Commonality Analysis for a Library of ODE Solvers (LODES)

Paul Aoanan

December 9, 2017

1 Revision History

Date	Version	Notes
October 20, 2017	1.0	Initial draft.
December 9, 2017	1.1	Addressed comments and first LODES revision.

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	type	description
dy/dx	-	$\mathbb{R}$	Rate of change of y with respect to x
F	-	$\mathbb{R}^3 \to \mathbb{R}$	Order function applied to the Runge Kutta Method
$f(x, y)$	-	$\mathbb{R}^2 \to \mathbb{R}$	Explicit form of the ODE function containing (x, y)
$f_x(x, y)$	-	$\mathbb{R}^2 \to \mathbb{R}$	Explicit form of the derivative of $f(x, y)$ with respect to x
$f_y(x, y)$	-	$\mathbb{R}^2 \to \mathbb{R}$	Explicit form of the derivative of $f(x, y)$ with respect to y
h	-	$\mathbb{R}$. $h > 0$	Step-size from $x_{(0)}$ to the next point $x_{(1)}$, where $x_{(1)} = x_{(0)} + h$
K_1, K_2, K_3, K_4	$\mathbb{R}$	-	Intermediary variables used in the Runge-Kutta method
n	-	$\mathbb{R}$	Reference recursion step
o	-	$\mathbb{R}$	Solution variable
x_0	-	$\mathbb{R}$	Initial value x
x_k	-	$\mathbb{R}$	Final value x
x_n	-	$\mathbb{R}$	Intermediate n^{th} value x
y_0	-	$\mathbb{R}$	Initial value y
y_k	-	$\mathbb{R}$	Final value y
y_n	-	$\mathbb{R}$	Intermediate n^{th} value y
y'	-	$\mathbb{R} \to \mathbb{R}$	Implicit form of the first order ODE $= f(x, y)$
$y^{(\mathrm{n})}$	-	$\mathbb{R} \to \mathbb{R}$	Implicit form of the ODE to the n^{th} order
y	-	1 x $\mathbb{R}^k$	The array containing all intermediate y_n values

[Usually n and k are used in the opposite way. That is, the total number of steps is n and the intermediate step is k. —SS] [Duly noted - in the scope of this project, the convention is as listed above. —PA]

2.3 Abbreviations and Acronyms

symbol	description
A	Assumption
C	Calculation
DD	Data Definition
GS	Goal Statement
IM	Instance Model
IVP	Initial Value Problem
ODE	Ordinary Differential Equation
PS	Physical System Description
SRS	Software Requirements Specification
LODES	Library of ODE Solvers
T	Theoretical Model
O	Output

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, 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 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 is the starting point for the subsequent software development activities, including writing the requirements specification, design specification, code, software verification, 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 **??**.

[For the characteristics of intended reader try to be more specific about the education. What degree? What course areas? What level? —SS] [This is listed in Section **??** —PA]

3.4 Organization of Document

The organization of this document follows the template for a CA for scientific computing software proposed by **?**. The presentation follows the standard pattern of presenting goals, theories, definitions, and assumptions. For a bottom-up approach, readers can begin reviewing the instance models in Section **??** and trace back to review additional information required. 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 **??** shows the system context. A circle represents an external entity outside the software, the driver program in this case. The driver program 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.

