Rythmos: Solution and Analysis Package for Differential-Algebraic and Ordinary-Differential Equations

Curtis C. Ober, Numerical Analysis and Applications
Roscoe A. Bartlett, Oak Ridge National Laboratories
Todd S. Coffey, Simulation Modeling Sciences
Roger P. Pawlowski, Multiphysics Simulation Technology

September 19, 2013

Abstract

Time integration is a central component for most transient simulations. It coordinates many of the major parts of a simulation together, e.g., a residual calculation with a transient solver, solution with the output, various operator-split physics, and forward and adjoint solutions for inversion. Even though there is this variety in these transient simulations, there is still a common set of algorithms and procedures to progress transient solutions for ordinary-differential equations (ODEs) and differential-alegbraic equations (DAEs).

Rythmos is a collection of these algorithms that can be used for the solution of transient simulations. It provides common time-integration methods, such as Backward and Forward Euler, Explicit and Implicit Runge-Kutta, and Backward-Difference Formulas. It can also provide sensitivities, and adjoint components for transient simulations. Rythmos is a package within Trilinos, and requires some other packages (e.g., Teuchos and Thrya) to provide basic time-integration capabilities. It also can be coupled with several other Trilinos packages to provide additional capabilities (e.g., AztecOO and Belos for linear solutions, and NOX for non-linear solutions).

This document is broken down into three parts: Theory Manual, User's Manual, and Developer's Guide. The Theory Manual contains the basic theory of the time integrators, the nomenclature and mathematical structure utilized within Rythmos, and verification results demonstrating that the designed order of accuracy is achieved. The User's Manual provides information on how to use the Rythmos, description of input parameters through Teuchos Parameter Lists, and description of convergence test examples. The Developer's Guide is a high-level discussion of the design and structure of Rythmos to provide information to developers for the continued development of capabilities. Details of individual components can be found in the Doxygen webpages.

This document is still under construction, and only provides some information on its current capabilities. However several pieces of information would be very useful to the Rythmos community, and was felt that a release of the current documentation would be beneficial. Documentation is continuing, and will be included in subsequent versions. The reader is encouraged to contact one of the authors for additional information not found within this document.

Acknowledgment

The first author would like to thank several individuals for their help while he has been learning not only Rythmos but also the Trilinos environment: Andy Salinger, Roger Pawlowski, Todd Coffey, Ross Bartlett, and Brent Perschbacher. Their guidance and helpful suggestions have greatly improved the overall experience of learning and struggles of this complex set of codes.

Contents

Part I
Theory Manual

1 Introduction

Here we design and describe a set of C++ interfaces and concrete implementations for the solution of a broad class of transient ordinary differential equations (ODEs) and differential algebraic equations (DAEs) in a consistent manner.

2 Mathematical Formulation of DAEs/ODEs for Basic Time Steppers

Here we describe the basic mathematical form of a general nonlinear DAE (or ODE) for the purpose of presenting it to a forward time integrator. At the most general level of abstraction, we will consider the solution of fully implicit DAEs of the form

$$\begin{aligned} f(\dot{x}, x, t) &= 0, \text{ for } t \in [t_0, t_f], \quad (1) \\ x(t_0) &= x_0 \quad (2) \end{aligned}$$

where

$x \in \mathcal{X}$ is the vector of differential state variables,
$\dot{x} = d(x)/d(t) \in \mathcal{X}$ is the vector of temporal derivatives of x,
$t, t_0, t_f \in \mathbb{R}$ are the current, initial, and the final times respectively,
$f(\dot{x}, x, t) \in \mathcal{X}^2 \times \mathbb{R} \to \mathcal{F}$ defines the DAE vector function,
$\mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the vector space of the state variables x, and
$\mathcal{F} \subseteq \mathbb{R}^{n_x}$ is the vector space of the output of the DAE function $f(\ldots)$.

Here we have been careful to define the Hilbert vector spaces $\mathcal{X}$ and $\mathcal{F}$ for the involved vectors and vector functions. The relevance of defining these vector spaces is that they come equipped with a definition of the space's scalar product (i.e. $< u, v >_{\mathcal{X}}$ for $u, v \in \mathcal{X}$) which should be considered and used when designing the numerical algorithms.

The above general DAE can be specialized to more specific types of problems based on the nature of Jacobian matrices $\partial f/\partial \dot{x} \in \mathcal{F}|\mathcal{X}$ and $\partial f/\partial x \in \mathcal{F}|\mathcal{X}$. Here we use the notation $\mathcal{F}|\mathcal{X}$ to define a linear operator space that maps vectors from the vector space $\mathcal{X}$ to the vector space $\mathcal{F}$. Note that the adjoint of such linear operators are defined in terms of these vector spaces (i.e. $< Au, v >_{\mathcal{F}} = < A^T v, u >_{\mathcal{X}}$ where $A \in \mathcal{F}|\mathcal{X}$ and A^T denotes the adjoint operator for A). Here we assume that these first derivatives exist for the specific intervals of $t \in [t_0, t_f]$ of which such an time integrator algorithm will be directly applied the the problem.

The precise class of the problem is primarily determined by the nature of the Jacobian $\partial f/\partial \dot{x}$:

- $\partial f/\partial \dot{x} = I \in \mathcal{F}|\mathcal{X}$ yields an explicit ODE
- $\partial f/\partial \dot{x}$ full rank yields an implicit ODE
- $\partial f/\partial \dot{x}$ rank deficient yields a general DAE.

In addition, the ODE/DAE may be linear or nonlinear and DAEs are classified by their index [3]. It is expected that a DAE will be able to tell a integrator what type of problem it is (*i.e.*, explicit ODE, implicit ODE, general DAE) and which, if any, of the variables are linear in the problem. This type of information can be exploited in a time integration algorithm.

Another formulation we will consider is the semi-explicit DAE formulation:

$$\begin{aligned} \dot{y} &= f(y, z, t) \text{ for } t \in [t_0, t_f], \\ 0 &= g(y, z, t) \text{ where } x = [y, z], \quad (3) \\ x(t_0) &= x_0. \end{aligned}$$

For each transient computation, the formulation will be cast into the general form in (1)–(2) to demonstrate how this general formulation can be exploited in many different contexts.

It is important for DAE initial-value problems that the initial values of the solution, x_0, and time derivative, $\dot{x}_0$, satisfy the DAE residual[4]

$$f(\dot{x}_0, x_0, t_0) = 0$$

3 Formulation for Explicit Time Steppers for ODEs

Explicit integration methods are primarily only attractive for non-stiff explicit and implicit ODEs but some classes of DAEs can be considered as well (*i.e.*, by nonlinearly eliminating the algebraic variables from the semi-explicit DAE formulation, Eq. (3) [3]). For this discussion, we will also assume that the DAE has been written in the explicit ODE form (*i.e.*, $\partial f/\partial \dot{x} = I$). Note that implicit ODEs can always be written as explicit ODEs by multiplying the implicit ODE from the left with $(\partial f/\partial \dot{x})^{-1}$ as

$$\begin{aligned} f(\dot{x}, x, t) &= 0 \\ &\Rightarrow \\ \frac{\partial f}{\partial \dot{x}}\dot{x} + \hat{f}(x,t) &= 0 \\ &\Rightarrow \\ \left(\frac{\partial f}{\partial \dot{x}}\right)^{-1}\left(\frac{\partial f}{\partial \dot{x}}\dot{x} + \hat{f}(x,t)\right) &= 0 \\ &\Rightarrow \\ \dot{x} &= -\left(\frac{\partial f}{\partial \dot{x}}\right)^{-1}\hat{f}(x,t) \\ &= \bar{f}(x,t) \end{aligned}$$

where $\dot{x} = \bar{f}(x,t)$ is the new explicit form of the ODE that is considered by the explicit time integration strategies below. The above transformation of course requires that the matrix $(\partial f/\partial \dot{x})^{-1}$ be fairly easy to invert.

