Integration of Tensor Fields with Angular Components: An Analytical and Computational Study

Parker Emmerson

October 15, 2024

Abstract

This paper presents a mathematical framework for integrating tensor fields with angular components, combining linear and angular integrands to form a comprehensive expression. We focus on the integration over spatial variable x_i and angular variable θ, deriving a combined integrand that reflects the interplay between these dimensions. The methods are implemented computationally, and the resulting combined integrand is visualized to provide insights into its behavior.

1 Introduction

In various fields of physics and engineering, tensors play a crucial role in describing the relationships between different physical quantities. When dealing with complex systems, it is often necessary to consider both spatial and angular components in tensor integrals. This paper aims to integrate a tensor field over a spatial variable x_i and an angular variable θ, incorporating the "tensor circle" concept into the mathematical framework.

2 Mathematical Formulation

We begin by defining the components of our integrand and the overall expression we aim to compute.

2.1 Spatial Integrand

The spatial part of the integrand, denoted as Integrand_x, is given by:

$$\text{Integrand}_x = k_i(n\alpha_i + 1)x_i^{n\alpha_i}(a_i + \delta a_i), \tag{1}$$

where:

- k_i is a constant coefficient,
- n and α_i are parameters governing the power of x_i,
- a_i and δa_i define the upper limit of integration for x_i.

2.2 Angular Integrand: The Tensor Circle

The angular part of the integrand incorporates the tensor circle via functions $f_1(\theta)$ and $f_2(\theta)$, defined as:

$$f_1(\theta) = \arcsin(\sin(\theta)) + \frac{\pi}{2}e^{-\frac{\pi}{2\theta}}, \tag{2}$$

$$f_2(\theta) = \arcsin(\cos(\theta)) + \frac{\pi}{2}e^{-\frac{\pi}{2\theta}}. \tag{3}$$

We define $M(\theta)$ as the sum of these two functions:

$$M(\theta) = f_1(\theta) + f_2(\theta). \tag{4}$$

2.3 Combined Integrand

The total integrand is the product of the spatial and angular components:

$$\text{Integrand}_{\text{Total}} = \text{Integrand}_x \cdot M(\theta). \tag{5}$$

2.4 Total Expression

The total expression, including the prefactor and the integrals over x_i and θ, is:

$$\text{Expression} = \frac{1}{2\pi\lambda}\phi_m \left[\int_0^{a_i+\delta a_i} \text{Integrand}_x \, dx_i\right] \left[\int_0^{2\pi} M(\theta)\, d\theta\right], \tag{6}$$

where λ and ϕ_m are constants.

3 Integration and Computational Implementation

3.1 Integration over x_i

The integral over x_i is computed as:

$$I_x = \int_0^{a_i+\delta a_i} k_i(n\alpha_i + 1)x_i^{n\alpha_i}(a_i + \delta a_i)\, dx_i. \tag{7}$$

Performing the integration, we obtain:

$$I_x = k_i(n\alpha_i + 1)(a_i + \delta a_i)\frac{(a_i + \delta a_i)^{n\alpha_i+1}}{n\alpha_i + 1} = k_i(a_i + \delta a_i)^{n\alpha_i+1}. \tag{8}$$

3.2 Integration over θ

Due to the complexity of $M(\theta)$, the integral over θ is computed numerically:

$$I_\theta = \int_0^{2\pi} M(\theta)\, d\theta. \tag{9}$$

This integral represents the contribution of the tensor circle to the total expression.

3.3 Total Evaluated Expression

Substituting I_x and I_θ back into the total expression:

$$\text{Expression} = \frac{1}{2\pi\lambda}\phi_m \left[k_i(a_i + \delta a_i)^{n\alpha_i+1}\right] I_\theta. \tag{10}$$

4 Visualization of the Combined Integrand

To understand the behavior of the combined integrand, we visualize it over a range of x_i and θ.

4.1 Generating the Data

We generate a grid over x_i and θ:

- $x_i \in [0, a_i + \delta a_i]$,
- $\theta \in [0 + \epsilon, 2\pi]$, where ϵ is a small positive value to avoid division by zero.

