RK Generator

Alexander Schaap

October 14, 2017

1 Revision History

Date	Version	Notes
October 14, 2017	1.0	Initial version

2 Reference Material

This section records information for easy reference.

2.1 Table of Units

Since the application of this software is dependent on the program using the functions provided by this program family, no physical units are used throughout this document.

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 literature describing ODEs.[1] The symbols are listed in alphabetical order.

symbol	unit	description
$\mathbb{C}$	N/A	Complex numbers
$\mathbf{f}$	N/A	ODE for which to solve
h	N/A	step size
$\mathbf{k}_{1..4}$	N/A	intermediate variables in the RK4 method, similar to $\mathbf{Y}_1$
$\mathbb{R}$	N/A	Real numbers
t	N/A	an independent variable, commonly time
t_0	N/A	beginning of interval
t_N	N/A	end of interval
t_k	N/A	any t on the interval $[t_0..t_N]$
$\mathbf{x}(\mathbf{t})$	N/A	a function of t
$\mathbf{x}'$	N/A	derivative of $\mathbf{x}(\mathbf{t})$
$\mathbf{x}_0$	N/A	initial values for $\mathbf{f}$
$\mathbf{x}_k$	N/A	(numerical approximation of) a point on the solution for $\mathbf{f}$
$\mathbf{Y}_1$	N/A	intermediate variables in the RK2 method, similar to $\mathbf{k}_{1..4}$

[1] Specifically, Corless and Fillion (2013).

2.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
ODE	Ordinary Differential Equation
PS	Physical System Description
R	Requirement
RK	Runge-Kutta
RK2	Second order Runge-Kutta method
RK4	Fourth order Runge-Kutta method
rkf45.ml	Family of programs based on the RK2 / RK4 method(s)
SRS	Software Requirements Specification
STEM	Science, Technology, Engineering & Mathematics
T	Theoretical Model

Contents

3 Introduction

This document provides an overview of the commonality analysis (CA) for the rkf45.ml program family. Members of program family are produced by a code generator. Generated members provide numerical approximations for given ordinary differential equations (ODEs) using Runge-Kutta (RK) methods. Most of the calculations happens during code generation, producing a different family member for each given combination of RK method, ODE, interval, step size, and initial values. The current section describes the purpose of this document, the scope of this family, the organization of the remainder of the document and the characteristics of the intended reader.

3.1 Purpose of Document

The main purpose of this document is to provide sufficient information to understand what rkf45.ml is expected to do. The goals and theoretical models used in the rkf45.ml generator are provided, as are assumptions and unambiguous definitions.

3.2 Scope of the Family

The responsibility of the family is solving the provided ODEs for given intervals, initial values and step sizes using the specified RK method. The user will have to ensure the RK method specified is appropriate for their particular problem(s), and they will have to ensure appropriate and correct input is provided (see Assumptions). Additionally, the user will also have to process the output appropriately.

3.3 Characteristics of Intended Reader

The reader is expected to have some undergraduate STEM and functional programming background. Ideally, they have been exposed to some calculus and programming courses.

3.4 Organization of Document

The organization of this document follows the template for a commonality analysis (CA) provided by Dr. W. S. Smith, which is in turn adapted from Lai (2004),Smith and Lai (2005), Smith (2006) and Smith et al. (2007).

The structure of this document is essentially top-down, with detail added as one proceeds through it. A general system description is provided first, followed by commonalities and variabilities. References are listed at the end.

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 than external entity outside the software, which is the "user", or parent program making use of the generator as a library. A rectangle is used to represent the generator as well as the rkf45.ml family members it can generate.

Ideally, the parent program knows the ODE(s) and other inputs at compile time, and uses each generated family member many times at run-time to maximize the benefit of code generation. Alternatively, the ODE(s) or other inputs are not known at compile time; code generation can still happen at run-time, but hopefully the parent program can offset the incurred cost of this on-the-fly generation by running the optimized generated code enough times.