3.1 Forward Euler

Forward Euler (Explicit Euler) is simply obtained by first-order differencing the time derivative

$$\frac{x_n - x_{n-1}}{\Delta t} = \bar{f}(x,t)$$

or as an update formula

$$x_n = x_{n-1} + \Delta t\, \bar{f}(x,t).$$

Because of the first-order approximation, Forward Euler is first-order accurate as can be seen in the global-convergence plot in Fig. (1).

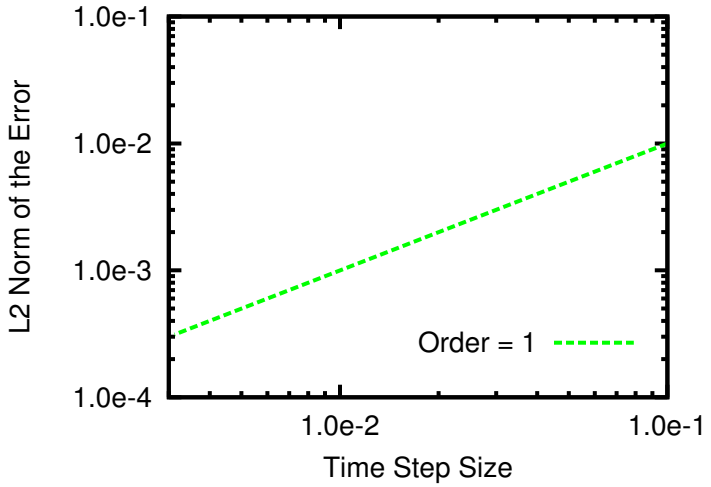


Figure 1: Order of accuracy for the SinCos Problem (Section 6.2) using Forward Euler.

3.2 Explicit Runge-Kutta Methods

The general Runge-Kutta method for s-stages, can be written as

$$X_i = x_{n-1} + \Delta t \sum_{j=1}^{s} a_{ij}\, \bar{f}(X_j, t_{n-1} + c_j \Delta t)$$

$$x_n = x_{n-1} + \Delta t \sum_{i=1}^{s} b_i\, \bar{f}(X_i, t_{n-1} + c_i \Delta t)$$

where X_i are intermediate approximations to the solution at times, $t_{n-1} + c_i \Delta t$, (*stage solutions*) which may be correct to a lower order of accuracy than the solution, x_n. We should note that these lower-order approximations are combined through b_i so that error terms cancel out and produce a more accurate solution [2, p. 80]. One can also write this in terms of $\dot{X}_i$ (or $\bar{f}(x,t)$)

$$\dot{X}_i = \bar{f}\left(x_{n-1} + \Delta t \sum_{j=1}^{s} a_{ij}\, \dot{X}_j, t_{n-1} + c_i \Delta t \right)$$

$$x_n = x_{n-1} + \Delta t \sum_{i=1}^{s} b_i\, \dot{X}_i$$

A convenient method to convey Runge-Kutta methods is to use the Butcher Tableau, which displays the coefficients in a "table" form.

Table 1: Schematic for a Butcher Tableau.

c_1	a_{11}	a_{12}	$\dots$	a_{1s}
c_2	a_{21}	a_{22}	$\dots$	a_{2s}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_s	a_{s1}	a_{s2}	$\dots$	a_{ss}
	b_1	b_2	$\dots$	b_s

Notes:

1. c_i is the fractional time step that the approximate solution X_i is known. It is possible for c_i to be outside the time step range [0,1], however it is odd for a One-Step methods like Runge-Kutta to have stage solutions outside the current time step.
2. $c_i = \sum_{j=1}^{s} a_{ij}$ for $i = 1, \dots, s$
3. For explicit methods, *e.g.*, Forward Euler and Explicit RK4, $c_1 = 0$ and $a_{1j} = 0$ for all j indicates that one needs the solution, x_{n-1}, and its time derivative, $\dot{x}_{n-1}$, (or basically an evaluation of $\bar{f}(x_{n-1}, t_{n-1})$) to start the time step.
4. If $a_{ij} = 0$ for $j \geq i$, the Runge-Kutta (RK) method is explicit (also known as ERK), since each X_i is given in terms of known quantities.
5. If $a_{ij} \neq 0$ for any $j \geq i$, the Runge-Kutta (RK) method is implicit (also known as IRK), since an implicit solve is required for at least some of the stages.
6. If $a_{ij} = 0$ for $j > i$, the method is known as a Diagonally Implicit Runge-Kutta (DIRK) method. DIRK methods require an implicit solution for each stage, but are not coupled to other stages.

7. If $a_{ij} = 0$ for $j > i$ and $a_{ii} = C$, the method is known as a Singly Diagonally Implicit Runge-Kutta (SDIRK) method. Like with DIRK methods, an implicit solve is needed at each stage, but since a_{ii} is the same, some of the preportory calculations can be reused for each stage.

8. b_i are the weighting of the intermediate solutions, X_i, to obtain the final solution, and they are a partition of unity, $\sum_{i=1}^{s} b_i = 1$.

9. If $a_{sj} = b_j$ for all j or $a_{i1} = b_1$ for all i, and a_{ij} is a nonsingular matrix, then A-stable RK methods are L-stable [9, p. 45][2, p. 103], *e.g.*, Backward Euler and two SDIRK methods. Each of the above conditions are not a necessary condition for L-stability, but they are a sufficient condition.

3.3 Explicit RK Forward Euler

Forward Euler can also be written for Runge-Kutta methods, Fig. 2, and obtains convergence results similar to Forward Euler, Section 3.1.

a)
$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

b)

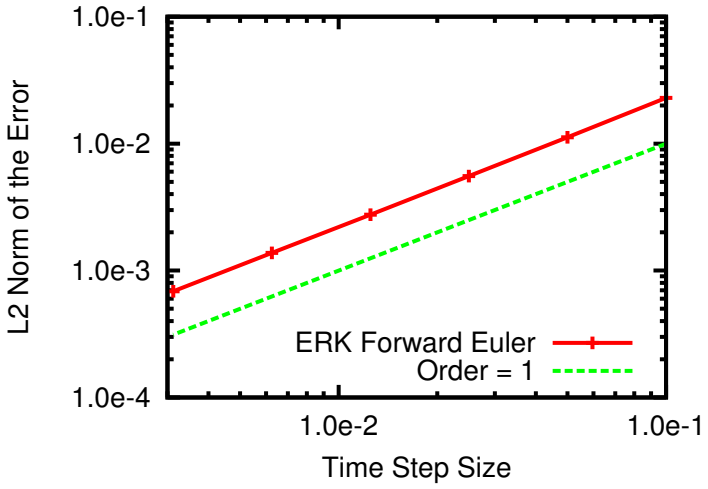


Figure 2: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK Forward Euler.