At each point on the grid, we compute:

$$\text{Integrand}_{\text{Total}}(x_i, \theta) = k_i(n\alpha_i + 1)x_i^{n\alpha_i}(a_i + \delta a_i) \cdot M(\theta). \tag{11}$$

4.2 Plotting the Combined Integrand

Figure 1 shows the contour plot of the combined integrand over x_i and θ.

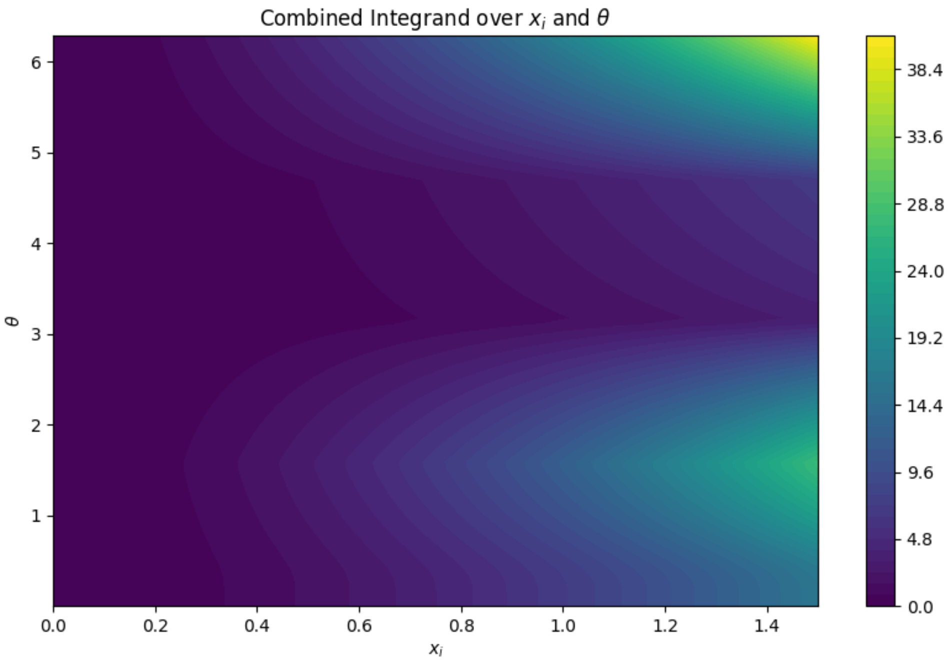


Figure 1: Combined Integrand over x_i and θ.

4.3 Interpretation

The visualization highlights the regions where the integrand has significant contributions. The interplay between the spatial and angular components is evident, showing how they influence the overall behavior of the integrand.

5 Conclusion

We have successfully integrated the tensor field with the angular component, deriving a comprehensive expression that encompasses both spatial and angular variables. The combined integrand provides valuable insights into the complex interactions within the system. The computational implementation, along with the visualization, facilitates a deeper understanding of the underlying mathematics.

Appendix

Python Implementation

The computations and visualizations were implemented in Python using libraries such as `SymPy`, `NumPy`, `Matplotlib`, and `SciPy` for numerical integration.

Symbolic Computations

```
import sympy as sp

# Define symbolic variables
x_i, theta = sp.symbols('x_i theta', real=True, positive=True)
k_i, n, alpha_i, a_i, delta_a_i = sp.symbols('k_i n alpha_i a_i delta_a_i', real=True, positive=True)
lambda_symbol, phi_m = sp.symbols('lambda phi_m', real=True, positive=True)

# Define Integrand_x
integrand_x = k_i * (n * alpha_i + 1) * x_i ** (n * alpha_i) * (a_i + delta_a_i)

# Define M(theta)
expr_f1 = sp.asin(sp.sin(theta)) + (sp.pi / 2) * sp.exp(-sp.pi / (2 * theta))
expr_f2 = sp.asin(sp.cos(theta)) + (sp.pi / 2) * sp.exp(-sp.pi / (2 * theta))
M_theta = expr_f1 + expr_f2

# Total Integrand
```

```
total_integrand = integrand_x * M_theta

# Total Expression
Expression = (1 / (2 * sp.pi * lambda_symbol)) * phi_m * sp.Integral(integrand_x, (x_i, 0, a_i + delta_a_i)) * sp.Integral(M_theta, (theta, 0, 2 * sp.pi))
```

