hidden states, memory decay, and adaptive control policies. However, solving these equations analytically—especially in nonlinear and time-dependent scenarios—remains a major challenge. The *Jani Transform*, with its gamma-distribution-inspired kernel and flexible integral form, offers a novel route to tackle such systems efficiently.

In this section, we illustrate how the Jani Transform can be used to solve differential equations arising in various AI models. Each example highlights the modeling utility and interpretive power of the transform.

8.1. **Example 1: Hidden State Evolution in Neural ODEs.** Neural Ordinary Differential Equations (Neural ODEs) describe the evolution of a hidden state $h(t)$ as a continuous function of time:

$$\frac{dh(t)}{dt} = f(h(t), \theta), \tag{8.1}$$

where f is a neural network parameterized by weights θ. In the linearized case, the equation becomes:

$$\frac{dh(t)}{dt} = -ah(t), \quad h(0) = h_0, \tag{8.2}$$

where $a > 0$ is a constant decay rate.

Step 1: Apply the Jani Transform. Taking the Jani Transform of both sides:

$$\mathcal{J}_n\left\{\frac{dh(t)}{dt}\right\} = -a\,\mathcal{J}_n\{h(t)\}.$$

Using the derivative property of the Jani Transform:

$$\mathcal{J}_n\left\{\frac{dh(t)}{dt}\right\} = -n\,\mathcal{J}_n\{h(t)\} + \frac{h_0}{\Gamma(n)},$$

we get:

$$-n\,\mathcal{J}_n\{h(t)\} + \frac{h_0}{\Gamma(n)} = -a\,\mathcal{J}_n\{h(t)\}.$$

Step 2: Solve algebraically in the transform domain. Rearranging terms:

$$(a-n)\,\mathcal{J}_n\{h(t)\} = -\frac{h_0}{\Gamma(n)},$$

thus:

$$\mathcal{J}_n\{h(t)\} = \frac{h_0}{\Gamma(n)\,(n+a)}.$$

Step 3: Apply the Inverse Jani Transform. From the standard table of Jani Transforms:

$$\mathcal{J}_n^{-1}\left\{\frac{1}{n+a}\right\} = e^{-at},$$

therefore:

$$h(t) = h_0\, e^{-at}.$$

Final Solution:

$$\boxed{h(t) = h_0\, e^{-at}}$$

This shows that, in the linearized setting, the hidden state decays exponentially over time with rate a, preserving the memory-decay interpretation of the Jani Transform's Gamma kernel.

Interpretation: This shows that the hidden state in a Neural ODE decays exponentially—exactly what the Jani Transform confirms in closed form.

8.2. **Example 2: Leaky Integrate-and-Fire (LIF) Neuron Model.** The Leaky Integrate-and-Fire (LIF) neuron models the membrane potential $V(t)$ as:

$$\frac{dV(t)}{dt} = -\frac{V(t)}{\tau} + I(t), \tag{8.3}$$

where $\tau > 0$ is the membrane time constant and $I(t)$ is the input current. Consider a constant step input:

$$I(t) = I_0, \quad t \geq 0.$$

Step 1: Apply the Jani Transform. Taking the Jani Transform of both sides:

$$\mathcal{J}_n\left\{\frac{dV(t)}{dt}\right\} = -\frac{1}{\tau}\,\mathcal{J}_n\{V(t)\} + \mathcal{J}_n\{I_0\}.$$

Using the derivative property:

$$-n\,\mathcal{J}_n\{V(t)\} + \frac{V(0)}{\Gamma(n)} = -\frac{1}{\tau}\,\mathcal{J}_n\{V(t)\} + \frac{I_0}{n^n}.$$

Step 2: Solve algebraically in the transform domain. Rearranging:

$$\left(-n + \frac{1}{\tau}\right)\mathcal{J}_n\{V(t)\} = -\frac{V(0)}{\Gamma(n)} + \frac{I_0}{n^n},$$

which gives:

$$\mathcal{J}_n\{V(t)\} = \frac{\frac{V(0)}{\Gamma(n)} - \frac{I_0}{n^n}}{n - \frac{1}{\tau}}.$$

Step 3: Apply the inverse Jani Transform. From the Jani Transform table:

$$\mathcal{J}_n^{-1}\left\{\frac{1}{n-a}\right\} = e^{at}, \quad \mathcal{J}_n^{-1}\left\{\frac{1}{n^n}\right\} = 1,$$

the time-domain solution becomes:

$$V(t) = (V(0) - \tau I_0)\, e^{-t/\tau} + \tau I_0.$$

Final Solution:

$$\boxed{V(t) = (V(0) - \tau I_0)\, e^{-t/\tau} + \tau I_0}$$

This result shows the exponential decay of the initial membrane potential towards the steady-state value τI_0 under constant input.