3.4 Explicit RK 2 Stage 2 Order by Runge (Explicit Midpoint)

a)

0	0	
1/2	1/2	0
	0	1

b)

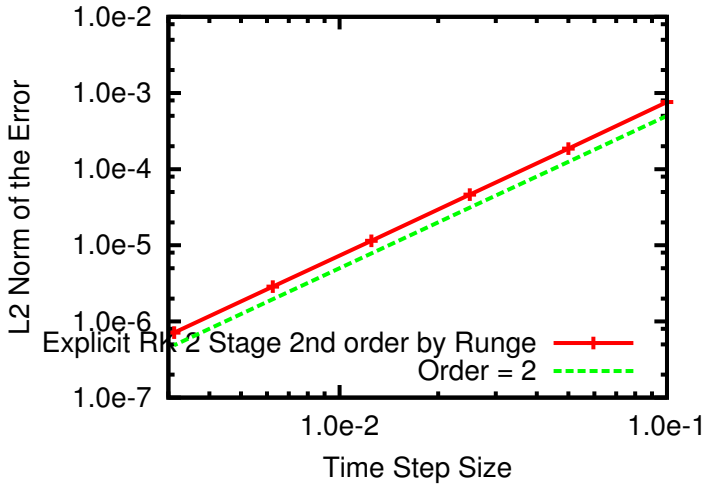


Figure 3: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK 2 Stage 2nd Order by Runge..

3.5 Explicit RK Trapezoidal

a)

0	0	
1	1	0
	1/2	1/2

b)

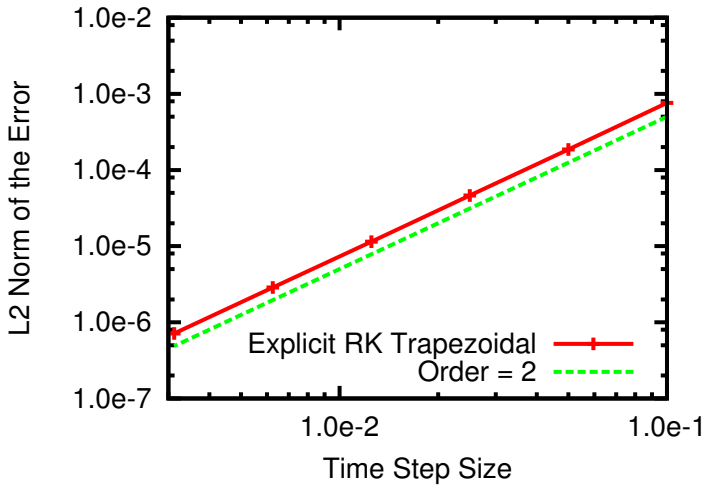


Figure 4: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK Trapezoidal.

3.6 Explicit RK 3 Stage 3 Order

a)

0	0		
1/2	1/2	0	
1	−1	2	0
	1/6	4/6	1/6

b)

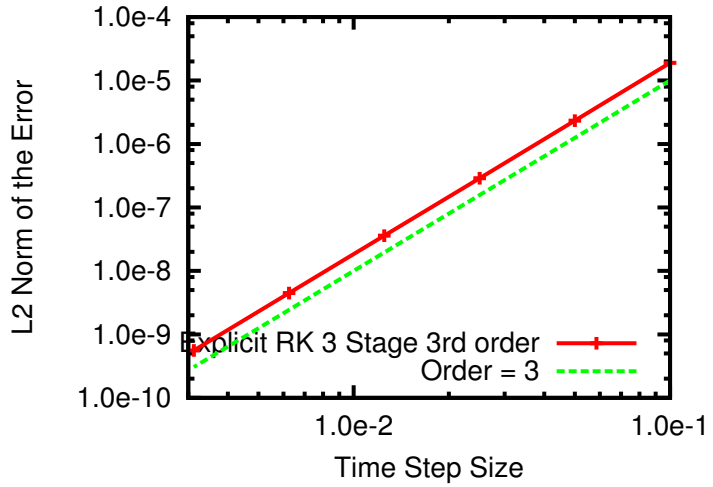


Figure 5: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK 3 Stage 3rd Order.

3.7 Explicit RK 3 Stage 3 Order by Heun

a)

0	0		
1/3	1/3	0	
2/3	0	2/3	0
	1/4	0	3/4

b)

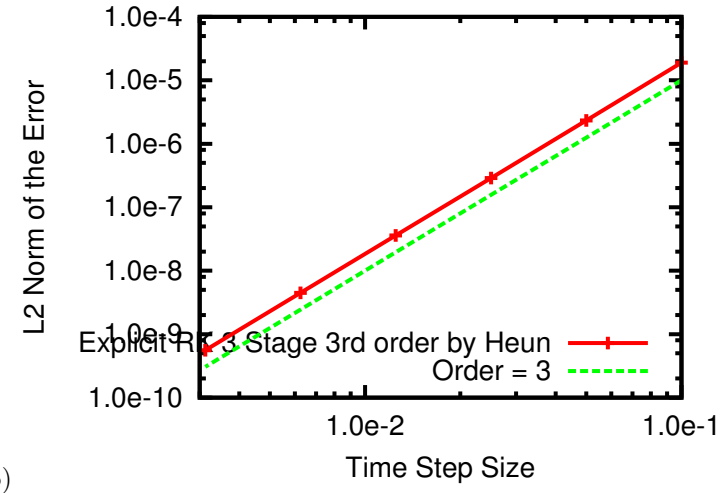


Figure 6: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK 3 Stage 3rd Order by Heun.

3.8 Explicit RK 3 Stage 3 Order TVD

a)

0	0		
1	1	0	
1/2	1/4	1/4	0
	1/6	1/6	4/6

b)

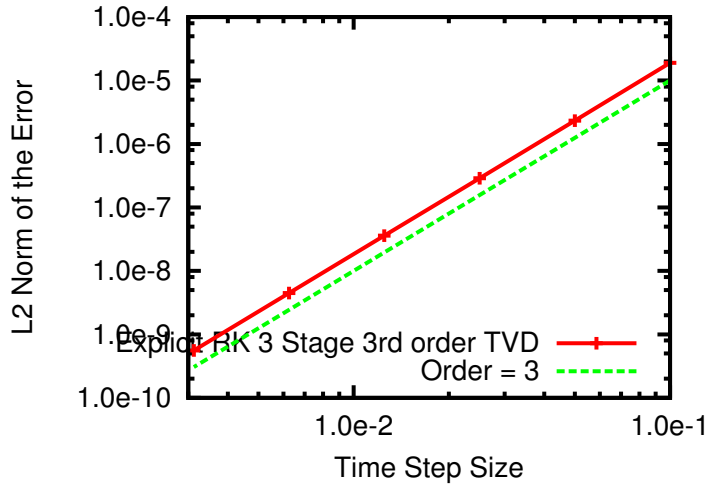


Figure 7: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK 3 Stage 3rd Order TVD.

3.9 Explicit RK 4 Stage 3 Order by Runge

a)

0	0				
1/2	1/2	0			
1	0	1	0		
1		0	1	0	
	1/6	2/3	0	1/6	

b)

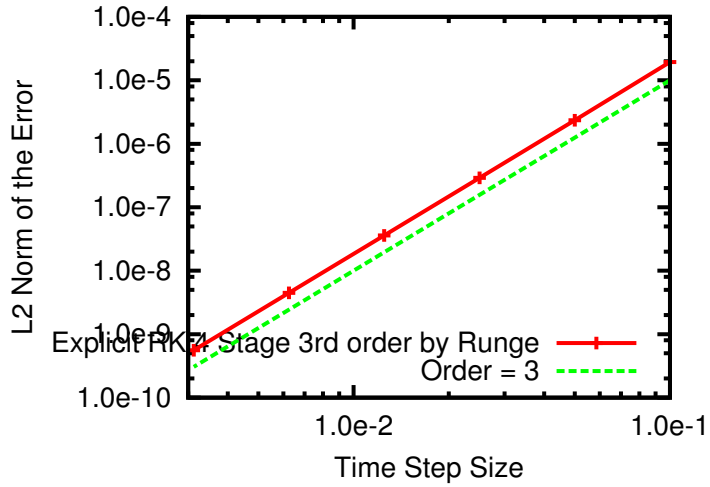


Figure 8: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK 4 Stage 3rd Order by Runge.

