DDRKAM Reference Manual
Data-Driven Runge-Kutta and Adams Methods

Shyamal Suhana Chandra

2025

Contents

1 Introduction

This manual provides comprehensive documentation for the DDRKAM (Data-Driven Runge-Kutta and Adams Methods) framework. The framework implements numerical methods for solving ordinary differential equations (ODEs) with support for traditional and hierarchical data-driven approaches.

2 Runge-Kutta 3rd Order Method

2.1 Overview

The Runge-Kutta 3rd order method provides a good balance between accuracy and computational efficiency for solving ODEs.

2.2 API Reference

2.2.1 rk3_step

Performs a single integration step using RK3.

```c
double rk3_step(ODEFunction f, double t0, double* y0,
                size_t n, double h, void* params);
```

Parameters:

- `f`: Function pointer to the ODE system
- `t0`: Current time
- `y0`: Current state vector (modified in-place)
- `n`: Dimension of the system
- `h`: Step size
- `params`: User-defined parameters

Returns: New time value (t0 + h)

2.2.2 rk3_solve

Solves an ODE system over a time interval.

```
size_t rk3_solve(ODEFunction f, double t0, double t_end,
                 const double* y0, size_t n, double h,
                 void* params, double* t_out, double*
                    y_out);
```

Parameters:

- `f`: Function pointer to the ODE system
- `t0`: Initial time
- `t_end`: Final time
- `y0`: Initial state vector
- `n`: Dimension of the system
- `h`: Step size
- `params`: User-defined parameters
- `t_out`: Output time array (allocated by caller)
- `y_out`: Output state array (n × num_steps, allocated by caller)

Returns: Number of steps taken

2.3 Example

```
void lorenz(double t, const double* y, double* dydt,
   void* params) {
    double* p = (double*)params;
    double sigma = p[0], rho = p[1], beta = p[2];
    dydt[0] = sigma * (y[1] - y[0]);
    dydt[1] = y[0] * (rho - y[2]) - y[1];
    dydt[2] = y[0] * y[1] - beta * y[2];
}

double params[3] = {10.0, 28.0, 8.0/3.0};
```

```c
double y0[3] = {1.0, 1.0, 1.0};
double t_out[100];
double y_out[300];
size_t steps = rk3_solve(lorenz, 0.0, 1.0, y0, 3, 0.01,
                         params, t_out, y_out);
```

3 Adams Methods

3.1 Adams-Bashforth 3rd Order

Predictor step for multi-step integration.

```c
void adams_bashforth3(ODEFunction f, const double* t,
                      const double* y, size_t n, double
                         h,
                      void* params, double* y_pred);
```

3.2 Adams-Moulton 3rd Order

Corrector step for multi-step integration.

```c
void adams_moulton3(ODEFunction f, const double* t,
                    const double* y, size_t n, double h,
                    void* params, const double* y_pred,
                    double* y_corr);
```

4 Hierarchical Runge-Kutta Method

4.1 Overview

The hierarchical RK method uses a transformer-like architecture with multiple processing layers and attention mechanisms.

4.2 API Reference

4.2.1 hierarchical_rk_init

Initializes a hierarchical RK solver.

```c
int hierarchical_rk_init(HierarchicalRKSolver* solver,
                         size_t num_layers, size_t
                            state_dim,
                         size_t hidden_dim);
```

Returns: 0 on success, -1 on failure

4.2.2 hierarchical_rk_free

Frees resources allocated by the solver.

```c
void hierarchical_rk_free(HierarchicalRKSolver* solver);
```

4.2.3 hierarchical_rk_solve

Solves an ODE using the hierarchical method.

```c
size_t hierarchical_rk_solve(HierarchicalRKSolver*
   solver,
                             ODEFunction f, double t0,
                                double t_end,
                             const double* y0, double h,
                                 void* params,
                             double* t_out, double*
                                y_out);
```

5 Objective-C Framework

5.1 DDRKAMSolver

Main solver class for Objective-C applications.

```objc
DDRKAMSolver* solver = [[DDRKAMSolver alloc]
                         initWithDimension:3];
NSDictionary* result = [solver solveWithFunction:^(
   double t,
                                                     const
                                                        double
                                                        *
```

```
                                                       y
                                                       ,
                                                    double
                                                       *

                                                       dydt
                                                       ,
                                                    void
                                                       *

                                                       params
                                                       )

                                                       {
    // ODE definition
} startTime:0.0 endTime:1.0
initialState:@[@1.0, @1.0, @1.0]
stepSize:0.01 params:NULL];
```

5.2 DDRKAMVisualizer

Visualization component for plotting solutions.

```
DDRKAMVisualizer* viz = [[DDRKAMVisualizer alloc] init];
NSView* view = [viz createVisualizationViewWithTime:
   timeArray
                                              state:
                                                 stateArray

                                          dimension:3];
[viz exportToCSV:@"/path/to/output.csv"
            time:timeArray
           state:stateArray];
```

5.3 DDRKAMHierarchicalSolver

Hierarchical solver for Objective-C.

```
DDRKAMHierarchicalSolver* solver =
    [[DDRKAMHierarchicalSolver alloc]
    initWithDimension:3 numLayers:4 hiddenDim:32];
```

6 Platform Support

- macOS 10.13+
- iOS 11.0+
- visionOS 1.0+

7 Copyright