Interpretation: This is the classic LIF behavior with exponential decay and input charging, derived using Jani Transform.

8.3. **Example 3: Reinforcement Learning Policy Dynamics.** In continuous-time reinforcement learning, the evolution of the policy $\pi(t)$ can be modeled by the first-order differential equation:

$$\frac{d\pi(t)}{dt} = -\beta\left[\pi(t) - \pi^*\right], \quad \pi(0) = \pi_0, \tag{8.4}$$

where:

- $\pi(t)$ is the policy at time t,
- π^* is the optimal (target) policy,
- $\beta > 0$ is the adaptation (learning) rate.

Step 1: Rewrite in standard form.

$$\frac{d\pi(t)}{dt} = -\beta\pi(t) + \beta\pi^*.$$

Step 2: Apply the Jani Transform. Taking the Jani Transform of both sides and using the first derivative property:

$$-n\,\mathcal{J}_n\{\pi(t)\} + \frac{\pi_0}{\Gamma(n)} = -\beta\,\mathcal{J}_n\{\pi(t)\} + \mathcal{J}_n\{\beta\pi^*\}.$$

From the basic transform table:

$$\mathcal{J}_n\{\beta\pi^*\} = \frac{\beta\pi^*}{n^n}.$$

Step 3: Solve algebraically in the transform domain. Rearranging terms:

$$(-n + \beta)\,\mathcal{J}_n\{\pi(t)\} = -\frac{\pi_0}{\Gamma(n)} + \frac{\beta\pi^*}{n^n}.$$

Thus:

$$\mathcal{J}_n\{\pi(t)\} = \frac{\frac{\pi_0}{\Gamma(n)} - \frac{\beta\pi^*}{n^n}}{n - \beta}.$$

Step 4: Apply the inverse Jani Transform. Using:

$$\mathcal{J}_n^{-1}\left\{\frac{1}{n-a}\right\} = e^{at}, \quad \mathcal{J}_n^{-1}\left\{\frac{1}{n^n}\right\} = 1,$$

we obtain:

$$\pi(t) = \pi^* + (\pi_0 - \pi^*)\, e^{-\beta t}.$$

Final Solution:

$$\boxed{\pi(t) = \pi^* + (\pi_0 - \pi^*)\, e^{-\beta t}}$$

This expression shows that the policy exponentially converges from its initial value π_0 towards the optimal policy π^* at rate β.

Interpretation: This result demonstrates how a policy smoothly converges toward the optimal policy π^*, with the decay rate governed by β. The Jani Transform elegantly handles the dynamic adaptation process.

8.4. **Conclusion of Section.** These examples underscore the power of the Jani Transform as a systematic tool to simplify and solve differential equations arising in AI. Whether modeling neural hidden states, spiking dynamics, or reinforcement learning policies, the transform provides a bridge from time-domain complexity to algebraic clarity.

9. Jani Transform in Time-Series AI Models

Time-series data is foundational to a wide spectrum of AI applications, ranging from financial forecasting and healthcare monitoring to sensor data analysis and speech processing. Traditional approaches often rely on autoregressive (AR) models, Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks to capture temporal dependencies. However, many time-series signals are inherently nonlinear, noisy, and multi-scale in nature. This complexity demands powerful preprocessing and feature extraction techniques to enhance model interpretability and performance.

In this context, the **Jani Transform (JT)** emerges as a powerful alternative to classical transformation techniques like Fourier and Wavelet Transforms. Defined as:

$$\mathcal{J}_n\{f(t)\} = \int_0^\infty f(t) \cdot \frac{t^{n-1}e^{-nt}}{\Gamma(n)}\, dt, \tag{9.1}$$

the JT acts like a gamma-kernel-weighted smoothing operator. It is well-suited for time-series analysis due to its ability to emphasize recent dynamics while suppressing distant past variations in a controllable manner through the parameter n.

9.1. **Time-Series Forecasting with JT.** In traditional models such as AR or RNN, the input sequence $\{x_t\}$ is fed into a recursive architecture to learn temporal dependencies. Instead of feeding raw time-series data, one can preprocess the sequence using the Jani Transform:

$$x_t^{(\mathrm{JT})} = \mathcal{J}_n\{x_{t-\tau}\},$$

where τ is the look-back window. These smoothed or transformed signals help reduce temporal noise and emphasize recent events, thus improving model stability and convergence—especially in models like LSTMs which are sensitive to vanishing gradients over long sequences.

Example: In financial time-series forecasting, applying JT to daily price returns can reveal smoothed trends and reduce volatility spikes, enabling better learning in neural models.

9.2. **Signal Decomposition and Denoising.** Many real-world time-series are multi-scale and corrupted with high-frequency noise. Classical Fourier transforms offer global frequency insights but lack time-localization. Wavelet transforms improve localization but require choosing a mother wavelet.