3.10 Explicit RK4

a)

0	0				
1/2	1/2	0			
1/2	0	1/2	0		
1	0	0	1	0	
	1/6	1/3	1/3	1/6	

b)

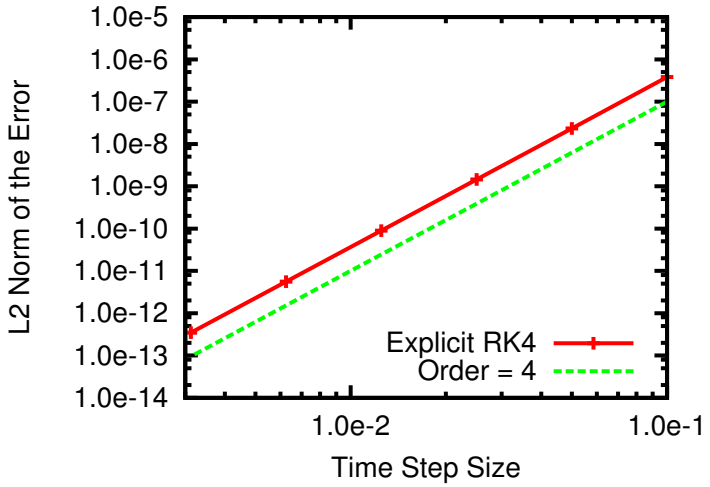


Figure 9: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK 4..

3.11 Explicit RK 3/8 Rule

a)

0	0				
1/3	1/3	0			
2/3	−1/3	1	0		
1	1	−1	1	0	
	1/8	3/8	3/8	1/8	

b)

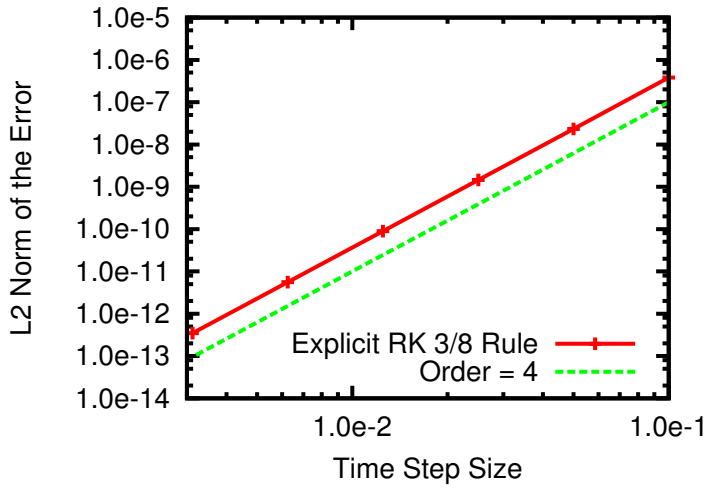


Figure 10: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Explicit RK 3/8 Rule.

4 Formulation for Implicit Time Steppers for ODEs and DAEs

Here we consider several different classes of implicit time stepping methods. For each class of method we show the set of general nonlinear equations that defines a single time step and then show how a linearized form of the equations may be formed to be solved by a Newton-type nonlinear equation solver.

In particular, for each method, we will show how to define a set of nonlinear equations of the form

$$r(z) = 0 \tag{4}$$

such that when solved will define an implicit time step from t_k to t_{k+1}, where $\Delta t = t_{k+1} - t_k$ denotes the time-step. In addition, for each method, we will show how to use the DAE residual evaluation $(\dot{x}, x, t) \to f$ to define the nonlinear time step equation (4). At the highest level, the time step method only requires very general convergence criteria for the time step equation (4) and therefore great flexibility is allowed in how the time step equation is solved. In general, the system in (4) must be solved such that $||x_{k+1} - x^*(t_{k+1})|| < \eta$, where $x_{k+1} \in \mathcal{X}$ is the computed step for the state, $x^*(t_{k+1}) \in \mathcal{X}$ is the exact state solution at t_{k+1}, and η is the maximum allowable local truncation error defined by the user.

Even though the time step equation can be solved by a variety of means, a large class of DAEs can also potentially provide support for a general Newton-type method for solving these equations and can therefore leverage general software for solving such problems (e.g. NOX). The foundation of Newton-like methods is the ability to solve linear systems similar to the Newton system

$$\frac{\partial r}{\partial z} \Delta z = -r(z_l) \tag{5}$$

where z_l is the current candidate solution of the nonlinear equations (which also defines the point where $\partial r / \partial z$ is evaluated) and $\Delta z = r_{l+1} - r_l$ is the update direction. Line-search Newton methods then define an update to the solution along the direction Δz. The essential functionality needed to perform a Newton-like method are the the abilities to evaluate the nonlinear residual $z \to r$ and to (approximately) solve linear systems involving the Jacobian matrix $\partial r / \partial z$. For each type of implicit time integration method, we will show, if possible, how to perform solves with $\partial r / \partial z$ by performing solves with the matrix

$$W = \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x}, \tag{6}$$

evaluated at points $(\dot{x}, x, t)$ selected by the time integration method and where $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$ is some constants also defined by the time integration method. Note that the matrix W above in (9) may not necessarily exactly represent $\partial r / \partial z$ and z and r may not simply lie in the vector spaces $\mathcal{X}$ and $\mathcal{F}$ respectively; but in many cases, they will.

The iteration matrix, W, is defined to be

$$W \equiv \frac{df}{dx_n} = \frac{\partial \dot{x}}{\partial x_n} \frac{\partial f}{\partial \dot{x}} + \frac{\partial x}{\partial x_n} \frac{\partial f}{\partial x} = \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x}$$

where $\alpha = \partial \dot{x} / \partial x_n$ and $\beta = \partial x / \partial x_n$, and recalling $f\left(\dot{x}(x_n), x(x_n)\right) = 0$.

4.1 Backward Euler

4.1.1 Constant Step Size

In Fig. 11, the order of accuracy is shown for Backward Euler for the SinCos problem.

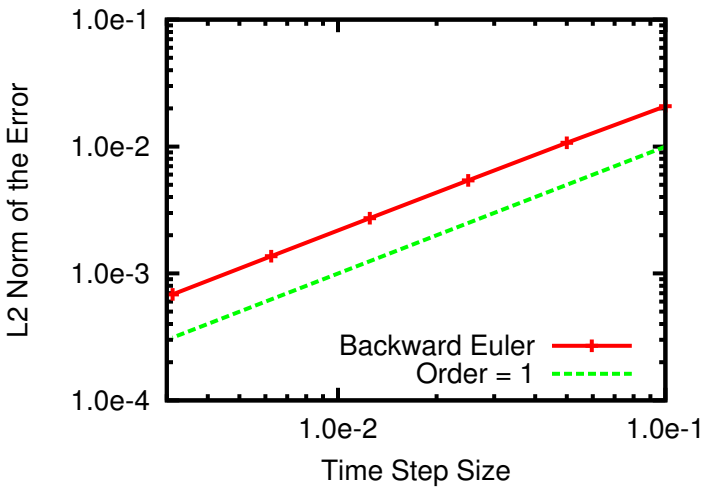


Figure 11: Order of accuracy for the SinCos Problem (Section 6.2) using Backward Euler.

4.1.2 Variable Step Size

A good test for variable step size is the Log-Time problem which is defined in Sec. 6.3. To adjust the step size, the FirstOrderErrorStepControlStrategy uses the difference between the predicted and final solution as an measure of the temporal error and sets the next step size to maintain the user specified level of error. Figure 12 shows this for the Backward Euler method for four different relative errors, $\epsilon_r = 10^{-2}$, 10^{-3}, 10^{-4}, and 10^{-5}.