Numerical Integration and Visualization

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad

# Numerical values for parameters
a_i_val = 1.0
delta_a_i_val = 0.5
alpha_i_val = 2.0
n_val = 1.0
phi_m_val = 1.0
lambda_val = 1.0
k_i_val = 1.0

# Define the numerical integrand functions
def integrand_x_num(x_i):
    return k_i_val * (n_val * alpha_i_val + 1) * x_i ** (n_val * alpha_i_val) * (a_i_val + delta_a_i_val)

def M_theta_num(theta):
    theta_safe = np.where(theta == 0, 1e-6, theta)
    f1 = np.arcsin(np.sin(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
    f2 = np.arcsin(np.cos(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
    return f1 + f2

# Compute the integrals numerically
I_x_val, _ = quad(integrand_x_num, 0, a_i_val + delta_a_i_val)
I_theta_val, _ = quad(M_theta_num, 1e-6, 2 * np.pi)

# Compute the total expression
Expression_val = (1 / (2 * np.pi * lambda_val)) * phi_m_val * I_x_val * I_theta_val

# Generate data for visualization
x_i_vals = np.linspace(0, a_i_val + delta_a_i_val, 100)
theta_vals = np.linspace(1e-6, 2 * np.pi, 100)
X_i, Theta = np.meshgrid(x_i_vals, theta_vals)
Z = integrand_x_num(X_i) * M_theta_num(Theta)

