Article

Not peer-reviewed version

Differential Expressions and Their Applications

Taha Arkian *

Posted Date: 9 July 2025

doi: 10.20944/preprints202507.0812.v1

Keywords: Differential expressions; learning; algebra

 doi:10.20944/preprints202507.0812.v1

Article

Differential Expressions and Their Applications

Taha Arkian

Independent Researcher; tahaarkian1@gmail.com

Abstract

This paper develops a symbolic framework for differential expressions, treating them as independent entities rather than derivatives of known functions. By introducing nodes and maps, we model the flow and redistribution of differential terms in space, enabling dynamic systems of computation and geometry. Expansion matrices are defined to capture local transformation behavior, and algebraic potential is introduced as a measure of expression change across differential spaces. Applications include modified line integrals, Green's Theorem under differential control, and image processing on discrete grids using learned coefficient maps.

Keywords: differential expressions; learning; algebra

---

1. Introduction

This article presents a formal framework to treat differential expressions as scalable algebraic structures, allowing transition from infinitesimal forms to integrated, finite functions. The central idea is to treat differential operators as algebraic objects and scale them through substitution and integration. This concept is inspired by expressions like:

$$dx^5 + dy^3 = 1 \tag{1}$$

and explores how constraints like $\frac{dy}{dx} = -\frac{x}{y}$ allow us to isolate differentials such as dx and reconstruct functions through integration. These ideas extend to differential nodes and maps, allowing increment simulation. Applications for these expressions are given with practical examples in deep learning, cryptography and algorithms.

2. Differential Operators as Algebraic Objects

We define a class of differential operators $\delta^n x$ such that:

$$\int\int \delta^2 x = x \tag{2}$$

Hence, $\delta^n x$ denotes a symbolic representation of an n-th order differential, such that $\delta x = dx$, $\delta^2 x = d^2 x$, and so on, emphasizing algebraic manipulation over analytic formulation. Other differential operators are discussed in other sections.

3. Scaling Rule and Reduction Process

We define a scalable differential expression as:

$$S(x, y, dx, dy) = P(dx, dy) = 1 \tag{3}$$

where P is a polynomial in dx and dy, with possible dependencies on x and y.

Introducing a slope constraint (e.g., $\frac{dy}{dx} = f(x, y)$), we substitute $dy = f(x, y)dx$ to reduce the expression to isolate for dx:

$$P(dx, f(x, y)dx) \tag{4}$$

 doi:10.20944/preprints202507.0812.v1

4. Integration and Reconstruction

Once $dx = \phi(x, y)$ is obtained, we integrate to construct a finite function:

$$x = \int \phi(x, y(x))dx \tag{5}$$

Or alternatively:

$$y = \int f(x, y)dx \tag{6}$$

depending on the variable of integration and substitution.

5. Framework Definition

We define a **Scaling Differential System** as a triple:

$$\Sigma = (E(x, y, dx, dy),\ \theta(x, y),\ \mathcal{I}) \tag{7}$$

where:

- $E(x, y, dx, dy) = 1$: a polynomial differential expression
- $\theta(x, y) = \frac{dy}{dx}$: a slope constraint
- $\mathcal{I}$: an integration strategy (w.r.t. x, y, or along a parametrized curve)

The reduction proceeds by:

1. Substituting $dy = \theta(x, y)dx$ into E
2. Solving for $dx = \phi(x, y)$
3. Integrating $dx = \phi(x, y)$ to find x or another variable

6. Worked Example

Given the differential expression:

$$dx^2 + dy = 1$$

with the slope:

$$\frac{dy}{dx} = x$$

we aim to reduce this expression by substitution and isolate dx as a function of x, using symbolic scaling.

From the given slope, we write:

$$dy = x \cdot dx$$

Substituting into the original expression:

$$dx^2 + x \cdot dx = 1$$

$$dx^2 + x \cdot dx = 1 \quad \Rightarrow \quad dx(dx + x) = 1$$

This is equivalent to solving a quadratic equation in dx:

$$dx^2 + x \cdot dx - 1 = 0$$

Using the quadratic formula:

$$dx = \frac{-x \pm \sqrt{x^2 + 4}}{2}$$

 doi:10.20944/preprints202507.0812.v1

Since we are interested in the positive differential step ($dx > 0$), we take the positive root:

$$dx = \frac{-x + \sqrt{x^2 + 4}}{2}$$

If desired, one can integrate dx to get the scaled value of x:

$$x = \int \frac{-x + \sqrt{x^2 + 4}}{2}\, dx$$

This integral may be evaluated numerically or further manipulated depending on the context.

7. Different Integration Strategy

Solve the differential equation:

$$dx^2 + dy = 1$$

subject to the constraint:

$$\frac{dy}{dx} = x$$

We will apply an alternate integration strategy using division and the inverse scaling operator, $\mathcal{S}_{dx}^{-1}$, with the following definition:

Definition (Inverse Scaling): It is differentiation with respect to the specified variable on both sides of the equation. If the operand is not a differential in dx, it remains unchanged:

$$\mathcal{S}_{dx}^{-1}[1] = 1$$

$$\frac{dx^2}{dx} + \frac{dy}{dx} = \frac{1}{dx} \quad \Rightarrow \quad dx + x = \frac{1}{dx}$$

since $\frac{dy}{dx} = x$.

$$dx^2 + x \cdot dx = 1$$

Apply $\mathcal{S}_{dx}^{-1}$ to both sides:

$$\mathcal{S}_{dx}^{-1}[dx^2 + x \cdot dx] = \mathcal{S}_{dx}^{-1}[1]$$

Compute each term:

$$\mathcal{S}_{dx}^{-1}[dx^2] = 2dx, \quad \mathcal{S}_{dx}^{-1}[x \cdot dx] = x, \quad \mathcal{S}_{dx}^{-1}[1] = 1$$

So:

$$2dx + x = 1$$

$$2dx = 1 - x \quad \Rightarrow \quad dx = \frac{1 - x}{2}$$

Comparison with Standard Strategy

The original strategy using substitution gave:

$$dx = \frac{-x + \sqrt{x^2 + 4}}{2}$$

This shows that different integration strategies lead to different solutions, each internally consistent, illustrating the flexibility of the scaling method framework.

 doi:10.20944/preprints202507.0812.v1

8. Comparing Expressions, Proxy Functions, Trajectory Prediction

We compare the differential expressions:

$$(1)\ dx^2 + dy = 1 \qquad \text{and} \qquad (2)\ dy^2 + dx = 1$$

by solving each for dx, then integrating over the interval $x \in [0, 1]$ to compare how much horizontal change x accumulates under each equation.

Constraint: Slope Condition

We impose the following slope condition:

$$\frac{dy}{dx} = -\frac{x}{y} \Rightarrow dy = \left(-\frac{x}{y}\right)dx$$

Proxy Function Assumption

Since both expressions include the variable y, we must evaluate expressions like $\frac{x}{y}$ during integration. To handle this without solving the full coupled system, we introduce a **proxy function**:

$$y(x) = 1 + x$$

This simplifies all expressions to functions of x only.

What is a Proxy Function?

A **proxy function** is a chosen approximation for a variable whose exact functional form is unknown or difficult to derive. It must:

- Be simple and differentiable
- Stay in a reasonable range over the domain
- Allow meaningful evaluation of related expressions

Here, $y = 1 + x$ ensures positivity and simple behavior over $[0, 1]$.

Expression 1: $dx^2 + dy = 1$

Substitute using the slope:

$$dx^2 + \left(-\frac{x}{y}\right)dx = 1 \Rightarrow dx^2 - \frac{x}{y}dx - 1 = 0$$

This is a quadratic in dx. Solving:

$$dx = \frac{\frac{x}{y} + \sqrt{\left(\frac{x}{y}\right)^2 + 4}}{2}$$

Substitute $y = 1 + x$:

$$dx(x) = \frac{\frac{x}{1+x} + \sqrt{\left(\frac{x}{1+x}\right)^2 + 4}}{2}$$

Expression 2: $dy^2 + dx = 1$

Substitute $dy = \left(-\frac{x}{y}\right)dx \Rightarrow dy^2 = \left(\frac{x^2}{y^2}\right)dx^2$. Then:

$$\frac{x^2}{y^2}dx^2 + dx = 1 \Rightarrow \frac{x^2}{y^2}dx^2 + dx - 1 = 0$$

 doi:10.20944/preprints202507.0812.v1

This is again a quadratic in dx. Solving:

$$dx = \frac{-1 + \sqrt{1 + 4 \cdot \frac{x^2}{y^2}}}{2 \cdot \frac{x^2}{y^2}}$$

Substitute $y = 1 + x$:

$$dx(x) = \frac{-1 + \sqrt{1 + 4 \cdot \left(\frac{x}{1+x}\right)^2}}{2 \cdot \left(\frac{x}{1+x}\right)^2}$$

Integration and Comparison

We numerically evaluate:

$$\int_0^1 dx(x)$$

for both expressions using the formulas above.

Results

- Expression 1: $\int_0^1 dx(x) \approx \boxed{1.167}$
- Expression 2: $\int_0^1 dx(x) \approx \boxed{0.912}$
- Difference: $\boxed{0.255}$

9. Trajectory Interpretation of Differential Expressions

We now interpret each differential expression as defining a **trajectory** in the (x, y)-plane. Rather than viewing dx and dy as static differentials, we treat them as *step sizes* that generate motion through space.

