Commonality Analysis for a Library of ODE Solvers (LODES)

Paul Aoanan

September 27, 2017

1 Revision History

Date	Version	Notes
September 27, 2017	1.0	Initial Draft.

2 Reference Material

This section records information for easy reference.

2.1 Table of Units

This section does not apply to this program family.

2.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The choice of symbols was made to be consistent with the numeral analysis and ordinary differential equation literature and with existing documentation for solving ordinary differential equations. The symbols are listed in alphabetical order.

symbol	**unit**	**description**
dy/dx	-	rate of change of y with respect to x
$f(x, y)$	-	Explicit form of the ODE function containing (x, y)
$f_x(x, y)$	-	Explicit form of the derivative of $f(x, y)$ with respect to x
$f_y(x, y)$	-	Explicit form of the derivative of $f(x, y)$ with respect to y
h	-	step-size from $x_{(0)}$ to the next point $x_{(1)}$, where $x_{(1)} = x_{(0)} + h$
n	-	reference recursion step.
x_0	-	Initial value x
x_k	-	Final value x
x_n	-	Intermediate n^{th} value x
y_0	-	Initial value y
y_k	-	Final value y
y_n	-	Intermediate n^{th} value y
y'	-	first order ODE = $f(x, y)$
$y^{(\mathrm{n})}$	-	ODE to the n^{th} order

2.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
IVP	Initial Value Problem
LC	Likely Change
ODE	Ordinary Differential Equation
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
LODES	Library of ODE Solvers
T	Theoretical Model

Contents

3 Introduction

In physical sciences, mathematical models are derived from scientific models to represent a real world phenomenon through formal mathematical constructs.

Scientific models in the study of radioactivity, carbon decay, and Newton's Law of Cooling involve the use of ordinary differential equations (ODEs).

Known elementary techniques of solving ODEs in the discrete domain use the linear approximation method wherein the solution is based upon assuming or "approximating" the slope of the tangent line from one reference point to the next until the target point has been reached.

The following section provides an overview of the Commonality Analysis (CA) for a program family of ODE solvers for Initial Value Problems (IVP). The developed program will be called Library of ODE Solvers (LODES). This section explains the purpose of this document,the scope of the family, and the characteristics of the intended readers and the organization of the document.

3.1 Purpose of Document

The main purpose of this document is to formally describe program families of the known well-known methods of solving Initial Value Problems of ODEs. The goals and mathematical models used in the LODES code are provided with an emphasis on explicitly identifying assumptions, constraints, and unambiguous definitions.

This document contains the description of the functionalities of the LODES software library. This document leads is the starting point for the subsequent software development activities, including writing the requirements specification, design specification, code, and the software verification and validation plan and execution.

3.2 Scope of the Family

The scope of the family is limited to the library of IVP ODE solvers. Given the appropriate inputs, each program in LODES is intended to find the solution to an Initial Value ODE problem.

3.3 Characteristics of Intended Reader

Reviewers of this document should have an elementary understanding of ordinary differential equations and numerical methods, as typically covered in first and second year Calculus courses. The users of LODES can have a lower level expertise, as explained in Section 4.2.

3.4 Organization of Document

The organization of this document follows the template for a CA for scientific computing software proposed by Smith (2006). The presentation follows the standard pattern of presenting

goals, theories, definitions, and assumptions. For readers that would like a more bottom up approach, they can start reading the instance models in Section 6.1 and trace back to find any additional information they require. The instance models provide the methods to solve Initial Value Problems (IVP) of Ordinary Differential Equations (ODEs).

4 General System Description

This section identifies the interfaces between the system and its environment, describes the potential user characteristics and lists the potential system constraints.

4.1 Potential System Contexts

Figure 1 shows the system context. A circle represents an external entity outside the software, the user/programmer in this case. The programmer shall use LODES in solving the ODE Initial Value Problems (IVPs). A rectangle represents the software system itself (LODES). Arrows are used to show the data flow between the system and its environment.