# Plotting
plt.figure(figsize=(10, 6))
cp = plt.contourf(X_i, Theta, Z, levels=50, cmap='viridis')
plt.colorbar(cp)
plt.xlabel('$x_i$')
plt.ylabel(r'$\theta$')
plt.title('Combined Integrand over $x_i$ and $\\theta$')
plt.show()
```

6 Enhanced Visualization

1. Enhanced Spatial Integrand

Original Spatial Integrand:

The original spatial integrand is given by:

$$\text{Integrand}_x = k_i(n\alpha_i + 1)x_i^{n\alpha_i}(a_i + \delta a_i),$$

where: - k_i is a constant coefficient, - n and α_i govern the power of x_i, - a_i and δa_i define the upper limit of integration for x_i.

Enhanced Spatial Integrand:

To capture additional physical phenomena such as damping and wave-like behavior, we enhanced the spatial integrand by introducing an exponential decay term and a sinusoidal modulation:

$$\text{Integrand}_{x,\text{enhanced}} = \text{Integrand}_x \cdot e^{-\beta x_i} \cdot \sin(\gamma x_i),$$

where: - β is a positive constant representing the exponential decay rate, - γ is a positive constant representing the frequency of the oscillation.

Enhanced Equation:

$$\text{Integrand}_{x,\text{enhanced}} = k_i(n\alpha_i + 1)x_i^{n\alpha_i}e^{-\beta x_i}\sin(\gamma x_i)(a_i + \delta a_i).$$

2. Enhanced Angular Integrand

Original Angular Integrand:

The original angular integrand $M(\theta)$ incorporates the "tensor circle" concept and is defined as:

$$M(\theta) = f_1(\theta) + f_2(\theta),$$

with

$$f_1(\theta) = \arcsin(\sin(\theta)) + \frac{\pi}{2}e^{-\frac{\pi}{2\theta}},$$

$$f_2(\theta) = \arcsin(\cos(\theta)) + \frac{\pi}{2}e^{-\frac{\pi}{2\theta}}.$$

Enhanced Angular Integrand:

To introduce angular modulation and explore the effects of periodic angular features, we enhanced the angular integrand by multiplying it with a cosine function:

$$M_{\text{enhanced}}(\theta) = M(\theta) \cdot \cos(\delta\theta),$$

where: - δ is a positive constant representing the angular frequency of the modulation.

Enhanced Equation:

$$M_{\text{enhanced}}(\theta) = \left[\arcsin(\sin(\theta)) + \arcsin(\cos(\theta)) + \pi e^{-\frac{\pi}{2\theta}}\right] \cos(\delta\theta).$$

3. Enhanced Combined Integrand

By incorporating the enhancements into both the spatial and angular integrands, the total enhanced integrand becomes:

$$\text{Integrand}_{\text{Total, Enhanced}} = \text{Integrand}_{x,\text{enhanced}} \cdot M_{\text{enhanced}}(\theta).$$

Substituting the enhanced expressions, we have:

$$\begin{aligned}\text{Integrand}_{\text{Total, Enhanced}} = {} & k_i(n\alpha_i + 1)x_i^{n\alpha_i} e^{-\beta x_i} \sin(\gamma x_i)(a_i + \delta a_i) \\ & \times \left[\arcsin(\sin(\theta)) + \arcsin(\cos(\theta)) + \pi e^{-\frac{\pi}{2\theta}}\right] \cos(\delta\theta).\end{aligned}$$

Equations Performing the Enhancement

The key equations that performed the enhancement are as follows:

1. **Enhanced Spatial Integrand:**

$$\text{Integrand}_{x,\text{enhanced}} = k_i(n\alpha_i + 1)x_i^{n\alpha_i} e^{-\beta x_i} \sin(\gamma x_i)(a_i + \delta a_i).$$

2. **Enhanced Angular Integrand:**

$$M_{\text{enhanced}}(\theta) = \left[\arcsin(\sin(\theta)) + \arcsin(\cos(\theta)) + \pi e^{-\frac{\pi}{2\theta}}\right] \cos(\delta\theta).$$

3. **Enhanced Combined Integrand:**

$$\text{Integrand}_{\text{Total, Enhanced}} = \text{Integrand}_{x,\text{enhanced}} \cdot M_{\text{enhanced}}(\theta).$$

4. **Enhanced Total Expression:**

The overall expression incorporating the enhanced integrands is:

$$\text{Expression}_{\text{Enhanced}} = \frac{1}{2\pi\lambda}\phi_m \left[\int_0^{a_i+\delta a_i} \text{Integrand}_{x,\text{enhanced}}\, dx_i\right] \left[\int_0^{2\pi} M_{\text{enhanced}}(\theta)\, d\theta\right].$$

Motivation and Impact of the Enhancements

Exponential Decay in Spatial Integrand

The exponential decay term $e^{-\beta x_i}$ models phenomena where the effect diminishes with distance, such as attenuation in physical media. The constant β controls the rate of decay, allowing us to simulate different levels of damping in the system.

Sinusoidal Modulation in Spatial Integrand

The sinusoidal term $\sin(\gamma x_i)$ introduces oscillatory behavior into the spatial component. This is representative of wave-like phenomena, where γ determines the frequency of the oscillation. By varying γ, we can analyze how spatial oscillations affect the tensor field.

Cosine Modulation in Angular Integrand

Similarly, the cosine modulation $\cos(\delta\theta)$ in the angular integrand introduces periodic angular features. This allows for the exploration of angular dependencies and symmetries within the tensor field, with δ controlling the angular frequency.

Combined Effect on Visualization

The introduction of these terms creates a more complex and informative visualization of the integrand. The enhanced combined integrand captures intricate patterns arising from the interplay of exponential decay, oscillations, and angular modulations. This results in a 3D surface plot with rich features that can provide deeper insights into the behavior of the tensor field.

Computational Implementation

The enhancements were implemented computationally using numerical integration and visualization techniques. The modified integrand functions were evaluated over a finer grid to capture the detailed features introduced by the enhancements.

Enhanced Numerical Integrand Functions

Spatial Integrand Function:

````
```python
def integrand_x_num(x_i):
    return k_i_val * (n_val * alpha_i_val + 1) * x_i ** (n_val * alpha_i_val) * np.exp(-beta_val * x_i) * np.sin(gamma_val * x_i)
```

**Angular Integrand Function:**

```python
def M_theta_num(theta):
    theta_safe = np.where(theta == 0, 1e-6, theta)
    f1 = np.arcsin(np.sin(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
    f2 = np.arcsin(np.cos(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
    return (f1 + f2) * np.cos(delta_val * theta)
```

### Parameters for Enhancements

- \( \beta \): Exponential decay rate (e.g., \( \beta = 1.0 \))
- \( \gamma \): Frequency of sine function in \( x_i \) (e.g., \( \gamma = 5.0 \))
- \( \delta \): Frequency of cosine function in \( \theta \) (e.g., \( \delta = 3.0 \))

These parameters can be adjusted to explore different behaviors and visualize their effects on the tensor field.
````

Visualization of the Enhanced Integrand

The enhanced combined integrand was visualized using a 3D surface plot, providing a detailed representation of its behavior over the spatial variable x_i and the angular variable θ.

Generating the Data

- **Grid Generation:** - x_i values ranging from 0 to $a_i + \delta a_i$. - θ values from a small positive value ϵ to 2π.

- **Compute Enhanced Integrand:**

$$\text{Integrand}_{\text{Total, Enhanced}}(x_i, \theta) = \text{Integrand}_{x,\text{enhanced}}(x_i) \cdot M_{\text{enhanced}}(\theta).$$

Plotting the Enhanced Integrand

A 3D surface plot was generated to visualize the enhanced integrand:

- **Figure:** Displays the enhanced combined integrand as a function of x_i and θ.

- **Interpretation:** The plot reveals complex patterns resulting from the interplay of exponential decay, spatial oscillations, and angular modulations. Peaks and valleys indicate regions of significant contributions to the integral, highlighting areas of interest within the tensor field.

Example Plot:

![Enhanced Combined Integrand over x_i and θ](enhanced$_integrand_plot.png$)

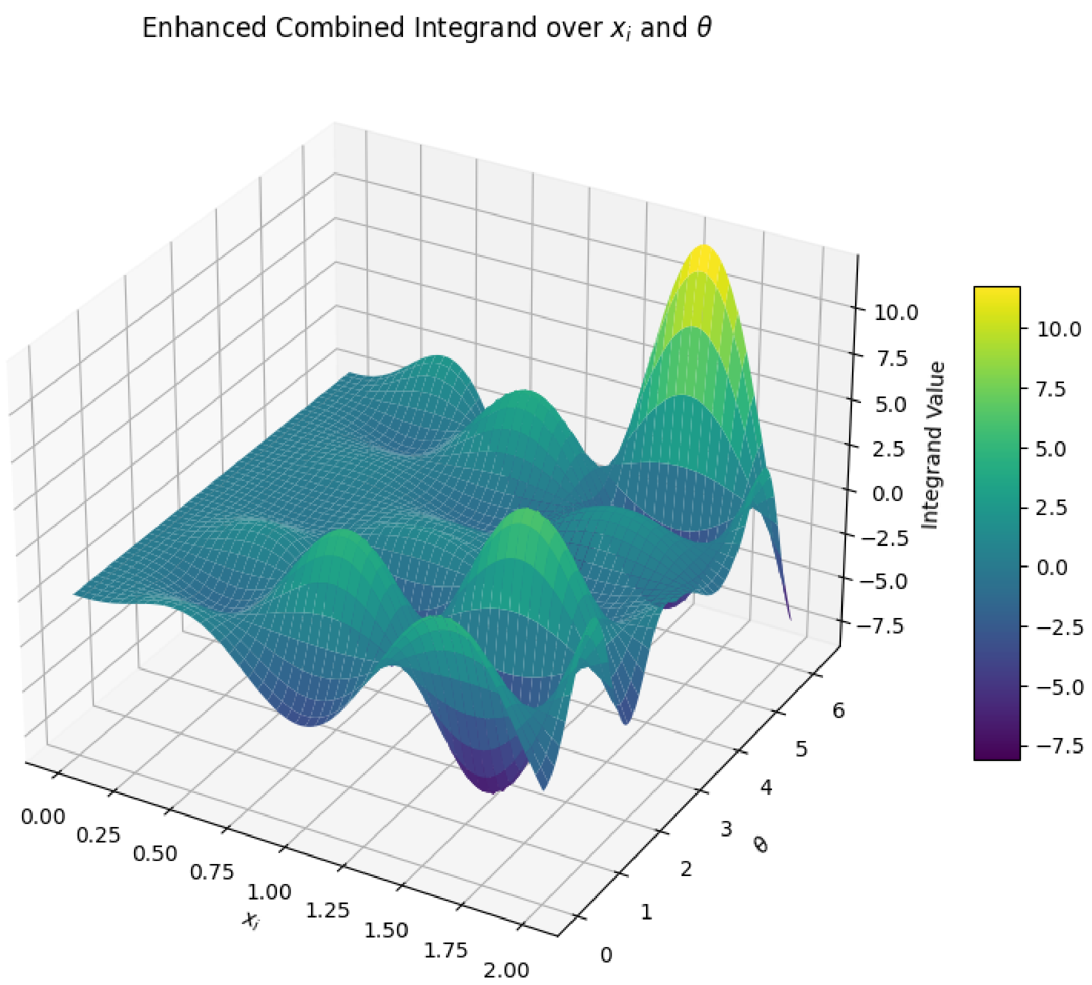
Note: The specific plot would be generated using the computational code provided and is indicative of the enhanced integrand's behavior.

Conclusion

The mathematical adaptations introduced in the "Enhanced Visualization" section involved augmenting the original integrand functions with exponential decay and trigonometric modulation. These enhancements allowed us to model more complex physical behaviors and provided a richer visualization of the tensor field's properties. The equations presented define these enhancements and demonstrate how they modify the integrand to achieve the desired analytical and computational effects.