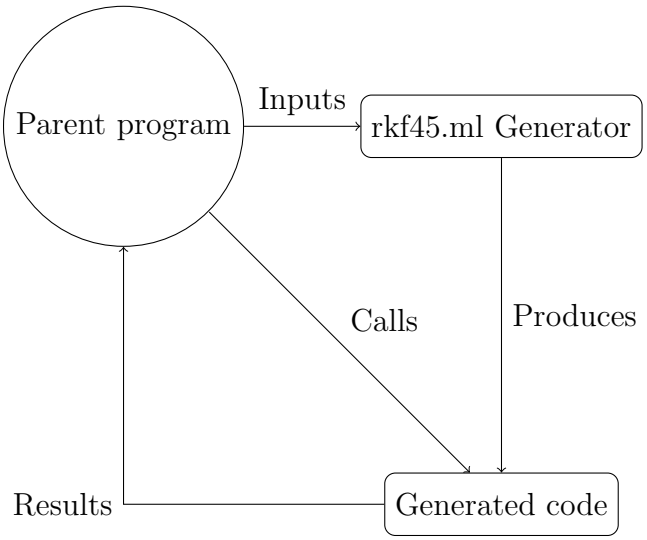

Figure 1: System Context

- User Responsibilities:
  - Provide appropriate input to the system
  - Ensure assumptions on the input are met (see subsection 6.1)
  - Correctly process the resulting output
- rkf45.ml Responsibilities:
  - Calculate the correct output for the given input

4.2 Potential User Characteristics

The most common user of rkf45.ml will be other programs. However, one or more programmers are needed to write the code that calls the functions provided by this family. These

programmers therefore should have an understanding of undergraduate calculus and OCaml programming. (Knowledge of MetaOCaml should not be necessary if the program family is able to hide its usage thereof sufficiently).

4.3 Potential System Constraints

The responsibility of generating family members in a type-safe way restricts the system to the MetaOCaml extension of the OCaml programming language.

5 Commonalities

This section contains the commonalities between the rkf45.ml family members. This section starts with succinct background information about the problem, followed by terminology and definitions, data definitions, goal statements and ending with theoretical models.

5.1 Background Overview

There are various numerical methods for approximating ordinary differential equation (ODE) given initial values. This problem is called an initial value problem (IVP, see T1). A well-known way to solve these problems is through Runge-Kutta methods.

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:

- Code generation: producing code programmatically.
- Compile-time: the time during which a program is compiled.
- Continuous function: a function that evaluates to a value that is not infinity or undefined for any input.
- Initial value problem (IVP): an ODE along with initial values for which a numerical approximation must be found.
- Initial values (initial condition): a starting point from which the rest of the approximation can be calculated. (See DD3.)
- Interval: interval inside which the ODE needs to be approximated. (See DD2.)
- MetaOCaml: an extension to the OCaml programming language which adds staging annotations that enable MSP.

- Metaprogramming: programming software that takes in and/or produces programs.
- Multi-stage programming (MSP): a form of metaprogramming where the execution of specific code fragments is delayed; these fragments are then combined into larger fragments and ultimately executed.
- OCaml: A multi-paradigm programming language (imperative & functional) with a syntax that will appear somewhat unusual to most.
- Ordinary differential equation (ODE): an equation that involves some ordinary (rather than partial) derivatives of a function. The goal is to find the function that satisfies the equation, and this is not always easily achievable through integration. (See DD1.)
- Runge-Kutta methods: a set of methods that approximate ODEs in IVPs for a given interval and step size.
- Run-time: the time during which a program is run.
- Step size: the distance between any two points for which to find approximations; it essentially determines how many points there will be once the interval is known. Once these points have been approximated, one could interpolate so that the whole interval is solved for. (See also DD4.)
- Stiff ODE: an ODE for which certain numerical methods are numerically unstable unless the step size is extremely small. There is no exact definition.