The JT provides a middle ground with its decaying exponential kernel and can be used to perform scale-sensitive signal decomposition:

$$f(t) = \sum_{n=1}^{N} c_n \cdot \phi_n(t), \quad \text{where } c_n = \mathcal{J}_n\{f(t)\},$$

and $\phi_n(t)$ denotes JT basis functions. This decomposition allows reconstructing the signal using meaningful components—low-n capturing global trends, and high-n capturing local variations.

9.3. **Feature Extraction using Jani Coefficients.** In machine learning pipelines, the transformed coefficients $\{c_n\}$ from JT can act as powerful features, similar to Fourier or Mel-frequency coefficients. These features are particularly beneficial in domains such as:

- **Speech processing:** capturing vocal tract dynamics.
- **Health signals:** such as ECG or EEG for detecting abnormal rhythms.
- **Climate modeling:** decomposing long-range dependencies in temperature patterns.

By adjusting the parameter n, one can zoom in or out of signal resolution, much like in multi-resolution analysis.

9.4. **Why Prefer Jani Transform in ML?.** Compared to Fourier (which assumes stationarity) and Wavelets (which require basis selection), the Jani Transform offers:

- Tunable locality via the parameter n,
- Natural decay for long-term memory effects (useful in RNNs),
- Analytical simplicity and interpretability,
- Direct connection to statistical distributions (gamma kernel).

Thus, JT can serve as a lightweight, adaptable, and theory-grounded alternative in AI-based time-series models.

10. Future Scope

While the foundational theory and applications of the Jani Transform (JT) in artificial intelligence (AI) are promising, several advanced directions remain open for exploration. In this section, we highlight potential areas where JT can be further extended and hybridized to enhance the modeling of complex systems such as fractional dynamics, fuzzy uncertainty, quantum computation, and energy-constrained AI applications.

10.1. **Jani Transform in Fractional Differential Equations and Fuzzy Systems.** Fractional differential equations (FDEs) are gaining significant traction in modeling memory-intensive and hereditary systems, such as viscoelastic materials, bioengineering, and even AI training dynamics [27]. The kernel structure of JT resembles the gamma distribution, which inherently aligns with the memory properties of Caputo-type derivatives. Therefore, developing a *fractional Jani Transform* could offer a novel analytical tool for solving nonlinear FDEs with greater flexibility and accuracy.

In parallel, fuzzy differential equations (FzDEs) are critical in handling vagueness and uncertainty in real-world systems, especially in fuzzy control and AI decision-making. JT's smoothing properties can be tailored to reduce fuzziness in differential models, thereby offering a bridge between fuzzy logic and continuous-time AI modeling.

10.2. **Hybrid JT-Laplace Framework.** Given that the Laplace transform excels at handling linear, time-invariant systems, and the JT is better suited for memory-sensitive, nonlinear, or time-local problems, a hybrid JT-Laplace framework could yield improved analytical power. Such a composite transform could be formally defined as:

$$\mathcal{L}\left[\mathcal{J}_n\{f(t)\}\right] \quad \text{or} \quad \mathcal{J}_n\left[\mathcal{L}\{f(t)\}\right],$$

depending on the application. This hybridization could enable exact solutions for complex systems such as switching neural networks, adaptive controllers, and hybrid discrete-continuous AI agents, enhancing the existing toolkit of applied mathematicians and AI researchers.

10.3. **JT for Energy-Efficient AI Modeling.** In the age of edge computing and climate-aware AI, energy efficiency is a priority. Many machine learning models face bottlenecks due to high-dimensional input data and recurrent computations. JT's ability to smooth and compress time-series data using tunable n parameters can reduce computational overhead without significant loss in model fidelity.

- JT can be embedded as a lightweight preprocessing layer in neural architectures.
- Time-domain signals can be filtered using JT to retain essential dynamics and discard high-frequency noise.

This approach could be particularly useful in real-time applications like drone navigation, wearable health monitors, or smart home sensors, where low-latency and power-aware models are necessary.

10.4. **Application in Quantum Machine Learning.** Quantum Machine Learning (QML) seeks to accelerate computation using the principles of quantum mechanics. Many quantum models, such as Quantum Reservoir Computing and Quantum Boltzmann Machines, depend on the evolution of quantum states governed by Schrödinger-type differential equations.

The JT, due to its compatibility with exponential-decay kernels, could act as a bridge to develop quantum-classical hybrid models. For instance, JT could be used to analyze quantum observables in time, offering smoother, interpretable statistics before inputting into classical decision-making layers.

Moreover, JT's integral form could potentially be adapted to work with quantum gates and continuous-variable quantum systems, making it suitable for use in photonic quantum computing platforms.