Differential Steps as Motion

A solved expression for dx represents an infinitesimal forward motion along the x-axis. Combined with the constraint:

$$\frac{dy}{dx} = -\frac{x}{y},$$

we can compute corresponding dy and hence update both coordinates:

$$x_{n+1} = x_n + dx, \quad y_{n+1} = y_n + dy.$$

Thus, iteratively applying a differential expression generates a **path** or **trajectory** through the plane.

Integration Builds a Trajectory

When we compute:

$$\int_0^1 dx(x)dx,$$

we accumulate the total horizontal displacement prescribed by the expression. This integration captures how far a particle governed by the expression would move in the x-direction over the interval $[0, 1]$.

Each differential expression thus defines a distinct path, depending on how it relates dx and dy at every point.

 doi:10.20944/preprints202507.0812.v1

Comparing Two Expressions via dx

To determine how similar two trajectories are, we compare their scaled differential steps:

$$\Delta x = \left| \int_0^1 dx_1(x)dx - \int_0^1 dx_2(x)dx \right|.$$

Interpretation:

- If $\Delta x \approx 0$, the two trajectories are similar — their motion is nearly identical along the path.
- If Δx is large, then the paths diverge in behavior and cannot substitute for each other.

This gives us a way to define a **local approximation metric** between differential expressions.

Application: Trajectory Simulation

This principle is useful in practical modeling:

- Suppose a real-world trajectory is governed by an unknown differential expression.
- We test candidate expressions, solve for dx, and compute the resulting integrated path.
- If the candidate expression yields a close enough trajectory, it can serve as a *surrogate model*.

This method enables the simulation and control of motion using approximations, even when the underlying system is only partially known.

10. Applications of Differential Expressions in Deep Learning

Differential expressions can be applied to the design of neural networks by treating each layer as a learned scaling transformation based on generalized differential relationships. In this setting, instead of classical linear or convolutional layers, we introduce a new family of *differential scaling layers*, where input-output behavior mimics the dynamics of expressions like:

$$dx^n + dy^m = 1 \quad \Rightarrow \quad dx = (1 - dy^m)^{1/n}$$

This approach treats layer activations as stepwise differential changes and allows for modeling trajectory-like learning in feature space.

Generalized Differential Scaling Network: Python Example

We present a PyTorch implementation of a generalized network architecture based on this idea. The key component is a `GeneralizedDifferentialScalingLayer` that computes dx from $dy = wx + b$ and integrates it to emulate scaled flow through a differential expression.

Key Concepts in Code

- **Differential Scaling Layer** computes a value of dx using a learned linear form of dy, and reconstructs dx using a differential constraint:

$$dx = (1 - (dy)^m)^{1/n}$$

- **Integration Step** applies a cumulative sum (`torch.cumsum`) to simulate scaled motion over dimensions, akin to integrating a differential equation.
- **Network Architecture** chains these scaling layers with standard feedforward layers, allowing for differential-style transformations between layers.
- **Target Function** is a complex nonlinear surface:

$$y = \log(|x_1 x_2| + 1) + 0.5x_1^2$$

The network learns to approximate this using only differential scaling logic.

 doi:10.20944/preprints202507.0812.v1

Code Listing

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import DataLoader, TensorDataset
import numpy as np
import matplotlib.pyplot as plt
# === Differential Scaling Layer ===
class GeneralizedDifferentialScalingLayer(nn.Module):
    def __init__(self, input_dim, n=2, m=3):
        super(GeneralizedDifferentialScalingLayer, self).__init__()
        self.weight = nn.Parameter(torch.randn(input_dim))
        self.bias = nn.Parameter(torch.zeros(input_dim))
        self.n = n
        self.m = m

    def forward(self, x):
        dy = x * self.weight + self.bias
        dy_term = dy ** self.m
        dx_powered = 1 - dy_term
        dx_powered = torch.clamp(dx_powered, min=1e-6)
        dx = dx_powered ** (1 / self.n)
        dx_integrated = torch.cumsum(dx, dim=1)
        return dx_integrated
# === Network Definition ===
class GeneralizedDifferentialScalingNet(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, n=2, m=3):
        super(GeneralizedDifferentialScalingNet, self).__init__()
        self.ds1 = GeneralizedDifferentialScalingLayer(input_dim, n, m)
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.ds2 = GeneralizedDifferentialScalingLayer(hidden_dim, n, m)
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = self.ds1(x)
        x = F.relu(self.fc1(x))
        x = self.ds2(x)
        x = self.fc2(x)
        return x
# === New Sample Dataset ===
np.random.seed(42)
x_np = np.random.uniform(-2, 2, (500, 2)).astype(np.float32)

# New nonlinear function: y = log(|x1*x2| + 1) + 0.5*x1^2
y_np = np.log(np.abs(x_np[:, 0] * x_np[:, 1]) + 1) + 0.5 * (x_np[:, 0] ** 2)
y_np = y_np.reshape(-1, 1).astype(np.float32)

x_tensor = torch.tensor(x_np)
y_tensor = torch.tensor(y_np)
```

 doi:10.20944/preprints202507.0812.v1


```
dataset = TensorDataset(x_tensor, y_tensor)
dataloader = DataLoader(dataset, batch_size=32, shuffle=True)

# === Initialize Model ===
model = GeneralizedDifferentialScalingNet(
    input_dim=2, hidden_dim=8, output_dim=1, n=3, m=2
)
loss_fn = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)

# === Training Loop ===
losses = []
for epoch in range(100):
    total_loss = 0
    for xb, yb in dataloader:
        pred = model(xb)
        loss = loss_fn(pred, yb)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    losses.append(total_loss / len(dataloader))
    if epoch % 10 == 0:
        print(f"Epoch {epoch} | Loss: {losses[-1]:.4f}")

# === Plot Loss ===
plt.plot(losses)
plt.title("Training Loss (New Dataset)")
plt.xlabel("Epoch")
plt.ylabel("MSE Loss")
plt.grid(True)
plt.show()

# === Test Example ===
test_input = torch.tensor([[1.0, -1.5]])
test_output = model(test_input).item()
test_true = np.log(abs(1.0 * -1.5) + 1) + 0.5 * (1.0**2)