```python
    # enhanced_mina_visualization.py

import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import quad

def visualize_enhanced_integrand(a_i_val, delta_a_i_val, alpha_i_val, n_val, k_i_val, beta_val, gamma_val, delta_val):
    # Define the numerical integrand functions
    def integrand_x_num(x_i):
        return k_i_val * (n_val * alpha_i_val + 1) * x_i ** (n_val * alpha_i_val) * np.exp(-beta_val * x_i) * np.sin(gamma_val * x_i) * (a_i_val + delta_a_i_val)

    def M_theta_num(theta):
        # Avoid division by zero
        theta_safe = np.where(theta == 0, 1e-6, theta)
        f1 = np.arcsin(np.sin(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
        f2 = np.arcsin(np.cos(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
        return (f1 + f2) * np.cos(delta_val * theta)

    # Generate data for visualization
    x_i_vals = np.linspace(0, a_i_val + delta_a_i_val, 300)
    theta_vals = np.linspace(1e-6, 2 * np.pi, 300)
    X_i, Theta = np.meshgrid(x_i_vals, theta_vals)

    with np.errstate(invalid='ignore', divide='ignore'):
        Z = integrand_x_num(X_i) * M_theta_num(Theta)

    # Plotting the enhanced integrand as a 3D surface plot
    fig = plt.figure(figsize=(12, 8))
```

```
    ax = fig.add_subplot(111, projection='3d')
    surf = ax.plot_surface(X_i, Theta, Z, cmap='viridis', linewidth=0, antialiased=True)
    ax.set_xlabel('$x_i$')
    ax.set_ylabel(r'$\theta$')
    ax.set_zlabel(r'Integrand Value')
    ax.set_title('Enhanced Combined Integrand over $x_i$ and $\\theta$')
    fig.colorbar(surf, shrink=0.5, aspect=10)
    plt.show()

def compute_enhanced_M_theta_integral(delta_val):
    # Define the numerical integrand function M_theta_num
    def M_theta_num(theta):
        # Avoid division by zero
        theta_safe = np.where(theta == 0, 1e-6, theta)
        f1 = np.arcsin(np.sin(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
        f2 = np.arcsin(np.cos(theta)) + (np.pi / 2) * np.exp(-np.pi / (2 * theta_safe))
        return (f1 + f2) * np.cos(delta_val * theta)