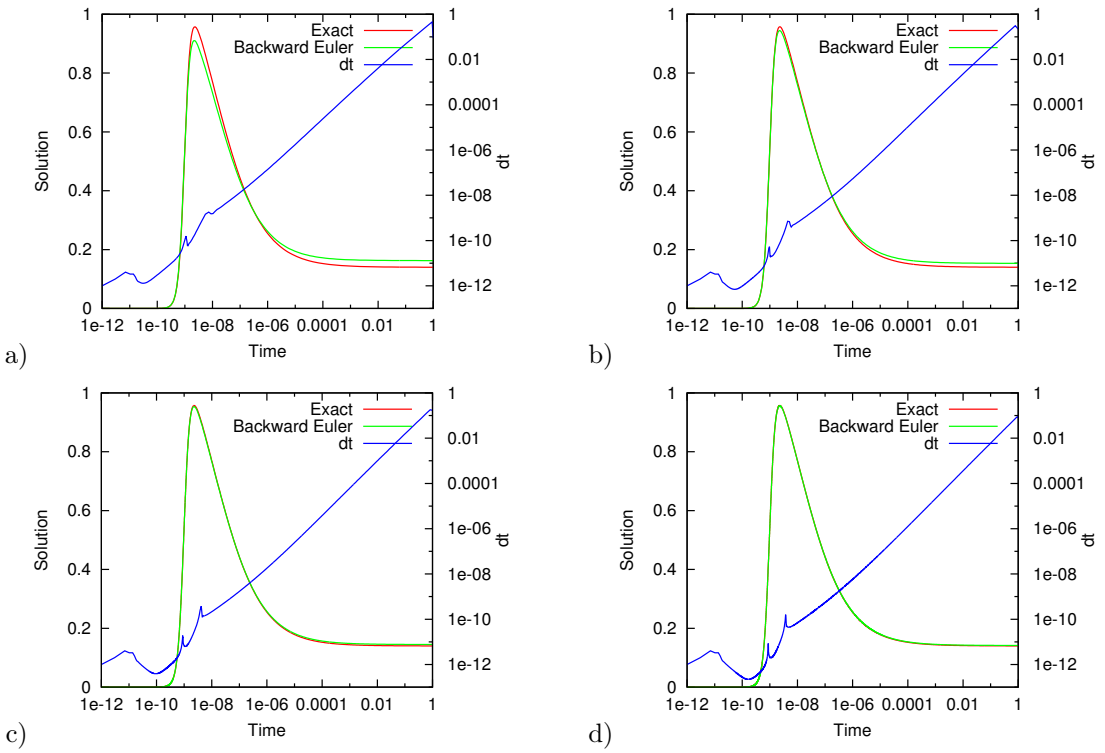


Figure 12: Solution and time step sizes for the Log-Time Problem (Section 6.3) using Backward Euler with various relative errors in the FirstOrderErrorStepControlStrategy: a) ϵ_r = a) 10^{-2}, b) 10^{-3}, c) 10^{-4}, and d) 10^{-5}.

Table 2: Log-Time Errors for Backward Euler with Variable Step Size for the Log-Time Problem.

ϵ_r	Number of Steps	Error Norm ($0 \leq t_n \leq 1$)	Global Error ($t = 1$)
10^{-2}	213	0.0224098	0.0224576
10^{-3}	563	0.0132322	0.0132634
10^{-4}	1534	0.00480765	0.00482358
10^{-5}	4168	0.00153523	0.00154173

In Table 2, the errors for this problem are shown. For each magnitude reduction in the relative-error specification, the number of steps increases by a factor of 2-3 and a similar reduction in the error norms and global errors.

The exponential growth in the time step size, Δt, over $10^{-8} < t < 1$ is approximately 8% ($\Delta t_n \approx 1.08 \Delta t_{n-1}$), which doubles the step size approximately every 10 time steps, and is unaffected by the relative error, ϵ_r. The primary effect of ϵ_r is to reduce Δt over the majority of the run, except for the initial few steps (~10). The two 'blips' near $t = 10^{-9}$ are evident with all ϵ_r, and correspond to the locations where the curvature in the solution are large.

4.2 Backward Difference Formulas

Backward Difference Formulas (BDF) are popular methods for stiff problems, and are derived by differentiating a polynomial representation of s past solutions, x_{n-i} for $i = 1, \ldots, s$, and setting the derivative $\dot{x}(t_n) = f(t_n, x_n)$. For evenly spaced intervals, $\Delta t = t_n - t_{n-1}$, an s-step BDF method (BDFs) is given by

$$\dot{x}_n = \bar{f}(t_n, x_n) = \frac{1}{\Delta t\, \beta_0} \sum_{i=0}^{s} \alpha_i\, x_{n-i} \tag{7}$$

where α_0 is normally scaled to one, and the order is equal to the number of steps, s.

The nonlinear time step equation to advance the solution from t_{n-1} to t_n is then formed by substituting $\dot{x} = \dot{x}_n$ in (7), $x = x_n$ and $t = t_n$ into (1) to obtain

$$f\left(\left[\frac{1}{\Delta t\, \beta_0} \sum_{i=0}^{s} \alpha_i\, x_{n-i}\right], x_n, t_n\right) = 0. \tag{8}$$

One can immediately identify the BDF time step equations (8) with the general form of the time step equations (4) and with unknown solution variables $z = x_n$. All of the other state variables x_{n-i}, for $i = 1 \ldots s$, are given.

Note that the first-order BDF method with $p = 1$, $\alpha_0 = 1$ and $\alpha_1 = -1$ is simply the standard backward Euler time integration method [2].

When considering a general Newton-like method for solving (8), note that the Newton Jacobian of these equations is

$$\frac{\partial r}{\partial z} = \frac{\alpha_0}{\Delta t\, \beta_0} \frac{\partial f}{\partial \dot{x}} + \frac{\partial f}{\partial x}, \tag{9}$$

which is evaluated at the point $\dot{x}$ in (7), $x = x_n$ and $t = t_n$. One can immediately identify (9) with the general form of the matrix M in (6) where $\alpha = \alpha_0/(\Delta t\, \beta_0)$ and $\beta = 1$. Note that the Jacobian (9) is the exact Jacobian for the nonlinear time step equations; this will not be true for some of the other methods.

There are three major variations of BDFs for variable time-step methods: fixed-coefficient formulas, variable-coefficient formulas, and fixed-leading-coefficient formulas. The fixed-coefficient formulas assume evenly spaced intervals so that the coefficients are fixed and can be precomputed for all orders[2, p. 130]. However when the time-step size changes the solution must be interpolated to evenly spaced intervals, and the amount of work for the interpolation is proportional to the number of equations in the system. Jackson and Sacks-Davis[11] have noted that fixed-coefficient formulas have worse stability properties than the other two variations. Also there can be an additional computational savings if $\partial f/\partial x$ is constant and the time step and order are consistently updated, fewer matrix factorizations and evaluations are possible.

For variable-coefficient formulas, the coefficients need to be recalculated if the time step has changed in the past s steps, which the amount of work depends on the number of steps but is independent of the number of equations. As noted earlier, variable-coefficient formulas have better stability properties which allow larger changes in time step and order during the solution of the problem. This can reduce the overall computational costs through larger time steps. It should be noted that the nonlinear matrix will need to be refactorized even if $\partial f/\partial x$ is constant, when the time step has changed in the past s steps thus increasing the of variable-coefficient formulas.

Variable-coefficient formulas would be the preferred approach except for the when $\partial f/\partial x$ is constant, when the time step has changed in the past s steps. Jackson and Sacks-Davis[11] noted that if the leading coefficient, α_0, is kept constant this exception can be removed, and these methods have been termed fixed-leading-coefficient formulas. thus is a hybrid between the two methods and tries to obtain the best of both worlds.