print("Predicted output:", test_output)
print("True value:      ", test_true)
```

Training and Results

The network is trained on 500 points generated by the nonlinear function:

$$y = \log(|x_1 x_2| + 1) + 0.5x_1^2$$

A training loop minimizes mean squared error using the Adam optimizer.

Result: After 100 epochs, the network is able to approximate the function well, indicating that differential scaling can model nonlinear behavior.

 doi:10.20944/preprints202507.0812.v1

Interpretation

Each differential scaling layer treats transformations as stepwise changes satisfying an implicit expression. The result is a hybrid between deep learning and symbolic modeling.

Use Cases

- Learning systems governed by physical constraints or motion.
- Embedding known differential behaviors into neural networks.

11. Symbolic Operators for Differential Scaling

We introduce four symbolic operators that provide a formal language for manipulating differential expressions during scaling procedures. These operators allow us to express integration, differentiation, which are introduced as "scaling".

The Scaling Operator

The scaling operator $\mathcal{S}_v$ denotes symbolic integration or accumulation with respect to the variable or differential v. It acts on expressions to increase their scale.

- $\mathcal{S}_x[f]$: integrate or scale f with respect to x
- $\mathcal{S}_{dx}[f]$: symbolically integrate with respect to dx
- $\mathcal{S}_x^n[f]$: apply scaling n times (nested integration)

Importantly, if the operand is not a function of v, it is left unchanged:

$$\mathcal{S}_{dx}^{-1}[1] = 1 \quad \text{(not zero)}$$

This operator represents the pseudoinverse of scaling — that is, symbolic differentiation or unwrapping of accumulated structure with respect to v. It reduces the order of the differential or cancels out a previous integration.

- $\mathcal{S}_x^{-1}[x^2] = 2x$
- $\mathcal{S}_{dx}^{-1}[dx^2] = 2dx$

Note: This operator only affects terms containing the subscript variable. All other constants or terms remain unaffected.

Integration Strategy as Operator Composition

An **integration strategy** is defined as a composition of these operators used to simplify or solve a differential expression. For example:

$$\mathcal{S}_{dx}^{-1}\left[dx^2 + xdx\right] = 2dx + x \quad \Rightarrow \quad dx = \frac{1-x}{2}$$

Different integration strategies (e.g., solving directly for dx, substituting dy/dx, or reducing via inverse scaling) may lead to different but related results. These operators form the symbolic toolkit of differential scaling theory, enabling flexible manipulation of complex expressions across physics, simulation, and learning systems.

General Scaling Operator

We define the **general scaling operator** $\mathcal{S}_g$ as the composition of scaling operations with respect to all variables and differentials involved:

$$\mathcal{S}_g := \mathcal{S}_{dx} \circ \mathcal{S}_{dy} \circ \mathcal{S}_x \circ \mathcal{S}_y$$

 doi:10.20944/preprints202507.0812.v1

Inverse Scaling (Reduction)

The pseudoinverse of a scaling operation is called **reduction**. For any operator $\mathcal{S}_v$, we define:

$$\text{Reduction:} \quad \mathcal{S}_v^{-1}(f) \quad \text{is the inverse operation of } \mathcal{S}_v$$

Standard Scaling Algorithm

To apply the **standard scaling algorithm**, follow these steps:

1. Begin with general scaling or reduction using $\mathcal{S}_g$ or its inverse.
2. End with specific scaling with respect to a particular variable or differential.

Delta and Epsilon Operators

We introduce two special symbolic operators:

- **Delta** δ: an operator such that scaling it n times yields the base variable:

$$\mathcal{S}_x^n[\delta^n] = x$$

For standalone terms:

$$\delta y = dx \quad \text{and} \quad \delta y = dy$$

- **Epsilon** ε: an operator such that:

$$\mathcal{S}_x[\varepsilon(x)] = x'$$

These operators **disappear** only under general scaling $\mathcal{S}_g$.

Important Scaling Rules

- Scaling a differential restores the original variable:

$$\mathcal{S}_{dx}(dx) = x, \qquad \mathcal{S}_{dy}(dy) = y$$

- Inverse scaling a variable yields its differential:

$$\mathcal{S}_x^{-1}(x) = dx, \qquad \mathcal{S}_y^{-1}(y) = dy$$

- Scaling/reducing operations do not eliminate differentials (i.e., they cannot reduce to zero order).
- Differentials can be reduced to the first degree, then an other scaling operator can transform that differential into a a variable that is not a differential like x or y.
- The following may not always be true:

$$\mathcal{S}^{-1}(\mathcal{S}(f)) = f$$

12. Theory of Nodes in Differential Expressions

Motivation

Traditional differential expressions, such as:

$$dx^2 + dy = 1,$$

are evaluated algebraically as a whole. However, in our framework, we introduce a modular decomposition into programmable units called **nodes**.

 doi:10.20944/preprints202507.0812.v1

Definition of a Node

A **node** is a symbolic container for a differential term. It is written in the notation:

$$[\text{term}]$$

Each node consists of:

- A *term* involving variables and/or differentials
- An *accumulator* that stores a value over time
- A *rate of increment*, which is the evaluated term at each time step

Formal Definition:

Let $T(x, y, dx, dy, \ldots)$ be a differential term. Then the node

$$[T] \longrightarrow A(t)$$

evolves according to the recurrence:

$$A(t+1) = A(t) + T(x(t), y(t), \ldots)$$

where t represents time steps.

Decomposing a Differential Expression into Nodes

Any differential expression can be decomposed into nodes. For example:

$$dx^2 + dy = 1 \quad \Rightarrow \quad [dx^2] + [dy] = 1$$

Here, each term becomes an independent node with its own program of accumulation.

The equality to 1 on the right-hand side is symbolic. It is not required for the numerical values in each node to sum to 1; rather, the equality indicates a balanced structural form in the symbolic algebra.

Increment Logic and Node Behavior

Nodes serve as independent computational agents:

- The internal term acts as the **acceleration**.
- The node accumulates its **value** over time using its term.

This mirrors physical systems:

$$\text{Velocity} = \frac{dx}{dt}, \quad \text{Displacement} = \int \text{Velocity}\, dt$$

where the accumulation step is symbolic rather than numeric.

Purpose and Utility

Nodes enable:

- Modular simulation of expressions
- Symbolic transformations via redistribution
- Analysis of flow between terms
- Construction of programmable cryptographic systems

Nodes give us a new algebra of expressions, allowing dynamic symbolic computation across steps of time and transformations in structure. While nodes represent independent units of symbolic accumulation, many systems require the ability to **redistribute terms** between nodes.

To achieve this, we introduce the concept of a **map**.

 doi:10.20944/preprints202507.0812.v1

Definition of a Map

A **map** is a directed graph that defines transitions of terms from one node to another over time. The purpose of a map is to model how the components of a differential expression shift their influence across the system.

Notation:

A map is denoted as a set of ordered pairs:

$$\mathcal{M} = \{\text{Node}_i \to \text{Node}_j, \ldots\}$$

Each element indicates that the term in Node_i is to be transferred to Node_j after a specific time period or event.

Behavior of Maps

Maps allow for:

- **Time-based redistribution**: At predefined steps, a node's term can be moved to another node.
- **Cumulative combination**: The target node accumulates both its original content and the transferred term.
- **Dynamic simulation**: The system evolves not just by incrementing values but also by **changing its structure**.

Interpretation and Execution

Let the system consist of n nodes:

$$[T_1] + [T_2] + \cdots + [T_n] = 1$$

Let the map be:

$$\mathcal{M} = \{\text{Node}_1 \to \text{Node}_2, \text{Node}_2 \to \text{Node}_3\}$$

The simulation proceeds as follows:

1. At each time step, every node computes its increment: $A_i(t+1) = A_i(t) + T_i(x(t), y(t), \ldots)$
2. After a predefined number of iterations, apply the map: move the term T_i from its current node to its mapped destination
3. Continue the iteration with the updated node assignments

Purpose of Maps

Maps introduce structure and control over the flow of symbolic computation. Their applications include:

- Modeling of dynamic systems with changing term contributions
- Symbolic encoding of flow and logic
- Simulating transitions in computational or physical systems
- Creating cryptographic keys through structural obfuscation

Trajectory Simulation

Each node acts like an independent dynamical component. By tracking the value of dx per node, we can simulate how multi-part systems behave over time—e.g., modeling physical motion.

Cryptographic Potential

Because node systems and mappings are programmable and evolve over time based on internal expressions, they could be used to create **obfuscated**, **dynamic** encoding schemes. These schemes

 doi:10.20944/preprints202507.0812.v1

evolve based on initial differential structures and maps and may be difficult to reverse-engineer without knowing the exact proxy differential and transition logic.

13. Function-Augmented Nodes in Differential Systems

Definition

We introduce a new type of node in the differential expression framework, referred to as a **function-augmented node**. These are denoted using the bracket notation with a superscript indicating the applied function. For example, the node

$$[f]\left[dx^2 + y\right]$$

indicates that the function f is first applied to the entire expression inside the node before solving or scaling. A typical case is squaring:

$$[\cdot^2]\left[dx^2 + y\right] \Rightarrow dx^4 + y^2 + 2y \cdot dx^2$$

where the result comes from expanding $(dx^2 + y)^2$ symbolically.

Notation

We define a function-augmented node as:

$$[f](E) := f(E)$$

where E is a differential expression involving dx, dy, and possibly x, y, and the function $f : \mathbb{R} \to \mathbb{R}$ is applied algebraically.

Computation Rule

Let E be a symbolic differential expression. Then:

1. Apply the function f symbolically to E
2. Distribute terms if possible (e.g., via binomial expansion)
3. Scale the resulting differential expression to isolate dx
4. Solve for dx

14. Worked Example: Differential Expression with Nodes, Cryptography Application

We are given the differential expression:

$$\left[x^2dx + dy^2x \; + \; \varepsilon\Big(x^2 + 2xdy + dy\Big)\right] + \left[\delta^2\Big(x^2 - y^2 + dx\Big) + \delta^2 y \cdot \; \varepsilon^2\left(\frac{1}{dx}\right)\right] + \left[\delta^2 x + x + y\right] = 1$$

with the map $1 \to 2 \to 3 \to 1$. This expression, along with iterations and a proxy function, is taken as input. There will be a corresponding output when integrating to get the x values. These values represent the encoded expression. When this expression is entered, it will give the same results, and the computer will know that it is the user it is looking for.

Node Structure (Fixed Terms)

- Node 1: fixed term dy^2x
- Node 2: fixed term $\delta^2 y \cdot \varepsilon^2(1/dx)$
- Node 3: fixed term $\delta^2 x$

 doi:10.20944/preprints202507.0812.v1

We define:

$$y = x^2 - 1, \quad x = t, \quad \text{and let } x_s = \int dx \text{ over the time interval}$$

We compute for $t_1 = 2$, $t_2 = 3$, $t_3 = 4$, $t_4 = 5$.

Iteration 1

Node 1: $x^2 dx + dy^2 x + \varepsilon(x^2 + 2xdy + dy) = 1$

Apply general scaling:

$$\mathcal{S}_g[\cdot] \Rightarrow \frac{x^4}{3} + \frac{dy^3}{3} \cdot \frac{x^2}{2} + 2x + 2dx \cdot \delta^2 y + \delta^2 y = 1$$

Apply $\mathcal{S}_{dy}^2$:

$$\frac{x^4}{3} + \frac{dy^5}{60} \cdot \frac{x^2}{2} + 2x + 2dxy + y = 1$$

Apply $\mathcal{S}_{dy}^{-4}$:

$$\frac{x^4}{3} + dy \cdot x^2 + 2x + 2dxy + y = 1$$

Apply $\mathcal{S}_{dy}$:

$$\frac{x^4}{3} + yx^2 + 2x + 2dxy + y = 1$$

Isolate dx:

$$dx = \frac{1 - \frac{x^4}{3} - yx^2 - 2x - y}{2y}$$

Using $y = x^2 - 1$, we get:

$$x_s = \int_2^3 \frac{1 - \frac{x^4}{3} - (x^2 - 1)x^2 - 2x - (x^2 - 1)}{2(x^2 - 1)} dx$$

Node 2: $\delta^2(x^2 - y^2 + dx) + \delta^2 y \cdot \varepsilon^2(1/dx) = 1$

Apply $\mathcal{S}_g$ twice:

$$x^2 - y^2 + dx + \frac{2y}{dx^3} = 1 \Rightarrow dx + \frac{2y}{dx^3} = 1 + y^2 - x^2$$

Apply $\mathcal{S}_{dx}$:

$$x + \frac{y}{dx^2} = 1 + y^2 - x^2 \Rightarrow dx = \sqrt{\frac{y}{1 + y^2 - x^2 - x}}$$

Substitute $y = x^2 - 1$, compute:

$$x_s = \int_2^3 \sqrt{\frac{x^2 - 1}{1 + (x^2 - 1)^2 - x^2 - x}} dx$$

Node 3: $\delta^2 x + x + y = 1$

Apply $\mathcal{S}_g$:

$$dx + \frac{x^2}{2} + \frac{y^2}{2} = 1 \Rightarrow dx = 1 - \frac{x^2}{2} - \frac{y^2}{2}$$

Substitute $y = x^2 - 1$, compute:

$$x_s = \int_2^3 \left(1 - \frac{x^2}{2} - \frac{(x^2 - 1)^2}{2}\right) dx$$

 doi:10.20944/preprints202507.0812.v1

Iteration 2

Node 1: $x + y + dy^2 x = 1$

Apply $\mathcal{S}_g^{-1}$:

$$dx + dy + 2dy \cdot dx = 1$$

Apply $\mathcal{S}_{dy}$:

$$dx + y + 2y \cdot dx = 1 \Rightarrow dx = \frac{1-y}{2y+1}$$

Substitute $y = x^2 - 1$, compute:

$$x_s = \int_3^4 \frac{1-(x^2-1)}{2(x^2-1)+1}\,dx = \int_1^2 \frac{2-x^2}{2x^2-1}\,dx$$

Node 2: $x^2 dx + \varepsilon(x^2 + 2xdy + dy) + \delta^2 y \cdot \varepsilon^2 (1/dx) = 1$

Apply $\mathcal{S}_g$ twice and then $\mathcal{S}_{dy}$:

$$\frac{x^5}{15} + x^2 + 2xy + y + \frac{2y}{dx^3} = 1 \Rightarrow dx = \sqrt[3]{\frac{2y}{1 - y - 2xy - x^2 - \frac{x^5}{15}}}$$

Substitute $y = x^2 - 1$, compute:

$$x_s = \int_3^4 \sqrt[3]{\frac{2(x^2-1)}{1-(x^2-1)-2x(x^2-1)-x^2-\frac{x^5}{15}}}\,dx$$

Node 3: $\delta^2(x^2 - y^2 + dx) + \delta^2 x = 1$

Apply $\mathcal{S}_g$ twice:

$$x^2 - y^2 + dx + x = 1 \Rightarrow dx = 1 - x - x^2 + y^2$$

Substitute $y = x^2 - 1 \Rightarrow y^2 = (x^2-1)^2$, compute:

$$x_s = \int_3^4 \left(1 - x - x^2 + (x^2-1)^2\right) dx$$

Iteration 3

Node 1: $\delta^2(x^2 - y^2 + dx) + dy^2 x = 1$

Apply $\mathcal{S}_g$ twice, then apply $\mathcal{S}_{dy}^{-1}$ three times, and finally apply $\mathcal{S}_{dy}$:

$$x^2 - y^2 + dx + \frac{yx^3}{6} = 1 \Rightarrow dx = 1 - \frac{yx^3}{6} + y^2 - x^2$$

Substitute proxy function $y = x^2 - 1$:

$$dx = 1 - \frac{(x^2-1)x^3}{6} + (x^2-1)^2 - x^2$$

Then:

$$x_s = \int_4^5 \left[1 - \frac{(x^2-1)x^3}{6} + (x^2-1)^2 - x^2\right] dx$$

Node 2: $x + y + \delta^2 y \cdot \varepsilon^2\left(\frac{1}{dx}\right) = 1$

Apply $\mathcal{S}_g$ twice:

$$\frac{x^3}{6} + \frac{y^3}{6} + \frac{2y}{dx^3} = 1 \Rightarrow dx = \sqrt[3]{\frac{2y}{1 - \frac{x^3}{6} - \frac{y^3}{6}}}$$

 doi:10.20944/preprints202507.0812.v1

Substitute $y = x^2 - 1$, compute:

$$x_s = \int_4^5 \sqrt[3]{\frac{2(x^2-1)}{1 - \frac{x^3}{6} - \frac{(x^2-1)^3}{6}}} dx$$

Node 3: $x^2dx + \varepsilon(x^2 + 2xdy + dy) + \delta^2 x = 1$

Apply $\mathcal{S}_g$ twice, then $\mathcal{S}_d y$ then $\mathcal{S}_x^{-1}$:

$$\frac{x^5}{15} + x^2 + 2xy + y + x = 1 \Rightarrow \frac{x^4}{3} + 2x + 2dxy + y + dx = 1 \Rightarrow dx = \frac{1 - \frac{x^4}{3} - 2x - y}{2y + 1}$$

Substitute $y = x^2 - 1$, compute:

$$x_s = \int_4^5 \frac{1 - \frac{x^4}{3} - 2x - (x^2 - 1)}{2(x^2 - 1) + 1} dx$$

Results Table

After integrating to get each x, the outputs, we have the following results. As stated in the beginning, this is the encoded differential expression. Giving the differential expression and its proxy function as an input, if it matches the output, we know that it is the legitimate user.

Table 1. Computed values of x at each time step.

Time Step	Node 1	Node 2	Node 3
Iteration 1: from 2 to 3	-5.31172599738	0.547088100892	-17.4333333333
Iteration 2: from 3 to 4	-0.43470023222	-0.549139623878	117.7
Iteration 3: from 4 to 5	56.325	-0.319750173354	-4.16479804869

15. Example with Augmented Nodes: Discrete Movement Systems

In this example, we explore how differential expressions and scaling operations can be used to simulate discrete movement by decomposing motion into segments, each governed by an **augmented node**.

Original Differential Expression

We start with the base expression:

$$dy + dx^2 = 1$$

Apply general inverse scaling $\mathcal{S}_g^{-1}$:

$$y + 2dx = 1 \quad \Rightarrow \quad dx = \frac{1 - y}{2}$$

Let the proxy function be:

$$y = x$$

Augmented Node

We now square the original expression:

$$(dy + dx^2)^2 = 1 \quad \Rightarrow \quad dy^2 + dx^4 + 2dy \cdot dx^2 = 1$$

Apply $\mathcal{S}_g^{-1}$, then $\mathcal{S}_g$:

$$2y + dx^4 + 2dy \cdot dx^2 = 1$$

 doi:10.20944/preprints202507.0812.v1

Apply $\mathcal{S}_{dy}$:

$$2y + dx^4 + 2y \cdot dx^2 = 1$$

Isolate dx:

$$dx^4 + 2y \cdot dx^2 = 1 - 2y$$

Apply $\mathcal{S}_{dx}^{-1}$, then $\mathcal{S}_{dx}$:

$$dx^4 + 4y \cdot x = 1 - 2y \Rightarrow dx = \sqrt[4]{1 - 2y - 4yx}$$

Let the proxy function remain $y = x$. Thus:

$$dx = \sqrt[4]{1 - 2x - 4x^2}$$

Discrete Movement Integrals

We divide the movement into two discrete nodes (segments):

- **Node 1:** Integrate from $x = 0$ to $x = 0.15$
- **Node 2:** Integrate from $x = 0.15$ to $x = 0.3$

$$\Delta x_1 = \int_0^{0.15} \frac{(1 - x)}{2} dx$$

$$\Delta x_2 = \int_{0.15}^{0.3} \sqrt[4]{1 - 2x - 4x^2}\, dx$$

Results Table

Node	Interval	Δx
1	$[0, 0.15]$	0.069375
2	$[0.15, 0.3]$	0.111193888892
Total Movement	$[0, 0.3]$	**0.180568888892**

Interpretation and Application

This process defines a **discrete movement system**, where each segment of motion is governed by a different transformed or augmented differential expression. The system allows simulation of motion in a stepwise fashion.

Applications:

- Modeling physical systems where only discrete measurements are possible (e.g., robotics with time-stepped control).
- Simulating discrete systems in theoretical physics, where movement as a continuous function is not available.
- Fitting real movement data by selecting appropriate **proxy functions** and **augmentations** to match known Δx values.

16. Variable Schemes, Contractions, and Expansion Matrices

Variable Schemes

A **variable scheme** is a formal transformation of variables where each variable is reassigned to another. For example:

$$x \to dy, \quad dy \to dx, \quad y \to x$$

is a valid variable scheme.

We define two special types of variable schemes:

- **General Contraction** $\mathcal{C}_g$: a scheme that maps all variables i to their differentials di.
- **General Extraction** $\mathcal{E}_g$: a scheme that maps all differentials di back to their base variables i.

 doi:10.20944/preprints202507.0812.v1

Applications to Physics

In physical systems, particularly in the context of differential velocities, we can define:

$$v = \mathcal{S}_g^{-1}(\mathcal{E}_g(m))$$

where m is mass. The operator $\mathcal{E}_g(m)$ represents the extraction of mass—i.e., converting mass to position—while $\mathcal{S}_g^{-1}$ extracts the differential of that quantity, giving us velocity as the differential of position. This is a potential and theoretical application to velocity in physics, and is only symbolic for now.

Rates of Contraction and Extraction

For a given differential expression F, we define:

$$\text{Rate of Contraction: } \frac{\mathcal{C}_g(F)}{F}$$
$$\text{Rate of Extraction: } \frac{\mathcal{E}_g(F)}{F}$$

Expansion Potential and Area of Reach

The **expansion potential** $T(F)$ of a differential expression is defined as:

$$T(F) = \frac{\mathcal{C}_g(F) \cdot \mathcal{E}_g(F)}{F^2}$$

This represents the normalized directional expansion potential of the expression.

We define the **potential difference** as:

$$P(F) = \mathcal{E}_g(F) - \mathcal{C}_g(F)$$

The **area of reach** is then defined as:

$$A(F) = P(F) \cdot T(F) = \frac{\mathcal{C}_g(F) \cdot \mathcal{E}_g(F)}{F^2} \cdot \left(\mathcal{E}_g(F) - \mathcal{C}_g(F)\right)$$

This quantity has geometric interpretation similar to the determinant of an expansion matrix.

Expansion Matrix

We define the **expansion matrix** $M(F)$ for a differential expression F as:

$$M(F) = \begin{bmatrix} \frac{\mathcal{C}_g(F)}{F} & \frac{\mathcal{E}_g(F)}{F} \\ -\frac{\mathcal{C}_g(F)^2}{F} & \frac{\mathcal{E}_g(F)^2}{F} \end{bmatrix}$$

This matrix captures both directional scaling and potential variation.

Compression Schemes

A **compression scheme** is a variable scheme in which all variables are mapped to a single differential (e.g., dx), such as:

$$x \to dx, \quad y \to dx, \quad dy \to dx$$

The *step differential*, usually denoted dx, is the differential with respect to which the expression is ultimately solved. Depending on the context, dy may be used as the step differential instead.

Rate of Change of Differential Expressions

We define the **rate of change** of a differential expression F as the normalized difference between its general extraction and its general contraction. That is, we are interested in how much the expression

 doi:10.20944/preprints202507.0812.v1

shifts away from its differential form toward its algebraic form (or vice versa) relative to its total magnitude.

The **rate of change** $R(F)$ of the differential expression is defined as:

$$R(F) = \frac{\mathrm{Eg}(F) - \mathrm{Cg}(F)}{F}$$

This value represents the net directional shift of the expression: if $R(F) > 0$, the expression is biased toward algebraic expansion; if $R(F) < 0$, it is biased toward differential contraction.

This operator can serve as an important metric in systems where we compare algebraic behavior versus dynamic behavior over time, especially in simulations or learning models involving transitions between symbolic states and differential states.

Combining Differential Expressions

To combine multiple differential expressions within the same system, we multiply their expansion matrices:

$$M = M_1 \cdot M_2$$

This allows us to compound transformations and determine cumulative effects of successive expressions.

Worked Example: Expansion Matrix and Flow from a Differential Expression

Let us consider the differential expression:

$$F : dx^2 + dy = 1$$

Step 1: Solving for the step differential dx**:** We apply the inverse general scaling $\mathcal{S}_g^{-1}$, followed by $\mathcal{S}_y$ twice:

$$2dx + \delta^2 y = 1 \quad \Rightarrow \quad 2dx + y = 1 \quad \Rightarrow \quad dx = \frac{1-y}{2}$$

Step 2: Apply General Extraction $\mathcal{E}_g(F)$ By replacing all differentials with their original variables:

$$\mathcal{E}_g(F) : x^2 + y = 1 \Rightarrow x = \sqrt{1-y}, \quad dx = \frac{-1}{2\sqrt{1-y}}$$

Step 3: Apply General Contraction $\mathcal{C}_g(F)$ Since all terms in F are already in differential form:

$$\mathcal{C}_g(F) = F : dx^2 + dy$$

Step 4: Compute Expansion Potential $T(F)$ The expansion potential is:

$$T(F) = \frac{\mathcal{C}_g(F) \cdot \mathcal{E}_g(F)}{F^2} = \frac{\mathcal{E}_g(F)}{F}$$

Substituting:

$$T(F) = \left(\frac{-1}{2\sqrt{1-y}}\right) \cdot \left(\frac{2}{1-y}\right) = \frac{-1}{(1-y)^{3/2}}$$

 doi:10.20944/preprints202507.0812.v1

Step 5: Compute Matrix Entries

$$\begin{aligned}
\frac{\mathcal{C}_g(F)}{F} &= 1 \\
\frac{\mathcal{E}_g(F)}{F} &= \frac{-1}{2\sqrt{1-y}} \cdot \frac{2}{1-y} = -\frac{1}{(1-y)^{3/2}} \\
\frac{-\mathcal{C}_g(F)^2}{F} &= \frac{y-1}{2} \\
\frac{\mathcal{E}_g(F)^2}{F} &= \left(\frac{-1}{2\sqrt{1-y}}\right)^2 \cdot \frac{1}{F} = \frac{1}{4(1-y)} \cdot \frac{1}{F} = \frac{1}{2(1-y)^2}
\end{aligned}$$

Step 6: Expansion Matrix and Flow We define the expansion matrix:

$$M = \begin{bmatrix} 1 & -\frac{1}{(1-y)^{3/2}} \\ \frac{y-1}{2} & \frac{1}{2(1-y)^2} \end{bmatrix}$$

We define the **Flow** of the differential expression as the sum of the matrix entries:

$$\text{Flow} = 1 - \frac{1}{(1-y)^{3/2}} + \frac{y-1}{2} + \frac{1}{2(1-y)^2}$$

Step 7: Self-Combination (Matrix Squaring) Squaring the matrix:

$$M^2 = \begin{bmatrix} 1 & -\frac{1}{(1-y)^{3/2}} \\ \frac{y-1}{2} & \frac{1}{2(1-y)^2} \end{bmatrix}^2$$

Compute:

$$M^2 = \begin{bmatrix} 1 - \frac{y-1}{2(1-y)^{3/2}} & -\frac{1}{(1-y)^{3/2}} - \frac{1}{2(1-y)^{7/2}} \\ \frac{y-1}{4(1-y)^2} + \frac{y-1}{2} & \frac{1}{4(1-y)^4} - \frac{y-1}{2(1-y)^{3/2}} \end{bmatrix}$$

This squared matrix represents the result of combining the differential expression with itself—useful for recursive differential systems, expansion compounding, or higher-order flow modeling.

17. Image Processing with Grids and Differential Expressions

Differential expressions can be used in image processing by interpreting images as 2D grids where traversal paths follow directional flows defined by expressions such as

$$dx^2 + dy = 1 \quad \Rightarrow \quad dx = \frac{1-y}{2}, \quad dy = 1 - 2x.$$

Given a discrete path on a grid (e.g., a diagonal from top-left to bottom-right), we aim to learn *coefficient maps* for each grid cell that scale these differential steps. These coefficients determine how strongly each direction (dx or dy) contributes to matching the actual path. This approach can later be generalized to image filters or learned spatial transformations.

Learning the Coefficients

Let a and b be coefficients applied to the respective dx and dy expressions:

$$dx = a \cdot \frac{1-y}{2}, \quad dy = b \cdot (1 - 2x).$$

 doi:10.20944/preprints202507.0812.v1

The objective is to find values of a and b that best transition a point (x, y) to the next point (x', y') in a given path. This is framed as an optimization problem, minimizing the squared error between predicted and target positions.

Differential Strategy Variants

Depending on the nature of the movement (e.g., slow horizontal growth or dominant vertical expansion), we may define the effective step as:

$$\text{Advance step} = \min\left(dx, \frac{1}{dx}\right),$$

to normalize large or small movements depending on the dynamic scaling.

Example: Learning Coefficients on a Grid Path

Below is a Python implementation that illustrates this approach for a simple 5x5 grid, each grid being of unit 2 in both directions, and a diagonal path.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
import pandas as pd

# === Setup ===
grid_size = 5
unit = 2  # Each step is worth 2 units
path = [(i * unit, i * unit) for i in range(grid_size)]  # Diagonal path

# === Differential expression-derived functions ===
def dx_expr(x, y, a):
    return a * ((1 - y) / 2)

def dy_expr(x, b):
    return b * (1 - 2 * x)

# === Optimization to match step from (x, y) to (x_target, y_target) ===
def optimize_step(x, y, x_target, y_target):
    def loss(params):
        a, b = params
        dx = dx_expr(x, y, a)
        dy = dy_expr(x, b)
        return (x + dx - x_target) ** 2 + (y + dy - y_target) ** 2
    result = minimize(loss, x0=[1.0, 1.0])
    return result.x if result.success else (np.nan, np.nan)

# === Coefficient maps ===
a_map = np.zeros((grid_size, grid_size))
b_map = np.zeros((grid_size, grid_size))
coeff_table = []

# === Process path ===
for i in range(len(path) - 1):
    x, y = path[i]
```