    # Define the subintervals
    theta_intervals = [(1e-6, np.pi), (np.pi, 2 * np.pi)]

    total_integral = 0.0
    total_error = 0.0

    for interval in theta_intervals:
        result, error = quad(M_theta_num, interval[0], interval[1], limit=100)
        total_integral += result
        total_error += error

    return total_integral, total_error

def main():
    # Numerical values for parameters
    a_i_val = 1.0
    delta_a_i_val = 1.0
    alpha_i_val = 2.0
    n_val = 1.0
    phi_m_val = 1.0
    lambda_val = 1.0
    k_i_val = 1.0
    beta_val = 1.0  # Exponential decay rate
    gamma_val = 5.0  # Frequency for sine function in x_i
    delta_val = 3.0  # Frequency for cosine function in theta

    # Define the numerical integrand function for x_i
    def integrand_x_num(x_i):
        return k_i_val * (n_val * alpha_i_val + 1) * x_i ** (n_val * alpha_i_val) * np.exp(-beta_val * x_i) * np.sin(gamma_val * x_i) * (a_i_val + delta_a_i_val)

    # Compute the integral over x_i
    I_x_val, _ = quad(integrand_x_num, 0, a_i_val + delta_a_i_val, limit=1000)

    # Compute the integral over theta using interval splitting
    I_theta_val, I_theta_error = compute_enhanced_M_theta_integral(delta_val)

    # Compute the total expression
    Expression_val = (1 / (2 * np.pi * lambda_val)) * phi_m_val * I_x_val * I_theta_val

    # Print the computed values
    print("I_x_val =", I_x_val)
    print("I_theta_val =", I_theta_val)
    print("I_theta_error =", I_theta_error)
    print("Expression_val =", Expression_val)

    # Visualize the enhanced combined integrand
    visualize_enhanced_integrand(
        a_i_val=a_i_val,
        delta_a_i_val=delta_a_i_val,
        alpha_i_val=alpha_i_val,
        n_val=n_val,
        k_i_val=k_i_val,
        beta_val=beta_val,
        gamma_val=gamma_val,
        delta_val=delta_val
    )

if __name__ == "__main__":
    main()
```



References

[1] R. Abraham, J. E. Marsden, and T. Ratiu, *Manifolds, Tensor Analysis, and Applications*, 2nd ed., ser. Applied Mathematical Sciences. Springer-Verlag, 1988.

[2] T. Frankel, *The Geometry of Physics: An Introduction*, 3rd ed. Cambridge University Press, 2011.

[3] G. B. Arfken and H. J. Weber, *Mathematical Methods for Physicists*, 7th ed. Academic Press, 2012.

[4] A. Meurer, C. P. Smith, M. Paprocki, *et al.*, "SymPy: symbolic computing in Python," *PeerJ Computer Science*, vol. 3, p. e103, 2017.

[5] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[6] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[7] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[9] C. D. Hansen and C. R. Johnson, *The Visualization Handbook*. Academic Press, 2005.

[10] R. S. Laramee, H. Hauser, L. Zöckler, and H. Doleisch, "Interactive visual analysis of a tensor field," in *Proceedings of the 14th IEEE Visualization 2004 Conference*, 2004, pp. 27–34.

[11] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, 7th ed. McGraw-Hill Education, 2015.

[12] P. M. Morse and H. Feshbach, *Methods of Theoretical Physics*. McGraw-Hill, 1953.

[13] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th ed. Brooks/Cole, Cengage Learning, 2011.

[14] G. van Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009.

[15] C. Feuersaenger, *PGFPlots - A LaTeX Package to Create Normal and Logarithmic Plots in Two and Three Dimensions*, 2019.

[16] M. Voss, *The `mathtools` package*, 2017.

‘‘‘