Fixed-Leading-Coefficient Formula[1] To begin, we need a predictor polynomial, $\phi_p(t)$, that uses the past solution values (possiblity at uneven intervals)

$$\phi_p(t_{n-i}) = x_{n-i} \quad \text{for } i = 1, \ldots, s$$

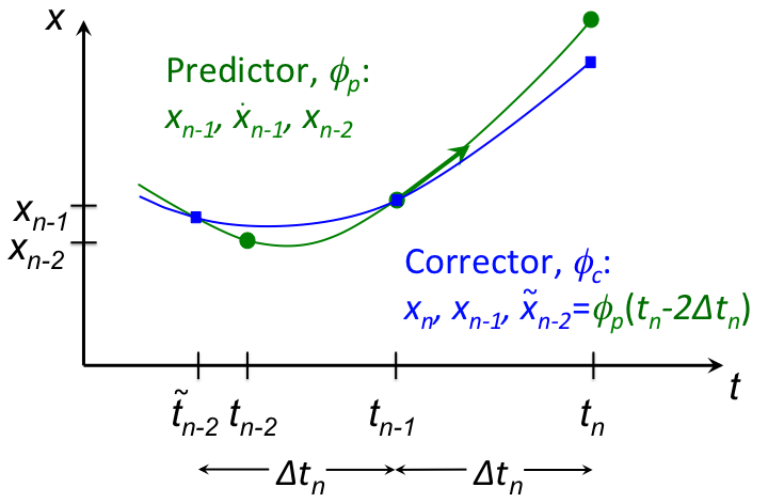


Figure 13: Schematic of a BDF2 method with predictor and corrector polynomials when the variable time stepping is increasing.

and the time derivative

$$\dot{\phi}_p(t_{n-i}) = \dot{x}_{n-i}$$

A corrector polynomial, $\phi_c(t)$, is constructed on even intervals of $\Delta t_n = t_n - t_{n-1}$, and is equal to $\phi_p(t)$ at those locations.

$$\begin{aligned} \phi_c(t_n) &= x_n \\ \phi_c(t_n - i\Delta t_n) &= \phi_p(t_n - i\Delta t_n) \quad \text{for} \quad i = 1, \ldots, s \end{aligned}$$

Thus ϕ_c passes through the unknown solution at t_n. The schematic of these polynomials are shown in Fig. 13 for a BDF2 method. Note that in this illustration, the variable time stepping is increasing, thus $t_n - 2\Delta t_n$ is further back in time than t_{n-2}.

With ϕ_p and ϕ_c being s-th order polynomials, they can be easily related through a new polynomial, $w(t)$, as

$$\phi_c(t) = \phi_p(t) + w(t)[\phi_c(t_n) - \phi_p(t_n)] \tag{10}$$

where

$$\begin{aligned} w(t_n) &= 1 \\ w(t_n - i\Delta t_n) &= 0 \quad \text{for} \quad i = 1, \ldots, s \end{aligned}$$

With these roots of a s-th order polynomial, one can write

$$w(t) = C \prod_{i=1}^{s} [t - (t_n - i\Delta t_n)]$$

and with $w(t_n) = 1$, the coefficient becomes $C = 1/(s!\,\Delta t_n^s)$.

Plugging the corrector, Eq. 10, into the ODE, Eq. 7, to generate the nonlinear equation to solve, and noting that $\phi_c(t_n) = x_n$ and $\phi_p(t_n) = x_{n(0)}$, one gets

$$\dot{\phi}_c(t) = \dot{\phi}_p(t) + \dot{w}(t)[x_n - x_{n(0)}].$$

Evaluating at $t = t_n$ to get the solution at the next time step and using $\dot{\phi}_c(t_n) = \dot{x}_n$, $\dot{\phi}_p(t_n) = \dot{x}_{n(0)}$, and $\dot{w}(t_n) = [1/1 + 1/2 + \ldots + 1/s]/\Delta t_n = 1/(\beta_0\,\Delta t_n)$, one gets

$$\dot{x}_n = \dot{x}_{n(0)} + \frac{1}{\beta_0\,\Delta t_n}[x_n - x_{n(0)}]$$

or

$$x_n = x_{n(0)} + \beta_0\,\Delta t_n[\dot{x}_n - \dot{x}_{n(0)}]$$

or

$$x_n - x_{n(0)} = \beta_0\,\Delta t_n[\dot{x}_n - \dot{x}_{n(0)}].$$

Note that the predictor and corrector are interwined. One should not try to use a different predictor as this will change the predictor polynomial, $\phi_p(t)$, and its presentation of past solution values. Also the predictor is dependent on the time derivative at the last time step, so one should be careful to keep the solution, x_{n-1}, and its time derivative, $\dot{x}_{n-1}$, used by the predictor in "sync" and consistent with previous time steps (e.g., clipping the solution between time steps will cause issues).

4.2.1 Convergence Test for Implicit BDF

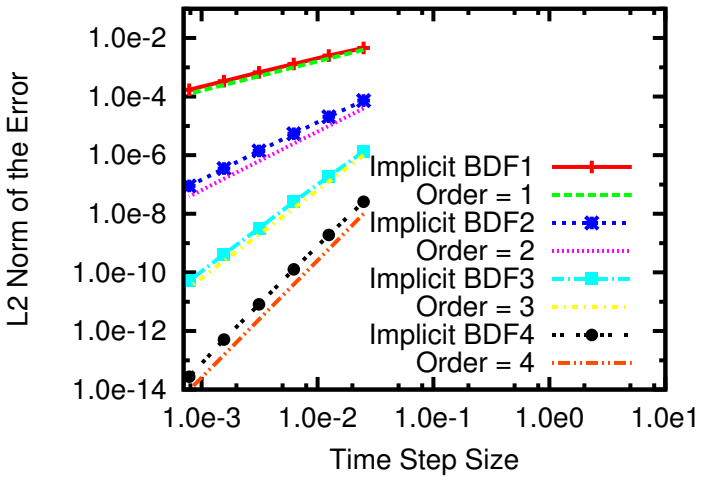


Figure 14: Order of accuracy for the SinCos Problem (Section 6.2) using Implicit BDF.

4.3 Implicit Runge-Kutta methods

We now consider a class of powerful and popular one-step methods for solving implicit DAEs, implicit Runge-Kutta (RK) methods. The most general form of implicit RK methods requires the simultaneous solution of s sets of coupled nonlinear equations that take the form

$$r_i(z) = f\left(\dot{X}_i,\ x_{n-1} + \Delta t \sum_{j=1}^{s} a_{ij}\,\dot{X}_j,\ t_{n-1} + c_i \Delta t\right) = 0 \tag{11}$$

for $i = 1, \ldots, s$ where $\dot{X}_i$ are essentially approximations to the derivatives $\dot{x}(t_{n-1} + c_i \Delta t)$ called *stage derivatives* and $z = [\dot{X}_1, \dot{X}_2, \ldots, \dot{X}_s]^T$ are the unknowns in this set of equations. After this set of coupled equations is solved, the state solution x_n is given as the linear combination

$$x_n = x_{n-1} + \Delta t \sum_{i=1}^{s} b_i\,\dot{X}_i$$

It is clear how to form the residual for the fully coupled system for $r(z)$ in Eq. (11) just from individual evaluations. How the Newton system for such a system is solved will vary greatly based on the structure and properties of the Butcher matrix, a_{ij}.