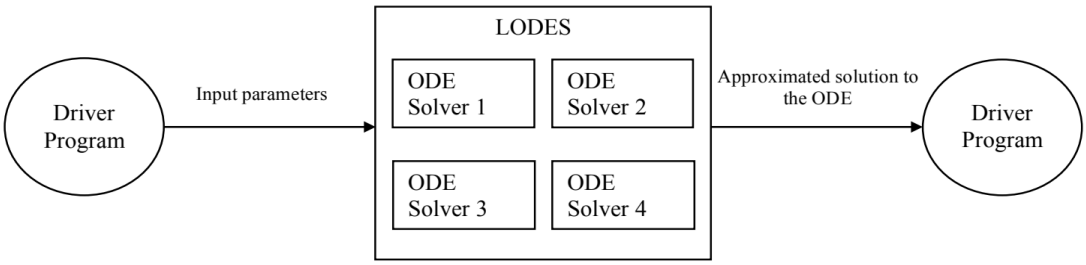

Figure 1: System Context

Programs in LODES are used inside a driver 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 primarily 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 meanings, 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 an arbitrary point (x_n, y_n) of the domain to the next (x_{n+1}, y_{n+1}), where $x_{n+1} = x_n + h$. [It appears that you have the same step size throughout your computation. You should add this fact as an assumption. Also, when you are presenting the step size, it would be better to use a generic step, rather than the first step. It is a more general way to convey the information. —SS] [Noted - the definition has been revised. —PA]
- Derivative: The amount by which a function changes at any given point as an instantaneous rate of change. [missing a period. —SS] [Period has been added. —PA]
- 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 call of a procedure, program, or subroutine to itself. [I think you can find a better definition of recursion. With this definition a for loop is recursion. :-) —SS] [Addressed. Definition has been updated. —PA]
- Initial Value Problem: A type of ODE problem where the initial conditions are stated and used to find the solution to the ODE.
- Linear Ordinary Differential Equation: A linear ordinary differential equation (ODE) is a type of ODE that has variables and their derivatives multiplied by constants. It is of the form:

$$a_n(x)y^n(x) + a_{n-1}(x)y^{n-1}(x)... + a_1(x)y^1(x) + a_0(x)y^0(x) = f(x, y)$$

  [A definition for linear ODEs would be more appropriate. —SS] [Addressed. ODE is defined. —PA]
- Continuous Function: A function which is fully defined across its domain interval.

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 $f(x, y)$. [T**??**] h is the step size.
Source	Nagle, et al, "Solutions and Initial Value Problems," [Use "quote" to get correct quotation marks —SS] [Typo addressed. —PA] in *Fundamentals of Differential Equations and Boundary Value Problems*, 3rd ed. USA: Addison Wesley Longman, 2000, ch. 1, p. 7. **?**
Ref. By	IM**??**, IM**??**, IM**??**, IM**??**

5.4 Goal Statements

GS1: Given an explicit first order continuous linear ordinary differential equation (ODE) problem represented by $y' = f(x, y)$, the set of initial values x_0, y_0 that satisfy $y(x_0) = y_0$, and x_k, and a constant step-size h used for approximation, return the set of y which contains the intermediate values, y_n to finding the solution 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.

[What about the step size h? Do you want to just solve for the final value, or do you want the intermediate values? Most of the time the scientist or engineer will be interested in the intermediate values. —SS]

[Step size h is assumed constant. The requirements to the software have been changed to produce not only the final value of y_k, but also the intermediate values. —PA]

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	[I've had a quick look for it, but I cannot find an assumption that you are only accepting the explicit form. Does that mean that you are also considering the implicit form? —SS] [Please see Assumption **??**. —PA] 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	DD??, T??, IM??, IM??, IM??, IM??, A??, A??, A??, A??, A??, A??, O??

[If you have a reference by entry, then the referenced by chunk (or its derivation) should actually reference the chunk that has it as an entry. —SS] [I'm afraid I do not quite understand what this means. Please clarify. —PA]

Number	T2
Label	**Existence and Uniqueness of the Solution**
Equations	$y' = f(x, y)$ [T??] $y(x_0) = y_0$ 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	IM??, IM??, IM??, IM??, A??, A??, A??, A??, A??, A??, A??, A??, A??, A??, A??, O??

5.6 Common Calculations

CC1: Check the inputs if they satisfy the input assumptions.
[All items in Section **??**]

CC2: Parse the input ODE string which which exhibits the characteristics laid out by [A**??**], [A**??**], [A**??**], [A**??**], and [A**??**].