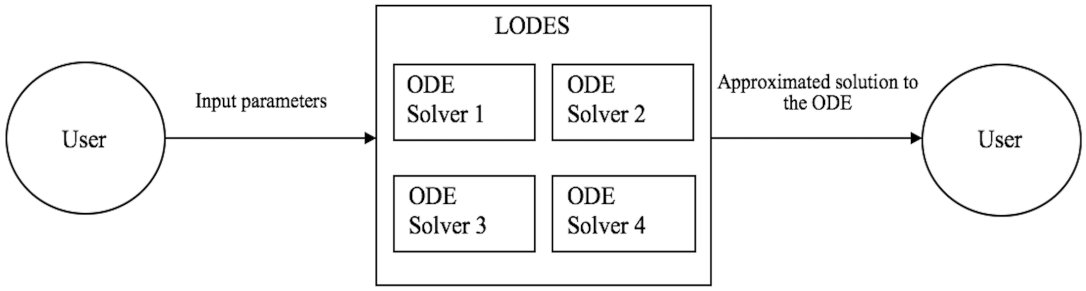

Figure 1: System Context

Programs in LODES are used inside a wrapper program. The external interaction is through program calls. The solution to the ODE IVP is the output of the function. The responsibilities of the user and the system are as follows:

- User Responsibilities:
  - Provide the correct program call, while adhering to conventions of the program's prototype
  - Provide the input details of the ODE to be solved, ensuring no errors in data entry
  - Declaration of the ODE method to be used in solving the ODE
  - Provide a valid ODE function which assumes a first order continuous ODE

- LODES Responsibilities:
  - Detect an improper input, such as invalid characters in the ODE statement and incomplete input arguments
  - Parse the input ODE string into a meaningful ODE equation that can be manipulated by the machine
  - Detect a data type mismatch where applicable, such as a string of characters in a floating point argument
  - Calculate the solution to the ODE problem
  - Provide the user access to the outputs of the program

4.2 Potential User Characteristics

The end user of LODES should have at least some basic programming experience and an understanding of undergraduate Level 1 Calculus.

4.3 Potential System Constraints

There are no system constraints applicable.

5 Commonalities

This section first presents the background and motivation of the program family, which gives a high-level view of the problem to be solved. This is followed by the terminologies used, data definitions, goals of the software, and the theoretical models used.

5.1 Background Overview

LODES is a software library developed to be provide a means to solve Initial Value Problems for ODEs using numerical methods. It can be used to solve different variations of ODEs given their initial values. It can be implemented to find the most accurate method (the method which produces the least error in their scientific computing implementation).

5.2 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- Initial values: "starting point" (x_0, y_0) of known values that exists in the domain of the solution

- Final values: An "ending point" (x_k, y_k) of unknown value y_k that exists in the domain of the solution
- Step size: The measure of arbitrary positive value (h) from the starting point (x_0, y_0) of the domain to the next (x_1, y_1), where $x_1 = x_0 + h$
- Derivative: The amount by which a a function changes at any given point as an instantaneous rate of change
- Numerical Analysis: A branch of mathematics and computer science wherein the solutions are numerical approximations taking into account the errors involved in the process.
- Numerical Approximation: An approximation of the exact solution to the problem with the use of known numerical techniques while minimizing error.
- Recursion: The repeated application of a procedure or program.
- Initial Value Problem: A type of ODE problem where the initial conditions are stated and used to find the solution to the ODE.

5.3 Data Definitions

This section collects and defines all the data needed to build the instance models. The dimension of each quantity is also given.

Number	DD1
Label	**Slope of the Tangent Line to a Curve**
Symbol	dy/dx
Equation	$dy/dx = \lim_{h \to 0} \frac{y(x+h)-y(x)}{h}$
Description	dy/dx is the slope of the tangent line to $y(x)$. dy/dx can be denoted as $y'(x)$ or $f(x, y)$. [T1] h is the step size.
Source	Nagle, et al, "Solutions and Initial Value Problems," in *Fundamentals of Differential Equations and Boundary Value Problems*, 3rd ed. USA: Addison Wesley Longman, 2000, ch. 1, p. 7.
Ref. By	IM1, IM2

5.4 Goal Statements

GS1: Given an explicit first order continuous ordinary differential equation (ODE) problem represented by $y' = f(x, y)$, the set of initial values x_0 and y_0 that satisfy $y(x_0) = y_0$, and x_k, return y_k such that $y(x_k) = y_k$ (the final values), where $y(x)$ is a function, $f(x, y)$ is a function, and x is an independent variable.