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	**Ordinary Differential Equation (ODE)**
Symbol	$\mathbf{f}$
Units	N/A
Equation	$\mathbf{x}' = \mathbf{f}(t, \mathbf{x}(t))$
Description	$\mathbf{f} : \mathbb{R} \times \mathbb{C}^n \to \mathbb{C}^n$ is the equation for which we ultimately want to find a numerical approximation.
Sources	Corless and Fillion (2013)
Ref. By	IM1, IM2, T1

Number	DD2
Label	**Interval**
Symbol	t_0, t_N
Units	N/A
Equation	N/A
Description	$[t_0, t_N] : \mathbb{R} \times \mathbb{R}$ is the interval for which to solve the ODE (see DD1). t_0 represents the beginning of the interval, t_N the end.
Sources	Corless and Fillion (2013)
Ref. By	IM1, IM2

Number	DD3
Label	**Initial values**
Symbol	$\mathbf{x}_0$
Units	N/A
Equation	$\mathbf{x}(t_0) = \mathbf{x}_0$
Description	$\mathbf{x}_0 : \mathbb{C}^n$ are initial values for solving ODE (see DD1).
Sources	Corless and Fillion (2013)
Ref. By	DD4, IM1, IM2, T1

Number	DD4
Label	**Step size**
Symbol	h
Units	N/A
Equation	$t_{k+1} - t_k = h$
Description	$h : \mathbb{R}$ is the distance between the points within interval $[t_0, t_N]$ for which to find approximations.
Sources	Corless and Fillion (2013)
Ref. By	IM1, IM2

5.4 Goal Statements

Given the non-stiff continuous ODE, the goal statements are:

GS1: Given an RK method specification, an interval and the desired step size, as well as an initial value, calculate a spline and generate (in a type-safe manner) a function that uses this spline to solve for specific points within the provided interval.

5.5 Theoretical Models

This section focuses on the general equations and laws that rkf45.ml is based on.

Number	T1
Label	**Initial value problem (IVP)**
Equation	$\mathbf{x}' = \mathbf{f}(t, \mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0$ where • $\mathbf{x} : \mathbb{R} \to \mathbb{C}^n$ is the vector solution as a function of time • $\mathbf{x}_0 \in \mathbb{C}^n$ is the initial condition (see DD3) • $\mathbf{f} : \mathbb{R} \times \mathbb{C}^n \to \mathbb{C}^n$ is the function describing the vector field (see DD1) Given an initial value $\mathbf{x}_0$, the goal is to find approximations on a given interval $[t_0..t_N]$.
Description	The standard form of an initial value problem is given above. The issue is that many IVPs are difficult to solve manually (or programmatically) and the correct solutions are often unknown. Numerical methods are close enough to be used in most applications.
Source	Corless and Fillion (2013) p. 510, 513
Ref. By	IM1, IM2

6 Variabilities

Assumptions on all variations of the program family are covered first. Variability in RK method in the rkf45.ml program family are represented by instance models in Calculation. Every unique input will result in slightly different code being generated, which could in turn produce different output compared to other family members. This allows for an infinite program family depending on one's definition of program family[2]. After instance models, the lack of variability in output is covered.

6.1 Assumptions

A1: The user will specify what order of RK method they want used during code generation, either second (IM1) or fourth (IM2).

A2: The ODE (DD1) provided will be continuous and non-stiff (otherwise the user accepts that the results will be inaccurate)

A3: The given initial values (DD3) are for the beginning of the interval (DD2), represented by t_0.

[2]Currently Dr. Smith and Dr. Carette agree they should discuss this definition to compare their overlapping but slightly differing viewpoints.

A4: Initial value (DD3) vector size is expected to match the ones produced by the ODE (DD1).

A5: The interval's bounds (DD2) satisfy $t_0 < t_N$ (both real numbers) and $t_N - t_0 = n \times h$.

A6: The step size (DD4) shall be a positive real number.

6.2 Calculation

The family can utilize either of the instance models listed below for any of its members.