 doi:10.20944/preprints202507.0812.v1


```
    x_t, y_t = path[i + 1]
    gx, gy = int(x / unit), int(y / unit)
    a_opt, b_opt = optimize_step(x, y, x_t, y_t)
    a_map[gy, gx] = a_opt
    b_map[gy, gx] = b_opt
    coeff_table.append({'From': (x, y), 'To': (x_t, y_t), 'a': a_opt, 'b': b_opt})

# === Visualize path on 5x5 grid ===
grid = np.zeros((grid_size, grid_size))
for x, y in path:
    gx, gy = int(x / unit), int(y / unit)
    grid[gy, gx] = 1

plt.imshow(grid, cmap='Greens', origin='lower')
plt.title("Path through Scaled Grid")
plt.colorbar(label="Path (1=on path)")
plt.show()

# === Plot coefficient maps ===
fig, ax = plt.subplots(1, 2, figsize=(12, 5))
im1 = ax[0].imshow(a_map, cmap='viridis', origin='lower')
ax[0].set_title("Coefficient Map: a")
fig.colorbar(im1, ax=ax[0])
im2 = ax[1].imshow(b_map, cmap='plasma', origin='lower')
ax[1].set_title("Coefficient Map: b")
fig.colorbar(im2, ax=ax[1])
plt.show()

# === Display coefficient table ===
df = pd.DataFrame(coeff_table)
print(df)
```