GS2: Provide the user the means of providing the required inputs, calling the ODE solver program, and returning the results.

5.5 Theoretical Models

This section focuses on the general equations and laws that LODES is based on.

Number	T1
Label	**Initial Value Problem (IVP) of First Order Ordinary Differential Equation (ODE)**
Equation	Implicit form of the first order ODE: $f(x, y, y') = o$ Explicit form of the first order ODE: $y' = f(x, y)$ Given the initial values: (x_0, y_0) such that $y(x_0) = y_0$, Find y_k such that $y(x_k) = y_k$.
Description	The above model gives the definition of an ordinary differential equation initial value problem. A differential equation is an equation, where the unknown is a function and both the function and its derivatives (rate of change) appear in the equation. An ordinary differential equation is a differential equation involving only ordinary derivatives with respect to a single independent variable. For an arbitrary ODE, the true solution will, in general, be unknown. Using the given initial values, numerical methods are used to find numerical approximations of the solution to the ODE.
Source	M. Heath, "Ordinary Differential Equations," in *Scientific Computing an Introductory Survey, 2*nd ed. New York: McGraw Hill, 2002, ch. 9.1, pp. 383 - 384.
Ref. By	DD1, T2, IM1, IM2

Number	T2
Label	**Existence and Uniqueness of the Solution**
Equations	$y' = f(x, y)$ [T1] $y(x_0) = y_0$ Assuming f and y' are continuous in $R = \{(x, y) : a < x < b, c < y < d\}$, where R is a rectangle and a, b, c, d are its vertices The initial value problem has a unique solution in some interval $x_0 - h < x < x_0 + h$
Description	The above theoretical model shows that when an equation satisfies the initial values, it is assured that a solution to the initial value problem exists. It is desirable to know whether or not the equation has an existing solution before effort is made to solve it. As well, the theoretical model states that if a solution is found, then it is the only solution to the initial value problem.
Source	Nagle, et al, "Solutions and Initial Value Problems," in *Fundamentals of Differential Equations and Boundary Value Problems*, 3rd ed. USA: Addison Wesley Longman, 2000, ch. 1, p. 12.
Ref. By	IM1, IM2

6 Variabilities

This section presents the variabilities in LODES. It details the varying instance models, gives the assumptions for the input, the variabilities in the calculations, and finally the target output.

6.1 Instance Models

This section transforms the problem defined in Section 5.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 5.5.

Number	IM1
Label	**Euler's Method of Finding the Solution to an ODE IVP**
Input	$y' = f(x, y), h, x_0, y_0, x_\mathrm{k}$
Output	y_k such that $y_\mathrm{k} = y(x_\mathrm{k})$ using the recursive formulas: $x_{\mathrm{n}+1} = x_\mathrm{n} + h, n = 0, 1, 2, ...$ $y_{\mathrm{n}+1} = y_\mathrm{n} + h * f(x_\mathrm{n}, y_\mathrm{n}), n = 0, 1, 2, ...$
Description	$y' = f(x, y)$ is the first order ODE. h is the constant step size. x_0 is the initial value of x. $x_{\mathrm{n}+1}$ is the value of x in the next equation iteration. x_k is the final value of x. y_0 is the initial value of y, such that $y_0 = y(x_0)$. $y_{\mathrm{n}+1}$ is the value of y in the next equation iteration. y_k is the final value of y. n is the reference recursion step. The above equations are used recursively until $x_{\mathrm{n}+1} = x_\mathrm{k}$ and $y_{\mathrm{n}+1} = y_\mathrm{k}$.
Sources	Nagle, et al, "The Approximation Method of Euler," in *Fundamentals of Differential Equations and Boundary Value Problems*, 3rd ed. USA: Addison Wesley Longman, 2000, ch. 1, sec. 1.5, pp. 31-32.
Ref. By	IM2