5.7 Common Outputs

CO1: The program returns the intermediate and final values of y_n y_k as y.
[GS**??**], [T**??**], [T**??**]

CO2: The program returns TRUE if it does not encounter exceptions in generating the solution to the ODE problem.

CO3: The program returns FALSE if it encounters exceptions in inputs or in its calculations or errors in generating the solution to the ODE problem.
[GS**??**]

6 Variabilities

This section presents the variabilities in LODES. It details the various 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 **??** into one which is expressed in mathematical terms. It uses concrete symbols defined in Section **??**.

Number	IM1
Label	**Euler's Method of Finding the Solution to an ODE IVP**
Input	$f(x,y), h, x_0, y_0, x_\mathrm{k}$
Output	y_k such that $y_\mathrm{k} = y(x_\mathrm{k})$ using the recursive formulas: $x_{\mathrm{n}+1} = x_\mathrm{n} + h, n = 0, 1, 2, ...k-1$ $y_{\mathrm{n}+1} = y_\mathrm{n} + h * f(x_\mathrm{n}, y_\mathrm{n}), n = 0, 1, 2, ..., k-1$
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, 3*rd ed. USA: Addison Wesley Longman, 2000, ch. 1, sec. 1.5, pp. 31-32. **?**
Ref. By	IM??, IM??, A??, A??, A??, A??, A??, A??, A??

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:

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

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	**Trapezoid Method of Finding the Solution to an ODE IVP**
Input	$f(x, y), h, x_0, y_0, x_\mathrm{k}$
Output	y_k such that $y_\mathrm{k} = y(x_\mathrm{k})$ using the recursive formulas: $x_{\mathrm{n+1}} = x_\mathrm{n} + h, n = 0, 1, 2, ..., k - 1$ $y_{\mathrm{n+1}} = y_\mathrm{n} + h * [f(x_\mathrm{n}, y_\mathrm{n}) + f(x_{\mathrm{n+1}}, y_{\mathrm{n+1}})]/2, n = 0, 1, 2, ..., k - 1$
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	M. Heath, "Numerical Solution of ODEs," in *Scientific Computing an Introductory Survey, 2*nd ed. New York: McGraw Hill, 2002, ch. 9.3, p. 399 - 400. **?**
Ref. By	A??, A??, A??, A??, A??, A??, A??

Detailed derivation of the Trapezoid Method

Using [IM**??**] as a starting point:

$$x_{n+1} = x_n + h, n = 0, 1, 2, ..., k-1$$
$$y_{n+1} = y_n + h * f(x_n, y_n), n = 0, 1, 2, ..., k-1$$

The Reverse Euler's Method can be derived by taking the values at the next iteration point while working backwards:

$$x_{n+1} = x_n + h, n = 0, 1, 2, ..., k-1$$
$$y_{n+1} = y_n + h * f(x_{n+1}, y_{n+1}), n = 0, 1, 2, ..., k-1$$

Using a trapezoid formed by the two approximations, the average can be combined into the trapezoid method:

$$x_{n+1} = x_n + h, n = 0, 1, 2, ..., k-1$$
$$y_{n+1} = y_n + h * [f(x_n, y_n) + f(x_{n+1}, y_{n+1})]/2, n = 0, 1, 2, ..., k-1$$

Repeating the process (recursion) until we reach y_k yields the Trapezoid Method.

Number	IM3
Label	**Heun's Method of Finding the Solution to an ODE IVP**
Input	$f(x, y), h, x_0, y_0, x_\mathrm{k}$
Output	y_k such that $y_\mathrm{k} = y(x_\mathrm{k})$ using the recursive formulas: $x_{\mathrm{n}+1} = x_\mathrm{n} + h, n = 0, 1, 2, ..., k - 1$ $y_{\mathrm{n}+1} = y_\mathrm{n} + \frac{h}{2}\{[f(x_n, y_n) + f(x_n + h, y_n + h * f(x_n, y_n)]\}, n = 0, 1, 2, ..., k - 1$
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	A??, A??, A??, A??, A??, A??, A??