Applications

This approach has the potential for use in:

- Adaptive image warping based on learned path behaviors.
- Directional filters derived from physically inspired differential rules.
- Learning spatial transformations using stored coefficient maps.

This method also allows for reuse: once a coefficient map is learned for a given differential expression and grid layout, it can be reused or fine-tuned on new paths, providing efficiency and interpretability.

18. Algebraic Potential of Algebraic Expressions in a Differential Space

In this section, we introduce the concept of **algebraic potential**, which measures the expected change of an algebraic expression $A(x, y)$ under motion governed by a differential expression.

Let dx and dy be the differential components obtained from solving a differential expression (or a node in a system). The **algebraic potential** ΔA of an algebraic expression A under this motion is defined as:

$$\Delta A = \frac{\partial A}{\partial x} \cdot dx + \frac{\partial A}{\partial y} \cdot dy$$

 doi:10.20944/preprints202507.0812.v1

This represents the directional change of A along a path determined by the differential expression, effectively embedding algebraic change within a dynamically defined geometric space.

Worked Example: Algebraic Potential in a Curved Differential Space

Let:

$$A(x, y) = x^2 + y$$

and let the governing differential expression be:

$$dy^2 + dx^4 + 2dy \cdot dx^2 = 1$$

Solving this differential expression gives:

$$dx = \sqrt[4]{1 - 2y - 4xy}, \quad dy = \frac{1 - 2y - 24x}{\frac{2}{3}x^3}$$

We compute:

$$\frac{\partial A}{\partial x} = 2x, \quad \frac{\partial A}{\partial y} = 1$$

Substituting at $x = \frac{1}{4}, y = \frac{1}{4}$

$$dx = \sqrt[4]{1 - 2\left(\tfrac{1}{4}\right) - 4\left(\tfrac{1}{4}\right)\left(\tfrac{1}{4}\right)} = \sqrt[4]{1 - \tfrac{1}{2} - \tfrac{1}{4}} = \sqrt[4]{\tfrac{1}{4}} = \tfrac{1}{\sqrt{2}} \approx 0.7071$$

$$dy = \frac{1 - 2\left(\frac{1}{4}\right) - 24\left(\frac{1}{4}\right)}{\frac{2}{3}\left(\frac{1}{4}\right)^3} = \frac{1 - \frac{1}{2} - 6}{\frac{2}{3} \cdot \frac{1}{64}} = \frac{-5.5}{\frac{1}{96}} = -528$$

We compute the partial derivatives:

$$\frac{\partial A}{\partial x} = 2x = \tfrac{1}{2}, \quad \frac{\partial A}{\partial y} = 1$$

Thus:

$$\Delta A = \tfrac{1}{2} \cdot \tfrac{1}{\sqrt{2}} + 1 \cdot (-528) = \frac{1}{2\sqrt{2}} - 528 \approx -527.646$$

Comparison: Algebraic Potential in a Simpler Space

Now take a simpler differential expression:

$$dx^2 + dy = 1 \Rightarrow dx = \frac{1 - y}{2}, \quad dy = 1 - 2x$$

At $x = \frac{1}{4}, y = \frac{1}{4}$:

$$dx = \frac{1 - \frac{1}{4}}{2} = \tfrac{3}{8}, \quad dy = 1 - 2\left(\tfrac{1}{4}\right) = \tfrac{1}{2}$$

$$\Delta A = \tfrac{1}{2} \cdot \tfrac{3}{8} + 1 \cdot \tfrac{1}{2} = \tfrac{3}{16} + \tfrac{8}{16} = \tfrac{11}{16} \approx 0.6875$$

Potential Change Between Spaces

The **potential difference** ΔP between these two differential environments is:

$$\Delta P = \Delta A_1 - \Delta A_2 = \left(\tfrac{1}{2\sqrt{2}} - 528\right) - \tfrac{11}{16} \approx -527.646 - 0.6875 = -528.33$$

 doi:10.20944/preprints202507.0812.v1

This gives the relative change in the algebraic function's trajectory across two differential geometries, emphasizing how much sharper the algebraic shift is under more rapidly varying space.

Interpretation and Use

- Algebraic potential measures how a function evolves in a space governed by differential dynamics.
- Different regions of space can obey different governing expressions, and comparing potentials between them helps in understanding *transitional behaviors* in dynamic systems (e.g., physical fields, control domains).

This concept can be expanded further to define scalar fields and metrics in learning algorithms or physical simulations where the "fabric" of space is encoded by differential expressions.

19. Expansion Matrices and Linear Paths

Expansion matrices derived from differential expressions describe how a differential system expands or contracts in a local region. These matrices represent linear approximations of the behavior of the system, and their eigenvectors correspond to invariant directions of linear change. However, it's crucial to understand that these linear paths described by the eigenvectors of an expansion matrix do not necessarily follow the nonlinear dynamics governed by the original differential expression. In other words, while the eigenvectors indicate dominant or preserved directions of transformation under linearization, they do not represent actual solution trajectories of the differential system. They are instead the axes of pure scaling or shearing inherent to the linear part of the system at a given point.

Worked Example: Eigenvectors of an Expansion Matrix at $y = -2$

Consider the expansion matrix:

$$M(y) = \begin{bmatrix} 1 & -\frac{1}{(1-y)^{3/2}} \\ \frac{y-1}{2} & \frac{1}{2(1-y)^2} \end{bmatrix}$$

At $y = -2$, we compute:

$$1 - y = 3, \quad (1-y)^{3/2} = 3^{3/2} = 3\sqrt{3}, \quad (1-y)^2 = 9$$

So the matrix becomes:

$$M(-2) = \begin{bmatrix} 1 & -\frac{1}{3\sqrt{3}} \\ -1.5 & \frac{1}{18} \end{bmatrix}$$

To find the eigenvectors, solve the characteristic equation:

$$\det\left(\begin{bmatrix} 1-\lambda & -\frac{1}{3\sqrt{3}} \\ -1.5 & \frac{1}{18}-\lambda \end{bmatrix}\right) = 0$$

Compute the determinant:

$$(1-\lambda)\left(\frac{1}{18}-\lambda\right) - \left(-\frac{1}{3\sqrt{3}}\right)(-1.5) = 0$$

$$(1-\lambda)\left(\frac{1}{18}-\lambda\right) - \frac{1.5}{3\sqrt{3}} = 0$$

$$= \left(\frac{1}{18} - \lambda - \frac{\lambda}{18} + \lambda^2\right) - \frac{1.5}{3\sqrt{3}} = 0$$

 doi:10.20944/preprints202507.0812.v1

$$= \lambda^2 - \frac{19}{18}\lambda + \frac{1}{18} - \frac{1.5}{3\sqrt{3}} = 0$$

Now simplify:

$$\lambda^2 - \frac{19}{18}\lambda + \left(\frac{1}{18} - \frac{1.5}{3\sqrt{3}}\right) = 0$$

This is a real-valued quadratic equation. You can solve it explicitly with the quadratic formula:

$$\lambda = \frac{19}{36} \pm \sqrt{\left(\frac{19}{36}\right)^2 - \left(\frac{1}{18} - \frac{1.5}{3\sqrt{3}}\right)}$$

Once eigenvalues λ are known, substitute into $(M - \lambda I)\vec{v} = 0$ to find the eigenvectors.

Result:

In this case, the expansion matrix has two directions in which local linear transformation occurs due to the underlying differential system. These directions are again not necessarily flow lines of the original differential expression but reveal the instantaneous linear geometry of expansion at the point $y = -2$.

20. Approximation of Differential Expressions with Infinite Series

Infinite series can be used to approximate and analyze differential expressions by expressing them as expansions involving powers of differentials. This allows for the use of truncated series to model more complex behavior while retaining analytic tractability.

General Framework

Consider a differential expression of the form:

$$dx = \sum_{n=0}^{\infty} \frac{dy^n}{n!}$$

This is the formal power series expansion of the exponential function:

$$dx = e^{dy}$$

Here, we interpret dx not as an infinitesimal increment in the classical sense, but as a differential expression defined by the infinite sum of powers of dy. The variable dy acts as the generator of this expansion.

