

# Data-Driven Hierarchical Runge-Kutta Methods

## For Nonlinear Dynamical Systems

Shyamal Suhana Chandra

Sapana Micro Software

2025

- Euler's Method (1st order)
- Data-Driven Euler's Method
- Runge-Kutta 3rd order method
- Data-Driven Runge-Kutta
- Adams-Bashforth and Adams-Moulton methods
- Data-Driven Adams Methods
- Hierarchical data-driven architecture
- Transformer-inspired ODE solver
- Objective-C framework for Apple platforms

## Algorithm

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

- Simplest numerical method
- First-order accuracy
- Local truncation error:  $O(h^2)$
- Fast computation
- Foundation for higher-order methods

## Enhanced Algorithm

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) + h \cdot \alpha \cdot \text{Attention}(y_n)$$

- Hierarchical transformer layers
- Attention mechanisms
- Adaptive correction
- Enhanced accuracy over standard Euler

## Algorithm

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + h/2, y_n + hk_1/2)$$

$$k_3 = f(t_n + h, y_n - hk_1 + 2hk_2)$$

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 4k_2 + k_3)$$

- Good balance of accuracy and efficiency
- Suitable for nonlinear systems
- Local truncation error:  $O(h^4)$

## Adams-Bashforth (Predictor)

$$y_{n+1} = y_n + \frac{h}{12}(23f_n - 16f_{n-1} + 5f_{n-2})$$

- Multi-step methods
- Predictor-corrector scheme
- Higher order accuracy

## Adams-Moulton (Corrector)

$$y_{n+1} = y_n + \frac{h}{12}(5f_{n+1} + 8f_n - f_{n-1})$$

# Hierarchical Architecture

- Transformer-inspired design
- Multiple processing layers
- Attention mechanisms
- Adaptive refinement
- Data-driven learning

## Key Features

- Hierarchical state transformations
- Learnable weights and biases
- Self-attention for ODE solutions
- Adaptive step size control

# Implementation

## Core

- C/C++ implementation
- High performance
- Memory efficient

## Framework

- Objective-C wrappers
- Visualization support
- macOS & VisionOS



# Test Cases: Exponential Decay

## ODE

$$\frac{dy}{dt} = -y, \quad y(0) = 1.0$$

Exact:  $y(t) = \exp(-t)$

## C/C++ Implementation

```
void exponential_ode(double t, const double* y,  
    double* dydt, void* params) {  
    dydt[0] = -y[0];  
}
```

## Results

All methods: 99.99999% accuracy

# Test Cases: Harmonic Oscillator

## ODE

$$\frac{d^2x}{dt^2} = -x, \quad x(0) = 1.0, v(0) = 0.0$$

Exact:  $x(t) = \cos(t)$ ,  $v(t) = -\sin(t)$

## C/C++ Implementation

```
void oscillator_ode(double t, const double* y,
    double* dydt, void* params) {
    dydt[0] = y[1]; // dx/dt = v
    dydt[1] = -y[0]; // dv/dt = -x
}
```

## Results

RK3/DDRK3: 99.68%, AM/DDAM: 99.32%

# Applications

- Nonlinear dynamical systems
- Chaotic systems (Lorenz, etc.)
- Engineering simulations
- Scientific computing
- Real-time visualization

# Thank You

## Questions?

`github.com/Sapana-Micro-Software/ddrkam`