Detailed derivation of Euler's Method

Let $y' = f(x, y)$, $y(x_0) = y_0$, h as a fixed positive number (step-size), and consider the equally spaced points in the domain:

$$x_n = x_0 + nh,\ n = 0, 1, 2...$$

The construction of values y_n that approximate the solution values proceeds as follows. At the point (x_0, y_0), the slope of the solution to $y' = f(x, y)$ is given by $dy/dx = f(x_0, y_0)$. Hence the tangent line to the solution curve at the initial point is:

$y = y_0 + (x - x_0) * f(x_0, y_0)$.

Using this tangent line to approximate the next point in the solution set, we find that for the point $x_1 = x_0 + h$, we can assume the following approximation:

$y_1 = y_0 + h * f(x_0, y_0)$.

Repeating the process (recursion), until we reach y_k yields the derivation of Euler's Method.

Number	IM2
Label	**Heun's Method of Finding the Solution to an ODE IVP**
Input	$y' = f(x, y), h, x_0, y_0, x_\mathrm{k}$
Output	y_k such that $y_\mathrm{k} = y(x_\mathrm{k})$ using the recursive formulas: $x_{\mathrm{n+1}} = x_\mathrm{n} + h, n = 0, 1, 2, ...$ $y_{\mathrm{n+1}} = y_\mathrm{n} + \frac{h}{2}\{[f(x_n, y_n) + f(x_n + h, y_n + h * f(x_n, y_n)]\}, n = 0, 1, 2, ...$
Description	$y' = f(x, y)$ is the first order ODE. h is the constant step size. x_0 is the initial value of x. $x_{\mathrm{n+1}}$ is the value of x in the next equation iteration. x_k is the final value of x. y_0 is the initial value of y, such that $y_0 = y(x_0)$. $y_{\mathrm{n+1}}$ is the value of y in the next equation iteration. y_k is the final value of y. n is the reference recursion step. The above equations are used recursively until $x_{\mathrm{n+1}} = x_\mathrm{k}$ and $y_{\mathrm{n+1}} = y_\mathrm{k}$.
Sources	Nagle, et al, "Improved Euler's Method," in *Fundamentals of Differential Equations and Boundary Value Problems*, 3rd ed. USA: Addison Wesley Longman, 2000, ch. 3, sec. 3.5, pp. 124-129.
Ref. By	IM??

Number	IM3
Label	**Second Order Taylor Series Method of Finding the Solution to an ODE IVP**
Input	$y' = f(x, y), h, x_0, y_0, x_\mathrm{k}$
Output	y_k such that $y_\mathrm{k} = y(x_\mathrm{k})$ by first differentiating $f(x, y)$ to obtain: $y'' = f_x(x, y) + f_y(x, y) * f(x, y)$. using the recursive formulas: $x_{\mathrm{n}+1} = x_\mathrm{n} + h, n = 0, 1, 2, ...$ $y_{\mathrm{n}+1} = y_\mathrm{n} + h * f(x, y) + \frac{h^2}{2} * (f_x(x, y) + f_y(x, y) * f(x, y)), n = 0, 1, 2, ...$
Description	$y' = f(x, y)$ is the first order ODE. h is the constant step size. x_0 is the initial value of x. $x_{\mathrm{n}+1}$ is the value of x in the next equation iteration. x_k is the final value of x. y_0 is the initial value of y, such that $y_0 = y(x_0)$. $y_{\mathrm{n}+1}$ is the value of y in the next equation iteration. y_k is the final value of y. $f_x(x, y)$ is the explicit form of the derivative of $f(x, y)$ with respect to x $f_y(x, y)$ is the explicit form of the derivative of $f(x, y)$ with respect to y n is the reference recursion step. The above equations are used recursively until $x_{\mathrm{n}+1} = x_\mathrm{k}$ and $y_{\mathrm{n}+1} = y_\mathrm{k}$.
Sources	M. Heath, "Numerical Solution of ODEs," in *Scientific Computing an Introductory Survey, 2*nd ed. New York: McGraw Hill, 2002, ch. 9.3, p. 404.
Ref. By	IM**??**