Finite Approximation

To work with this expression computationally or analytically, we may truncate the infinite series at a finite number of terms. For example, truncating at $n = 3$, we get:

$$dx \approx 1 + dy + \frac{dy^2}{2} + \frac{dy^3}{6}$$

Subtracting 1 from both sides and isolating the expression gives:

$$dx - dy - \frac{dy^2}{2} - \frac{dy^3}{6} = 1$$

This is a differential expression we can work with:

$$dx - dy - \frac{dy^2}{2} - \frac{dy^3}{6} = 1$$

 doi:10.20944/preprints202507.0812.v1

Solving for the Step Differential

$$dx = e^{dy} \Rightarrow dy = \ln(x)$$

This shows that the step differential dy can be recovered from the logarithmic inversion of the exponential differential expression.

21. Using Differential Expressions over a Closed Path in Green's Theorem

We visit Green's Theorem using a symbolic differential system, and construct a closed path composed of two legs: a forward curve and a return curve.

Vector Field and Differential Expressions

Let the vector field be:

$$\vec{F}(x,y) = (P(x,y), Q(x,y)) = (x^2, y^2)$$

We define two segments of the closed curve C:

- Forward path C_1: Parameterized with

$$x_1(t) = 1 + t, \quad y_1(t) = t^2, \quad t \in [0,1]$$

and differential expressions:

$$dx = \frac{1-x}{2}, \quad dy = 1 - 2y$$

- Return path C_2: Simple linear return from $(2,1)$ to $(1,0)$

$$x_2(t) = 2 - t, \quad y_2(t) = 1 - t, \quad t \in [0,1]$$

Line Integral over the Closed Path

We compute:

$$\oint_C x^2\,dx + y^2\,dy = \int_{C_1} x^2\,dx + y^2\,dy + \int_{C_2} x^2\,dx + y^2\,dy$$

Line Integral with Differential Expressions

The line integral becomes:

$$\int_C x^2\,dx + y^2\,dy = \int_0^1 (x(t)^2 \cdot dx(t) + y(t)^2 \cdot dy(t))\,dt$$

Substituting:

$$\int_0^1 ((1+t)^2 \cdot \left(\frac{-t}{2}\right) + (t^2)^2 \cdot (1 - 2t^2))\,dt$$

$$= \int_0^1 \left[-\frac{t(1+t)^2}{2} + t^4(1-2t^2)\right] dt$$

$$= \int_0^1 \left[-\frac{t + 2t^2 + t^3}{2} + t^4 - 2t^6\right] dt$$

Evaluating the integral:

$$= -\frac{1}{2}\int_0^1 t + 2t^2 + t^3\,dt + \int_0^1 t^4\,dt - 2\int_0^1 t^6\,dt$$

 doi:10.20944/preprints202507.0812.v1

$$= -\frac{1}{2}\left(\frac{1}{2} + \frac{2}{3} + \frac{1}{4}\right) + \frac{1}{5} - 2 \cdot \frac{1}{7}$$

$$= -\frac{1}{2} \cdot \frac{17}{12} + \frac{1}{5} - \frac{2}{7} = -\frac{17}{24} + \frac{1}{5} - \frac{2}{7} = -\frac{667}{840} = -0.79404\ldots$$

Return Leg (C_2) Using Symbolic Differential Expressions

We now compute the return leg C_2, forming a closed path. The return is along the path:

$$x_2(t) = 2 - t, \quad y_2(t) = 1 - t, \quad \text{for } t \in [0, 1]$$

We use the same differential expressions:

$$dx = \frac{1 - x}{2}, \quad dy = 1 - 2y$$

Compute each differential expression along this return path:

$$dx(t) = \frac{1 - x_2(t)}{2} = \frac{1 - (2 - t)}{2} = \frac{-1 + t}{2}$$
$$dy(t) = 1 - 2y_2(t) = 1 - 2(1 - t) = -1 + 2t$$

Now evaluate the line integral along C_2:

$$\int_{C_2} x^2\,dx + y^2\,dy = \int_0^1 x_2(t)^2 \cdot dx(t) + y_2(t)^2 \cdot dy(t)\,dt$$

$$x_2(t)^2 = (2 - t)^2 = 4 - 4t + t^2, \quad y_2(t)^2 = (1 - t)^2 = 1 - 2t + t^2$$

Substitute into the integrand:

$$(2 - t)^2 \cdot \frac{-1 + t}{2} + (1 - t)^2 \cdot (-1 + 2t)$$

Expanding both terms:

$$= \frac{(4 - 4t + t^2)(-1 + t)}{2} + (1 - 2t + t^2)(-1 + 2t)$$
$$= \frac{-4 + 8t - 5t^2 + t^3}{2} + (-1 + 4t - 5t^2 + 2t^3)$$

Combine into one integral:

$$\int_0^1 \left(-3 + 8t - \frac{15}{2}t^2 + \frac{5}{2}t^3\right) dt$$

Compute the integral:

$$= \left[-3t + 4t^2 - \frac{5}{2}t^3 + \frac{5}{8}t^4\right]_0^1 = -3 + 4 - \frac{5}{2} + \frac{5}{8} = -\frac{7}{8}$$

Result.

The return leg contributes:

$$\int_{C_2} x^2\,dx + y^2\,dy = -\frac{7}{8}$$

This closes the path and allows us to compare the total circulation around the full loop versus the classical Green's Theorem prediction.

 doi:10.20944/preprints202507.0812.v1

Total Closed Curve Integral

$$\oint_C x^2\,dx + y^2\,dy = -\frac{667}{840} - \frac{7}{8} = -\frac{701}{420} \approx -1.669$$

Transformation to Green's Theorem: Classical vs Differential Expressions

Classically, Green's Theorem states that for a simple closed, positively oriented curve C that bounds a region R, we have:

$$\oint_C P\,dx + Q\,dy = \iint_R \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dx\,dy$$

For our vector field $\vec{F}(x,y) = (x^2, y^2)$, we compute:

$$\frac{\partial Q}{\partial x} = \frac{\partial y^2}{\partial x} = 0, \quad \frac{\partial P}{\partial y} = \frac{\partial x^2}{\partial y} = 0$$

Thus, the classical Green's Theorem gives:

$$\oint_C x^2\,dx + y^2\,dy = \iint_R (0-0)\,dx\,dy = 0$$

Interpretation

We have constructed a closed curve where both legs obey symbolic differential expressions. The line integral around this loop is non-zero, despite the fact that:

$$\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} = 0$$

This shows that **differential expressions define a modified traversal geometry**, introducing flux-like contributions that are not detectable through classical computations. Green's Theorem assumes a uniform differential structure. This can be viewed as a discrepancy of differential expressions and Green's theorem in the classical context, but setting new definitions for differential path traveling eliminates this. This underscores the need for an expanded vector calculus framework, where integrals over differential expression-governed domains require new theorems beyond classical formulations.

22. Sequences of Differential Expressions and Continuous Differential Spaces

We introduce the notion of **sequences of differential expressions**, where each expression results from expanding a symbolic binomial form such as $(dx + dy)^n$. These form a **family of expressions** that share the same general structure and solution behavior.

Binomial Expansion of Differentials

The binomial expansion of differentials is given by:

$$(dx + dy)^n = \sum_{k=0}^{n} \binom{n}{k} dx^{n-k} dy^k$$

We treat each expression for fixed n as a differential expression to be solved.

Worked Binomial Expansions

First power:

$$dx + dy = 1 \Rightarrow dx = 1 - y \quad (\mathcal{S}_{dy})$$

Second power:

$$dx^2 + 2dxdy + dy^2 = 1$$

 doi:10.20944/preprints202507.0812.v1

Apply $\mathcal{S}_g^{-1}$: $2dx + 2\delta_y^2\delta_x^2 + 2dy = 1$, then apply $\mathcal{S}_g$ then $\mathcal{S}_y$: $2x + 2dxy + y^2 = 1$

$$dx = \frac{1 - y^2 - 2x}{2y}$$

Third power:

$$dx^3 + 3dx^2dy + 3dxdy^2 + dy^3 = 1$$

Apply $\mathcal{S}_g^{-1}$ twice, then $\mathcal{S}_g$, then $\mathcal{S}_{dx}, \mathcal{S}_{y^2}$

$$6x + 6xy + 3dxy^2 + y^3 = 1 \Rightarrow dx = \frac{1 - y^3 - 6x - 6xy}{3y^2}$$

Fourth power:

$$dx^4 + 4dx^3dy + 6dx^2dy^2 + 4dxdy^3 + dy^4 = 1$$

Apply $\mathcal{S}_g^{-1}$ three times, then $\mathcal{S}_g$, then $\mathcal{S}_{dx}^2, \mathcal{S}_{dy}^3$

$$24x + 24xy + 12xy^2 + 4dxy^3 + y^4 = 1 \Rightarrow dx = \frac{1 - y^4 - 12xy^2 - 24xy - 24x}{4y^3}$$

Theorem: General Solution for Scaled Binomial Sequence

For the sequence $(dx + dy)^k = 1$, the general solution operator is:

$$G = \mathcal{S}_y^{k-1} \circ \mathcal{S}_{dx}^{k-2} \circ \mathcal{S}_g \circ \mathcal{S}_g^{-(k-1)} \quad \text{for } k \geq 2$$

This operator scales down and then re-scales the expression to isolate. For each step in the sequence, an additional scaling and reduction is required, except for the $\mathcal{S}_g$ term, which is there to scale the differential term to its variable form.