11. Conclusion

The Jani Transform (JT) introduces a mathematically rigorous and versatile framework for modeling nonlinear, memory-dependent, and time-evolving systems in artificial intelligence. By leveraging a gamma-kernel-based operator, JT generalizes beyond Laplace and Fourier transforms, enabling interpretable solutions to problems in Neural ODEs, reinforcement learning, spiking dynamics, and time-series modeling.

Unlike purely data-driven black-box methods, JT emphasizes analytical transparency and provides algebraic simplification of complex

dynamics. This dual role as both a theoretical tool and a practical modeling technique positions JT as a bridge between classical differential equation theory and modern neural architectures.

At the same time, several open challenges remain, including efficient numerical inversion strategies, scalability to high-dimensional data, and systematic benchmarking against existing transforms. Addressing these limitations will be essential for translating JT's theoretical strengths into widespread adoption.

Looking ahead, promising directions include integrating JT with fractional calculus for long-memory systems, hybrid JT–Laplace frameworks for mixed linear–nonlinear dynamics, and applications in energy-efficient neuromorphic and quantum-inspired AI. Such extensions can transform JT into a unifying analytical tool for next-generation AI pipelines.

In essence, the Jani Transform advances the philosophy that sustainable progress in AI requires not only data and computational power, but also the revival of mathematical elegance—where rigorous theory informs practical innovation.

Declarations

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability: No datasets were generated or analyzed during the current study; hence, data sharing is not applicable.

Competing Interests: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural Ordinary Differential Equations. *Advances in Neural Information Processing Systems*, 31.

[2] Strogatz, S. H. (2018). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* (2nd ed.). CRC Press.

[3] Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378, 686–707.

[4] Jani, H. P., & Singh, T. R. (2025). A study of water infiltration in unsaturated soils by Aboodh transform homotopy perturbation method. *International Journal of Applied and Computational Mathematics*, 11(2), 52.

[5] Khalil, H. K. (2002). *Nonlinear Systems* (3rd ed.). Prentice Hall.
[6] Wiggins, S. (2003). *Introduction to Applied Nonlinear Dynamical Systems and Chaos* (2nd ed.). Springer.
[7] Jani, H. P., & Singh, T. (2025). Solution of fractional order Schrödinger equation by using Aboodh transform homotopy perturbation method. *Journal of Fractional Calculus and Applications*, 16(1), 1–13.
[8] Funahashi, K. I., & Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6), 801–806.
[9] Chang, B., Meng, L., Haber, E., Ruthotto, L., & Begert, D. (2019). AntisymmetricRNN: A dynamical system view on recurrent neural networks. *International Conference on Learning Representations (ICLR)*.
[10] Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press.
[11] Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural Computation*, 12(1), 219–245.
[12] Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*.
[13] Song, Y., & Ermon, S. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.
[14] Jani, H. P., & Singh, T. R. (2023). Study of one dimensional groundwater recharge through porous media by Aboodh transform homotopy perturbation method. *International Journal of Applied and Computational Mathematics*, 9(6), 132.
[15] Bracewell, R. (2000). *The Fourier Transform and Its Applications* (3rd ed.). McGraw-Hill.
[16] Oppenheim, A. V., Willsky, A. S., & Nawab, S. H. (1996). *Signals and Systems* (2nd ed.). Prentice Hall.
[17] Mallat, S. (1999). *A Wavelet Tour of Signal Processing*. Academic Press.
[18] Arfken, G. B., Weber, H. J., & Harris, F. E. (2013). *Mathematical Methods for Physicists: A Comprehensive Guide* (7th ed.). Academic Press.
[19] Jani, H. P., & Singh, T. R. (2022). Some examples of Swift–Hohenberg equation. *Examples and Counterexamples*, 2, 100090.
[20] Gerstner, W., & Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.
[21] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.
[22] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
[23] Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5), 961–1005.
[24] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. Wiley.
[25] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
[26] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
[27] Podlubny, I. (1998). *Fractional Differential Equations*. Academic Press.
[28] Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning—I. *Information Sciences*, 8(3), 199–249.

[29] Chakraborty, S., et al. (2022). Quantum Machine Learning: A Review and Current Status. *IEEE Access*, 10, 45632–45649.
[30] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
[31] Hairer, E., Lubich, C., & Wanner, G. (2006). *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer.
[32] Brunton, S. L., & Kutz, J. N. (2019). *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press.
[33] Jani, H. P., & Singh, T. R. (2023). Solution of time fractional Swift–Hohenberg equation by Aboodh transform homotopy perturbation method. *International Journal of Nonlinear Analysis and Applications*, 14(1), 1005–1013.
[34] Doetsch, G. (1974). *Introduction to the Theory and Application of the Laplace Transformation*. Springer.