Module Interface Specification for LODES (Library of ODE Solvers)

Paul Aoanan

November 22, 2017

1 Revision History

Date	Version	Notes
November 22, 2017	1.0	Initial draft.

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at the following Github link:
https://github.com/aoananp/cas741/blob/master/Doc/SRS/CA.pdf

Contents

3 Introduction

The following document details the Module Interface Specifications for LODES, the Library of ODE Solvers.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at the following link: https://github.com/aoananp/cas741.

4 Notation

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1|c_2 \Rightarrow r_2|...|c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by LODES.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of LODES uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, LODES uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	External Interface Module Euler's Method Module Trapezoidal Method Module Heun's Method Module Runge-Kutta's Method Module
Software Decision Module	Equation String Parser Module Output Format Module

Table 1: Module Hierarchy

6 MIS of External Interface Module

This module is the interface exposed to the external world or driver program. It provides access to the library and returns the solution to the ODE IVP.

6.1 Module

lodes

6.2 Uses

EqParse, Euler (Section 7), Trap (Section **??**), Heun (Section **??**), RK (Section **??**), Output (Section **??**)

6.3 Syntax

Name	In	Out	Exceptions
ODE_method	ODE_method $\in \{1, 2, 3, 4\}$	-	inputerror
ODE_eq	string	-	badODEEq
x_0	$\mathbb{R}$	-	badX0
y_0	$\mathbb{R}$	-	badY0
x_k	$\mathbb{R}$	-	badXK
h	h such that h $\in \mathbb{R}$ and h > 0	-	badH
plot	bool	plotDisplay	-
y	-	$[n$ x $1] \in \mathbb{R}$	-
success	-	BOOL	-

6.4 Semantics

6.4.1 State Variables

none

6.4.2 Access Routine Semantics

lodes(ODE_method, ODE_eq, x_0, y_0, x_k, h, plot):

- Pseudocode:
  (h: $\mathbb{R}$. h > 0) from input arguments

  bool eq_OK := EqParse(ODE_eq)

```
if NOT(eq_OK)
    return success = false

Select ODE_method:
    Case: 1
        y, success := euler(ODE_eq, x_0, y_0, x_k, h)
    Case: 2
        y, success := trap(ODE_eq, x_0, y_0, x_k, h)
    Case: 3
        y, success := heun(ODE_eq, x_0, y_0, x_k, h)
    Case: 4
        y, success := rk(ODE_eq, x_0, y_0, x_k, h)
```

7 MIS of Euler's Method

This module handles the implementation of solving an ODE IVP using Euler's Method.

7.1 Module

euler

7.2 Uses

None applicable.

7.3 Syntax

Name	In	Out	Exceptions
ODE_method	ODE_method $\in \{1, 2, 3, 4\}$	-	inputerror
ODE_eq	STRING	-	badODEEq
x_0	$\mathbb{R}$	-	badX0
y_0	$\mathbb{R}$	-	badY0
x_k	$\mathbb{R}$	-	badXK
h	h such that h $\in \mathbb{R}$ and h > 0	-	badH
y_k	-	$\mathbb{R}$	-
success	-	BOOL	-

7.4 Semantics

7.4.1 State Variables

none

7.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

8 MIS of [Module Name —SS]

[Use labels for cross-referencing —SS]

8.1 Module

[Short name for the module —SS]

8.2 Uses

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Exceptions
[accessProg —SS]	-	-	-

8.4 Semantics

8.4.1 State Variables

8.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- exception: [if appropriate —SS]

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

9 Appendix

[Extra information if required —SS]