Fully implicit RK methods present somewhat of a problem for developing general software since they involve the need to solve a fully coupled system of s sets of equations of the form of Eq. (11). Each block $\partial r_i/\partial z_j = \partial r_i/\partial \dot{X}_j$ of the full Jacobian $\partial r/\partial z$ is represented as

$$\begin{aligned} W_{ij} &= \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x} \\ &= \frac{\partial r_i}{\partial z_j} = \frac{\partial r_i}{\partial \dot{X}_j} \\ &= \frac{\partial \dot{x}}{\partial \dot{X}_j}\frac{\partial f}{\partial \dot{x}} + \frac{\partial x}{\partial \dot{X}_j}\frac{\partial f}{\partial x} \\ &= \delta_{ij}\frac{\partial f}{\partial \dot{x}} + \Delta t\, a_{ij}\frac{\partial f}{\partial x} \end{aligned} \tag{12}$$

for $i = 1, \ldots, s$ and $j = 1, \ldots, s$ which is evaluated at the points $(\dot{x}, x, t) = (\dot{X}_i,\ x_{n-1} + \Delta t \sum_{j=1}^{s} a_{ij}\,\dot{X}_j,\ t_{n-1} + c_i \Delta t)$. Note that the iteration matrix, W, has $\alpha = \delta_{ij}$ and $\beta = \Delta t\, a_{ij}$.

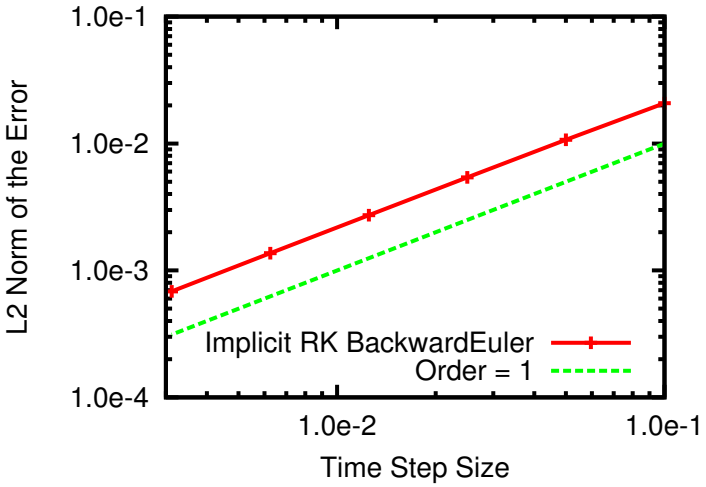
When considering a iterative method for solving systems with the block operator matrix $\partial r/\partial z$, it is easy to see how to use the iteration matrix W in Eq. (6) to implement a matrix-vector product, but it is not obvious how to precondition such a system. Clearly a block diagonal preconditioner could be used but the effectiveness of such a preconditioning strategy is open to question. Other preconditioning strategies are also possible just given the basic block operators and this is an open area of research. In some cases, however, it may be possible to form a full matrix object for $\partial r/\partial z$ but this is not something that can be expected for most applications.

Semi-explicit IRK methods Semi-explicit IRK methods are those IRK methods where the Butcher matrix, A, is lower diagonal and therefore gives rise to a block lower triangular Jacobian matrix $\partial r/\partial z$. For these types of methods, the nonlinear equations in Eq. (12) can be solved one at a time for $i = 1, \ldots, s$ which is easily accommodated using a Newton-type method where the Newton Jacobian for each i is given by Eq. (12), which is of our basic general form, Eq. (6).

Singly-Diagonal-implicit IRK methods The next specialized class of IRK methods that we consider are singly-diagonal-implicit IRK methods where the Butcher coefficients in a_{ij} and c give rise to a lower triangular Jacobian $\partial r/\partial z$ (and hence are also semi-explicit IRK methods) that has the same nonsingular matrix block of the form in Eq. (12) along the diagonal. This, of course, requires that $a_{11} = a_{22} = \cdots = a_{ss}$ and $c_1 = c_2 = \cdots = c_s$. (I am not sure that this is possible,

since $c_i = \sum_{j=1}^{s} a_{ij}$. The only method that satisfies this is IRK Backward Euler!? CCO) In this class of IRK methods, significant savings may be achieved since a single set of linear solver objects (*i.e.*, operators and preconditioners) can be utilized for the solution of the fully coupled system. In fact, it may even be possible to utilize multi-vector applications of the operator and preconditioner for matrices of the form Eq. (6) which can be supported by many applications.

4.3.1 Implicit RK Backward Euler

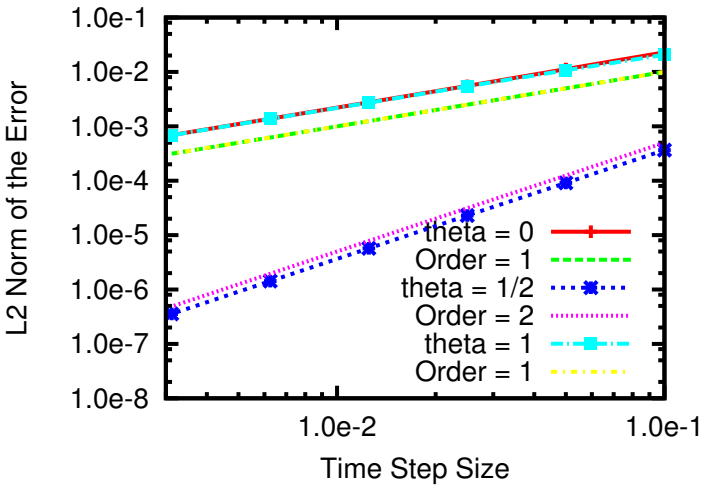


a) $\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$ b)

Figure 15: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for Implicit RK Backward Euler.

4.3.2 IRK 1 Stage Theta

This is a generalization of the midpoint method ($\theta = 1/2$).



a) $\begin{array}{c|c} \theta & \theta \\ \hline & 1 \end{array}$ b)

Figure 16: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for IRK 1 Stage Theta method.

4.3.3 IRK 2 Stage Theta

This is a generalization of the trapezoid method ($\theta = 1/2$).

a)
$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1-\theta & \theta \\ \hline & 1-\theta & \theta \end{array}$$

b)

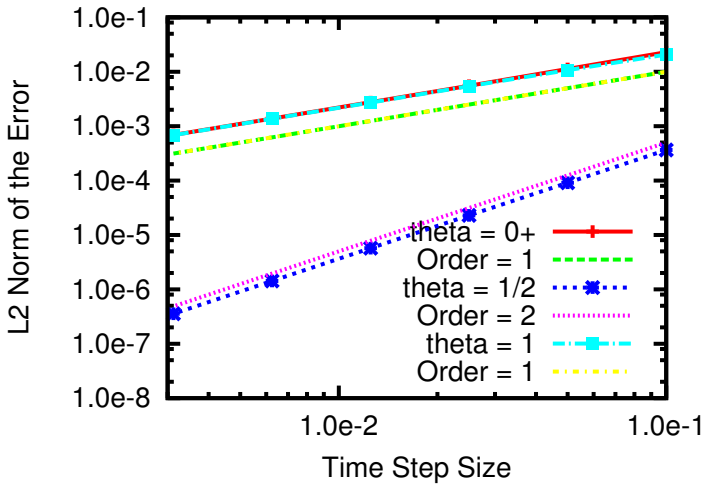


Figure 17: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for IRK 2 Stage Theta method.