Number	IM1
Label	**Fourth order Runge-Kutta method (RK4)**
Equation	• $t_1 = t_0 + h$ • $\mathbf{k}_1 = \mathbf{f}(t_0, \mathbf{x}_0)$ slope at x_0 • $\mathbf{k}_2 = \mathbf{f}(t_0 + \frac{h}{2}, \mathbf{x}_0 + \frac{h}{2}\mathbf{k}_1)$ slope of the point halfway between t_0 and t_1 when extrapolating slope $\mathbf{k}_1$ from point $\mathbf{x}_0$ • $\mathbf{k}_3 = \mathbf{f}(t_0 + \frac{h}{2}, \mathbf{x}_0 + \frac{h}{2}\mathbf{k}_2)$ slope of the point halfway between t_0 and t_1 when extrapolating slope $\mathbf{k}_2$ from point $\mathbf{x}_0$ • $\mathbf{k}_4 = \mathbf{f}(t_0 + h, \mathbf{x}_0 + h\mathbf{k}_3)$ slope of point at t_1 when extrapolating slope $\mathbf{k}_3$ from point $\mathbf{x}_0$ • $\mathbf{x}_1 = \mathbf{x}_0 + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$ new point created when extrapolating from point $\mathbf{x}_0$ using a weighted average of the previously calculated slopes
Description	The above equations can be used to calculate (an approximation of) a new point given the previous or starting point. Note that the above is a specific instance (for simplicity) of the general method (namely, the first step). Calculating a new point is necessary to create a spline to solve IVPs (T1). In addition to the ODE (DD1), this also requires an interval (DD2), step size (DD4) and initial condition (DD3).
Source	Corless and Fillion (2013) p. 618
Ref. By	N/A

Number	IM2
Label	**Second-order Runge-Kutta method (Improved Euler method)**
Equations	• $\mathbf{Y}_1 = \mathbf{x}_k + h\mathbf{f}(t_k, \mathbf{x}_k)$ • $\mathbf{x}_{k+1} = \mathbf{x}_k + h\left(\frac{1}{2}\mathbf{f}(t_k, \mathbf{x}_k) + \frac{1}{2}\mathbf{f}(t_{k+1}, \mathbf{Y}_1)\right)$
Description	The above equations can be used to calculate (an approximation of) a new point given the previous or starting point. Calculating a new point is necessary to create a spline to solve IVPs (T1). In addition to the ODE (DD1) this also requires an interval (DD2), step size (DD4) and initial condition (DD3).
Source	Corless and Fillion (2013) p. 616
Ref. By	N/A

6.3 Output

N/A. Every program family member will output a numerical approximation of an ODE in the form of a function that can be called by the user.

7 Traceability Matrices and Graphs

The traceability matrices below show the relationships between the various concepts defined in this document. Rows may be affected by changes in the items the columns consist of (especially in the context of assumptions.)

	DD1	DD2	DD3	DD4	T1	IM1	IM2
DD1					✓		
DD2					✓		
DD3	✓				✓		
DD4		✓			✓		
T1							
IM1	✓	✓	✓	✓	✓		
IM2	✓	✓	✓	✓	✓		

Table 1: Traceability matrix between instance models, data definitions and theory

	A1	A2	A3	A4	A5	A6
DD1		✓				
DD2					✓	
DD3			✓	✓		
DD4						✓
T1			✓			
IM1		✓				
IM2		✓				

Table 2: Traceability matrix for assumptions

References

Robert M. Corless and Nicolas Fillion. *A Graduate Introduction to Numerical Methods*. Springer New York, New York, NY, 2013. ISBN 978-1-4614-8452-3 978-1-4614-8453-0. URL http://link.springer.com/10.1007/978-1-4614-8453-0. DOI: 10.1007/978-1-4614-8453-0.

Lei Lai. Requirements documentation for engineering mechanics software: Guidelines, template and a case study. Master's thesis, McMaster University, Hamilton, Ontario, Canada, 2004.

W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006. URL http://www.ifi.unizh.ch/req/events/RE06/.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Àgerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.