[To make it clear that you have a family of methods, I would introduce an instance model for one-step initial value problem solvers. The equation would be

$$y_{k+1} = y_k + h\phi(t_k, x_k, h_k)$$

The only difference for your family members is how ϕ is defined. This approach would let you easily add new family members in the specification, simply by giving new values for ϕ. You have also (implicitly as far as I can tell) assumed that all h_k are equal to h. —SS]

[This change is slated for the next revision of this CA as this will be a formatting change. The intent of this CA revision is to prepare the code prior to testing. —PA]

Detailed Derivation of Heun's Method

By modifying the trapezoid scheme [IM**??**]:

$$x_{n+1} = x_n + h, n = 0, 1, 2, ..., k-1$$
$$y_{n+1} = y_n + h * [f(x_n, y_n) + f(x_{n+1}, y_{n+1})]/2, n = 0, 1, 2, ..., k-1$$

by using a predicted approximation for $y_{n+1} = y_n + h * f(x_n, y_n)$, we can derive:

$$x_{n+1} = x_n + h, n = 0, 1, 2, ..., k-1$$
$$y_{n+1} = y_n + \tfrac{h}{2}\{[f(x_n, y_n) + f(x_n + h, y_n + h * f(x_n, y_n)]\}, n = 0, 1, 2, ..., k-1$$

Repeating the process (recursion) until we reach y_k yields Heun's Method.

Number	IM4
Label	**Fourth Order Runge Kutta Method of Finding the Solution to an ODE IVP**
Input	$f(x, y), h, x_0, y_0, x_{\mathrm{k}}$
Output	y_{k} such that $y_{\mathrm{k}} = y(x_{\mathrm{k}})$ using the recursive formulas: $x_{\mathrm{n+1}} = x_{\mathrm{n}} + h, n = 0, 1, 2, ..., k - 1$ $y_{\mathrm{n+1}} = y_{\mathrm{n}} + \frac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), n = 0, 1, 2, ..., k - 1$ where: $K_1 = f(x_n, y_n)$, $K_2 = f(x_n + h/2, y_n + h * K_1/2)$ $K_3 = f(x_n + h/2, y_n + h * K_2/2)$ $K_4 = f(x_n + h, y_n + h * K_3)$
Description	$y' = f(x, y)$ is the first order ODE. h is the constant step size. K_1, K_2, K_3, and K_4 are the intermediate variables. 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. 405-406. **?**
Ref. By	A??, A??, A??, A??, A??, A??, A??

Derivation of the Fourth Order Runge Kutta's Method

Runge Kutta Methods can be derived by approximating the solution to the integral using quadrature rules:

$$y_{n+1} - y_n = \int_{x_n}^{x_{n+1}} f(x, y(x))dx$$

Starting from the Taylor series representation of an ODE:

$$y_{n+1} = y_n + h * F(x_n, y_n; h)$$

where the choice of F depends on the order of method.
The exhaustive derivation of the the Runge Kutta method can be analyzed in **?**.

6.2 Assumptions

A1: It is an input that the user will declare which of the four methods (Euler's Method, Heun's Method, Trapezoid Method, Fourth-Order Runge Kutta Method) will be used in solving the ODE IVP as shown in [IM**??**], [IM**??**], [IM**??**], and [IM**??**].

[This isn't an assumption, but an input. —SS] [Reworded. —PA]

[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 classical models were considered.]

[The idea of rationales here is nice. —SS]

A2: The ODE will be of the first order, given by $y' = f(x, y)$ as described in [T**??**] and taken as an input in [IM**??**], [IM**??**], [IM**??**], and [IM**??**].

[Rationale: First order ODEs are chosen for practicality and to limit the scope for this application.]