Number	IM4
Label	**Fourth Order Runge Kutta Method of Finding the Solution to an ODE IVP**
Input	$y' = f(x, y), h, x_0, y_0, x_\text{k}$
Output	y_k such that $y_\text{k} = y(x_\text{k})$ by first differentiating $f(x, y)$ to obtain: $y'' = f_x(x, y) + f_y(x, y) * f(x, y)$. using the recursive formulas: $x_{\text{n+1}} = x_\text{n} + h, n = 0, 1, 2, ...$ $y_{\text{n+1}} = y_\text{n} + h * f(x, y) + \frac{h^2}{2} * (f_x(x, y) + f_y(x, y) * f(x, y)), n = 0, 1, 2, ...$
Description	$y' = f(x, y)$ is the first order ODE. h is the constant step size. x_0 is the initial value of x. $x_{\text{n+1}}$ is the value of x in the next equation iteration. x_k is the final value of x. y_0 is the initial value of y, such that $y_0 = y(x_0)$. $y_{\text{n+1}}$ is the value of y in the next equation iteration. y_k is the final value of y. $f_x(x, y)$ is the explicit form of the derivative of $f(x, y)$ with respect to x $f_y(x, y)$ is the explicit form of the derivative of $f(x, y)$ with respect to y n is the reference recursion step. The above equations are used recursively until $x_{\text{n+1}} = x_\text{k}$ and $y_{\text{n+1}} = y_\text{k}$.
Sources	M. Heath, "Numerical Solution of ODEs," in *Scientific Computing an Introductory Survey, 2*nd ed. New York: McGraw Hill, 2002, ch. 9.3, p. 404.
Ref. By	IM??

6.2 Assumptions

A1: It is assumed that the user will declare which of the four methods (Euler's Method, Heun's Method, Second-Order Taylor Series, Fourth-Order Runge Kutta) will be used

in solving the ODE IVP.

[Rationale: The choice of which variability is decided upon by the user/programmer on compile time as the intent is to keep the library lightweight and functional - only four most popular models were considered.]

A2: The ODE will be of the first order, given by $y' = f(x, y)$.

[Rationale: First order ODEs are chosen for the purposes of having a portable library that can be used and implemented in a variety of popular non-math oriented programming languages.]

A3: The entry of $f(x, y)$ will be of the explicit form such that $y' = f(x, y)$.

A4: The entry $f(x, y)$ can be linear or non-linear.

A5: The entry $f(x, y)$ will be a continuous function.

A6: The entry $f(x, y)$ will not have complex roots.

A7: The entry ODE problem is strictly an initial value problem; boundary value problems will not be handled.

A8: The entry for h is from the set of positive $\mathbb{R}$.

A9: The entry for x_0 is from the set of $\mathbb{R}$.

A10: The entry x_0 is of a single dimension.

A11: The entry for y_0 is from the set of $\mathbb{R}$.

A12: The entry y_0 is of a single dimension.

A13: The entry for x_k is from the set of $\mathbb{R}$.

A14: The entry x_k is of a single dimension.

6.3 Calculation

C1: Check the inputs if they satisfy the input assumptions.

C2: Parse the input ODE string and check if it adheres to the input assumptions.

C3: Perform the ODE solver model according to the user program call.

6.4 Output

O1: The output destination for y_k can be to a file, on the command line, or to memory for use by another program.

O2: The encoding of y_k can be a binary or text as shown to the user.

O3: The possible values of y_k are the set of $\mathbb{R} \cup \{-\infty, \infty, \text{undefined}\}$

O4: The program returns TRUE if it does not encounter exceptions in generating the solution to the ODE problem.

O5: The program returns FALSE if it encounters exceptions in inputs or in its calculations or errors in generating the solution to the ODE problem.

7 Traceability Matrices and Graphs

[You will have to add tables. —SS]

8 Appendix

[Your report may require an appendix. For instance, this is a good point to show the values of the symbolic parameters introduced in the report. —SS]

8.1 Symbolic Parameters

[The definition of the requirements will likely call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance. —SS]