Data-Driven Hierarchical Runge-Kutta Methods

For Nonlinear Dynamical Systems

Shyamal Suhana Chandra

Sapana Micro Software

2025

Overview

- 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

Euler's Method

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

Data-Driven Euler's Method

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

Runge-Kutta 3rd Order

Algorithm

$$
\begin{aligned}
k_1 &= f(t_n, y_n) \\
k_2 &= f(t_n + h/2, y_n + hk_1/2) \\
k_3 &= f(t_n + h, y_n - hk_1 + 2hk_2) \\
y_{n+1} &= y_n + \frac{h}{6}(k_1 + 4k_2 + k_3)
\end{aligned}
$$

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

Adams Methods

Adams-Bashforth (Predictor)

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

Adams-Moulton (Corrector)

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

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

Parallel and Distributed Execution

Execution Modes

- OpenMP: Shared-memory multi-threading
- pthreads: POSIX threads for fine control
- MPI: Distributed computing
- Hybrid: MPI + OpenMP

- Parallel speedup up to N$\times$ with N workers
- Distributed scaling across nodes
- Concurrent execution of multiple methods

Real-Time, Online, and Dynamic Methods

Execution Modes

- Real-Time: Streaming data, minimal latency
- Online: Adaptive learning, incremental updates
- Dynamic: Adaptive step sizes, parameter tuning

- Suitable for live data feeds
- Adaptive to system changes
- Optimized for continuous operation

Nonlinear Programming Solvers

Methods

- Gradient Descent
- Newton's Method
- Quasi-Newton (BFGS)
- Interior Point
- Sequential QP
- Trust Region

- ODEs as optimization problems
- PDEs as optimization problems
- Enhanced convergence

Combined Solvers

- Distributed + Data-Driven
- Online + Data-Driven
- Real-Time + Data-Driven
- Distributed + Online
- Distributed + Real-Time

Benefits

Maximum flexibility and performance through method combinations

Stacked and 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

RK3: 0.000036s, 100.00% accuracy

DDRK3: 0.001129s, 100.00% 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: 0.000099s, 99.68% accuracy

DDRK3: 0.003575s, 99.68% accuracy

Applications

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

Thank You

Questions?

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