[This rationale doesn't really hold up to scrutiny. A system of first order ODEs, which is a more general version of your scope, can be implemented in a variety of popular languages too. I believe you made this decision for practical, scope related, reasons; it is okay to say that. :-) —SS] [Thank you - rationale has been reworded. —PA]

A3: The entry of $f(x, y)$ will be of the explicit form such that $y' = f(x, y)$ as described in [T**??**] and taken as an input in [IM**??**], [IM**??**], [IM**??**], and [IM**??**].

[Okay, I thought I would have seen this sooner. An explicit reference to this assumption from T1 would be helpful. —SS]

A4: The entry $f(x, y)$ is a linear ODE as described in [T??] and taken as an input in [IM??], [IM??], [IM??], and [IM??].

[You weren't really clear on the difference between linear and non-linear ODEs. If you google it, you should be able to find a clear explanation. If not, please let me know. —SS] [Thank you - the definition of linear ODE has been presented in the initial section. As well, the software is limited to linear ODEs. —PA]

A5: The entry $f(x, y)$ will be a continuous function as described in [T??] and [T??] and taken as an input in [IM??], [IM??], [IM??], and [IM??].

A6: The entry $f(x, y)$ will not have complex roots and solutions as described in [T??] and [T??] and taken as an input in [IM??], [IM??], [IM??], and [IM??].

A7: The entry ODE problem is strictly an initial value problem; boundary value problems will not be handled in the following [T??], [T??], [IM??], [IM??], [IM??], and [IM??].

[Your last assumptions aren't really assumptions; they are just type information on the inputs. A better way to capture this would be to include the type information in your document. Including it in the list of symbols would be an easy place to start. —SS] [Last assumptions were removed and type is stated in the list of symbols. —PA]

6.3 Calculation

C1: Perform the ODE solver model according to the user program call among the options listed by in [A??] and [T??] using [IM??], [IM??], [IM??], and [IM??].

[Some of your calculations don't seem to be variabilities. In that case, they are commonalities and should be labelled as such. —SS] [Common calculations are now moved to Commonalities Section ??. —PA]

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.
[GS??]

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

[The outputs that are not variabilities should be identified as commonalities. —SS] [Common outputs are now moved to Commonalities Section ??. —PA]

7 Non-Functional Requirements

NFR1: Given the same inputs, the execution time of each program in LODES, σ_{LODES}, shall not be greater than $4 * \sigma_{\text{MatLab}}$, where σ_{MatLab} is the execution time of each equivalent program/function in MatLab.

[Rationale: The program should have a reasonable/comparable execution time compared to that of one of the industry standard's.]

NFR2: Given the same inputs, the software shall provide an interface to compare the intermediate outputs (the elements of matrix y) with that of MATLAB's built-in ODE solver by providing ε.

[Rationale: The program should provide a meaningful tool for accuracy analysis.]

[In addition to the execution time, it would be great to see a comparison of the accuracy. —SS] [Comparison of accuracy has been added. —PA]

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column or row of that component that are marked with an "X" should be modified as well. Table **??** shows the dependencies of goals, theoretical models, data definitions, instance models, and calculations with the assumptions, calculations, and outputs.

	A??	A??	A??	A??	A??	A??	A??	O??	O??	CO??	CO??	CO??
GS??							X		X	X		X
T??		X	X	X	X	X	X			X		
T??					X	X	X			X		
IM??	X	X	X	X	X	X	X					
IM??	X	X	X	X	X	X	X					
IM??	X	X	X	X	X	X	X					
IM??	X	X	X	X	X	X	X					
CC??	X	X	X	X	X	X	X					
CC??			X	X	X	X	X					
C??		X										

Table 1: Traceability Matrix Showing the Connections Between Assumptions, Calculations, and Outputs with Other Items

9 Appendix

This section shows additional information pertinent to this Commonality Analysis.

9.1 Symbolic Parameters

symbol	unit	description
σ	seconds	The measure of time a program executes.
ε	-	The norm of the relative error vector.