Definition: Family of Expressions

A **family of expressions** consists of all differential expressions that share the same general solution form G.

Corollary

The step differential dx for any member of a sequence can be written as an algebraic formula.

Theorem: General Scaled Binomial Formula for dx

$$dx_k = \frac{1 - y^k - k! \cdot \mathcal{O}(x^{k-2}y^{k-2})}{ky^{k-1}}$$

Explanation: - y^k is always the last term, scaled from dy to y by keeping the same power, and is subtracted from 1. - All other terms involve powers no higher than $x^{k-2}y^{k-2}$ due to scaling. - The coefficient of the second-to-last term (which multiplies dx) is ky^{k-1}, giving the denominator.

Table of step differentials

k	Step Differential dx
1	$dx = 1 - y$
2	$dx = \frac{1-y^2-2x}{2y}$
3	$dx = \frac{1-y^3-6x-6xy}{3y^2}$
4	$dx = \frac{1-y^4-12xy^2-24xy-24x}{4y^3}$
k	$dx_k = \frac{1-y^k-k!\cdot\mathcal{O}(x^{k-2}y^{k-2})}{ky^{k-1}}$

 doi:10.20944/preprints202507.0812.v1

Time Integration from Differential Expressions and Velocity

Now that we have a general formula for dx_k in terms of the binomial expansion of differentials, we observe that at each point (x, y) in space, we are effectively dealing with a different differential expression. This means the nature of space itself is dynamic and encoded in dx_k.

We define the **local velocity** at a point as:

$$v(x) = \frac{dx_k}{dt} \quad \Rightarrow \quad dt = \frac{dx_k}{v(x)}$$

This gives a new way to calculate the **time it takes to traverse a path**, where the time increment is governed by the ratio of the differential expression to the velocity at that point.

Worked Example: Third Binomial Expansion with Proxy $y = x$

From the third binomial expansion:

$$dx = \frac{1 - y^3 - 6x - 6xy}{3y^2}$$

Using proxy $y = x$, we substitute:

$$dx = \frac{1 - x^3 - 6x - 6x^2}{3x^2}$$

Let velocity be $v(x) = e^x$. Then:

$$dt = \frac{dx}{v(x)} = \frac{1 - x^3 - 6x - 6x^2}{3x^2 e^x}$$

Total time to traverse from $x = 1$ to $x = 3$:

$$T = \int_1^3 \frac{1 - x^3 - 6x - 6x^2}{3x^2 e^x}\, dx = -1.17940998903$$

This integral encodes the time it takes to traverse a space whose geometry is defined by the differential expression dx, while moving at velocity e^x.

Chi Change of Direction

We define the **Chi Change of Direction** (χ) as the total change induced by a differential expression in a region of space. Formally, it is computed by evaluating the double integral of the step differential dx_k over both variables:

$$\chi = \iint_R dx_k(x, y)\, dy\, dx$$

Worked example: Let us use the 3rd-order binomial expansion differential expression:

$$(dx + dy)^3 = dx^3 + 3dx^2 dy + 3dx dy^2 + dy^3 = 1$$

Solving this (as previously derived), we obtain:

$$dx = \frac{1 - y^3 - 6x - 6xy}{3y^2}$$

We now compute the following double integral over some region of space defined by the following bounds:

$$\chi = \int_0^{0.75} \int_1^{2\sqrt{1-x}} \frac{1 - y^3 - 6x - 6xy}{3y^2}\, dy\, dx$$

 doi:10.20944/preprints202507.0812.v1

We evaluate the indefinite integral:

$$\int \frac{1 - y^3 - 6x - 6xy}{3y^2}\, dy$$

Distribute the denominator across each term in the numerator:

$$= \frac{1}{3}\int \left(\frac{1}{y^2} - \frac{y^3}{y^2} - \frac{6x}{y^2} - \frac{6xy}{y^2}\right) dy = \frac{1}{3}\int \left(y^{-2} - y - 6xy^{-2} - 6xy^{-1}\right) dy$$

Integrate term by term:

$$\int y^{-2}\, dy = -y^{-1} = -\frac{1}{y}$$
$$\int y\, dy = \frac{y^2}{2}$$
$$\int 6xy^{-2}\, dy = 6x\int y^{-2}\, dy = 6x\left(-\frac{1}{y}\right) = -\frac{6x}{y}$$
$$\int 6xy^{-1}\, dy = 6x\int y^{-1}\, dy = 6x\ln|y|$$

Combine all terms:

$$\frac{1}{3}\left(-\frac{1}{y} - \frac{y^2}{2} + \frac{6x}{y} - 6x\ln|y|\right) + C$$

$$= \frac{1}{3}\left(\frac{6x-1}{y} - \frac{y^2}{2} - 6x\ln|y|\right) + C$$

$$\int \frac{1 - y^3 - 6x - 6xy}{3y^2}\, dy = \frac{1}{3}\left(\frac{6x-1}{y} - \frac{y^2}{2} - 6x\ln|y|\right) + C$$

Step 1: Plug in bounds for y.
Let:

$$y_{\text{top}} = 2\sqrt{1-x}, \quad y_{\text{bottom}} = 1$$

Then the definite result of the inner integral becomes:

$$\Delta_y(x) = \frac{1}{3}\left[\left(\frac{6x-1}{2\sqrt{1-x}} - \frac{(2\sqrt{1-x})^2}{2} - 6x\ln(2\sqrt{1-x})\right) - \left((6x-1) - \frac{1}{2} - 6x\ln 1\right)\right]$$

Simplify:

$$\Delta_y(x) = \frac{1}{3}\left[\frac{6x-1}{2\sqrt{1-x}} - 2(1-x) - 6x\ln(2\sqrt{1-x}) - (6x - 1.5)\right]$$

Step 2: Integrate the resulting expression over $x \in [0, 0.75]$:

$$\chi = \int_0^{0.75} \Delta_y(x)\, dx \approx -0.42752218056$$

This value shows the total change of the general step differential in both variables x and y, and is useful for computing the total change in 2 dimensions.

23. Useful Reading Material

Many of the concepts in this paper rely on calculus [1,2], linear algebra [3], discrete mathematics [4] and mathematical analysis [5].

 doi:10.20944/preprints202507.0812.v1

24. Conclusions

This research has proposed a novel interpretation of differential expressions as dynamic, symbolic structures that redefine the role of space, transformation, and movement in both mathematical analysis and applied systems such as image processing and field integration. Central to this framework is the concept that differential expressions can be structured into families, possess general solution operators derived through a hierarchy of scaling transforms, and act as governing laws over localized spatial behavior. Rather than treating differentials as infinitesimal results of parameterized motion, we have shown that they can be solved and scaled directly to form algebraic step functions that govern the evolution of values across discrete or continuous fields.

Through worked examples such as the binomial expansion of $(dx + dy)^n = 1$, we derived a general solution for dx, showing how scaling and symbolic manipulation produce interpretable algebraic forms. This not only supports the classification of such expressions into expression families but also suggests a recursive structure where higher-order expansions reveal deeper geometric and functional characteristics of differential behavior.

In applied contexts, we adapted this framework to image grids and spatial learning, using coefficient maps to locally scale directional steps and match desired paths through a grid. These coefficients represent interpretable physical or geometric influences, learned through optimization and reusable across tasks. This is also applied to deep learning models with constraints. Furthermore, we demonstrated how classical vector calculus — exemplified by Green's Theorem — fails to hold when traversal is governed by symbolic differential expressions rather than time-parametrized paths, emphasizing the need for a generalized vector calculus in symbolic differential spaces.

In addition to symbolic differential expressions, this work also explored the structural interpretation of such expressions through nodes and maps. By decomposing a differential expression into accumulating nodes — each representing a localized dynamic governed by a subset of terms — we established a method for simulating time-based evolution across systems. These nodes interact through maps, which define the flow of terms between expressions over time, enabling the construction of dynamic networks with memory and programmable behavior.

Furthermore, the introduction of the expansion matrix provides a linearized lens through which local transformation behavior can be analyzed. This is expanded with the idea of algebraic potential of algebraic expressions, which gives the expected change under a differential expression.

This framework opens promising directions for extending symbolic calculus to nonlinear geometry, field-based cryptography, neural networks, and dynamic spatial systems, forming a foundation for further theoretical development and real-world application.

References

1. Larson, R.; Edwards, B.H. *Calculus: Early Transcendental Functions*, 6th ed.; Cengage Learning, 2013.
2. Ayres, F.; Mendelson, E. *Schaum's Outline of Calculus*, 6th ed.; McGraw-Hill Education, 2012.
3. Lipschutz, S.; Lipson, M. *Schaum's Outline of Linear Algebra*, 6th ed.; McGraw-Hill Education, 2017.
4. Lipschutz, S.; Lipson, M. *Schaum's Outline of Discrete Mathematics*, 4th ed.; McGraw-Hill Education, 2021.
5. Rudin, W. *Principles of Mathematical Analysis*, 3rd ed.; McGraw-Hill Education, 1976.