4.3.4 SDIRK 2 Stage 2 Order

For $\gamma = (2 \pm \sqrt{2})/2$, this method is 2nd order accurate and L-stable. Other values of γ will still produce an L-stable scheme, but with only be 1st order accurate.

a)
$$\begin{array}{c|cc} \gamma & \gamma & \\ 1 & 1-\gamma & \gamma \\ \hline & 1-\gamma & \gamma \end{array}$$

b)

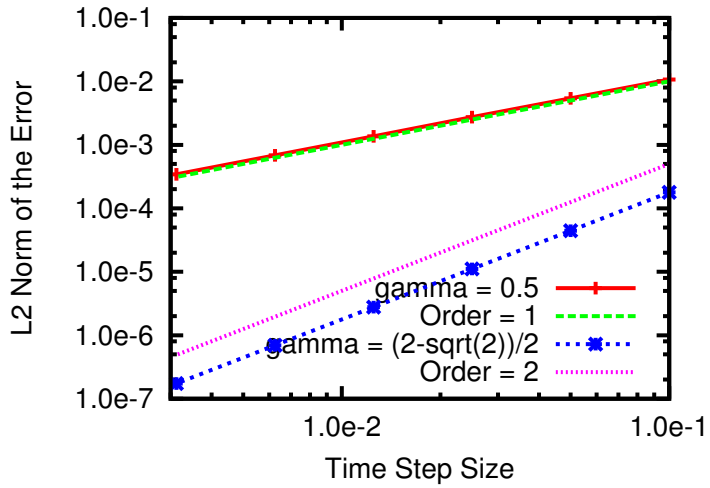


Figure 18: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for SDIRK 2 Stage 2 Order.

4.3.5 SDIRK 2 Stage 3 Order

For $\gamma = (3 \pm \sqrt{3})/6$, this method is 3rd order accurate and A-stable. For $\gamma = (2 \pm \sqrt{2})/2$, this method is only 2nd order accurate, but is then L-stable.

a)

γ	γ		
$1-\gamma$	$1-2\gamma$	γ	
	$1/2$	$1/2$	

b)

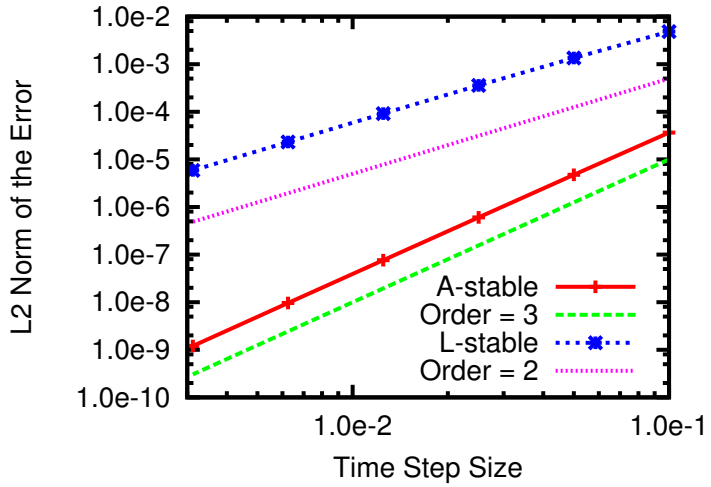


Figure 19: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for SDIRK 2 Stage 3 Order.

4.3.6 SDIRK 3 Stage 4 Order

The coefficients are $\gamma = 1/\sqrt{3}\cos(\pi/18) + 1/2$ and $\delta = (2\gamma - 1)^{-2}/6$.

a)

γ	γ			
$1/2$	$1/2-\gamma$	γ		
$1-\gamma$	2γ	$1-4\gamma$	γ	
	δ	$1-2\delta$	δ	

b)

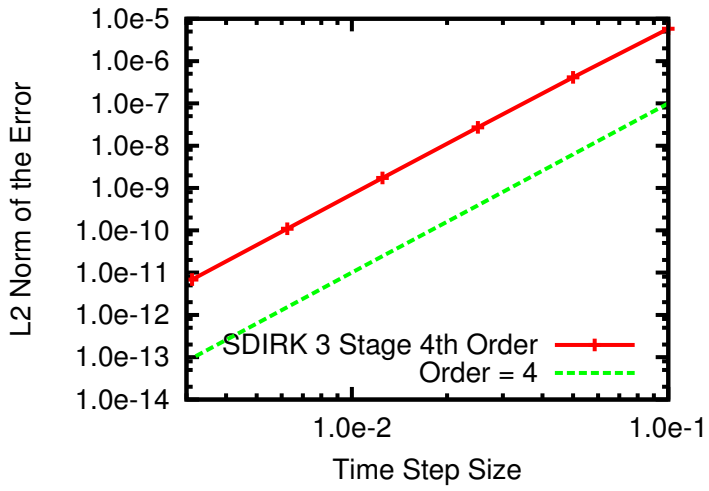


Figure 20: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for SDIRK 3 Stage 4 Order.

4.3.7 SDIRK 5 Stage 4 Order

1/4	1/4				
3/4	1/2	1/4			
11/20	17/50	−1/25	1/4		
1/2	371/1360	−137/2720	15/544	1/4	
1	25/24	−49/48	125/16	−85/12	1/4
	25/24	−49/48	125/16	−85/12	1/4
	59/48	−17/96	225/32	−85/12	0

Figure 21: Butcher Tableau for SDIRK 5 Stage 4 Order.

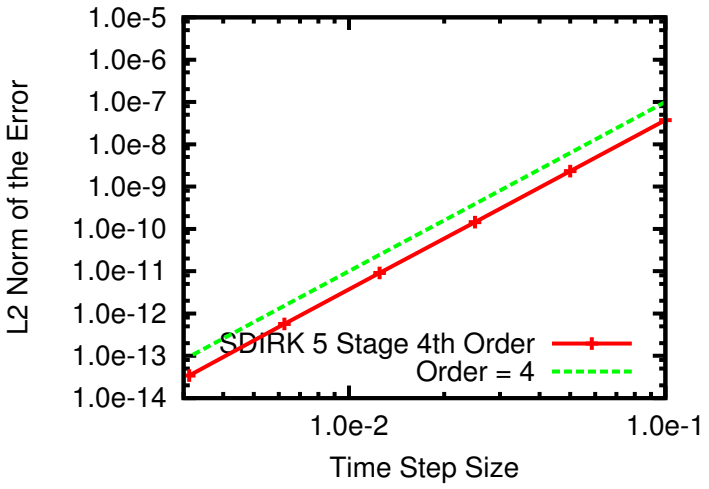


Figure 22: Order of accuracy for the SinCos Problem (Section 6.2) using SDIRK 5 Stage 4 Order.

4.3.8 SDIRK 5 Stage 5 Order

$\frac{6-\sqrt{6}}{10}$	$\frac{6-\sqrt{6}}{10}$				
$\frac{6+\sqrt{6}}{35}$	$\frac{-6+5\sqrt{6}}{14}$	$\frac{6-\sqrt{6}}{10}$			
1	$\frac{888+607\sqrt{6}}{2850}$	$\frac{126-161\sqrt{6}}{1425}$	$\frac{6-\sqrt{6}}{10}$		
$\frac{4-\sqrt{6}}{10}$	$\frac{3153-3082\sqrt{6}}{14250}$	$\frac{3213+1148\sqrt{6}}{28500}$	$\frac{-267+88\sqrt{6}}{500}$	$\frac{6-\sqrt{6}}{10}$	
$\frac{4+\sqrt{6}}{10}$	$\frac{-32583+14638\sqrt{6}}{71250}$	$\frac{-17199+364\sqrt{6}}{142500}$	$\frac{1329-544\sqrt{6}}{2500}$	$\frac{-96+131\sqrt{6}}{625}$	$\frac{6-\sqrt{6}}{10}$
	0	0	1/9	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$

Figure 23: Butcher Tableau for SDIRK 5 Stage 5 Order.

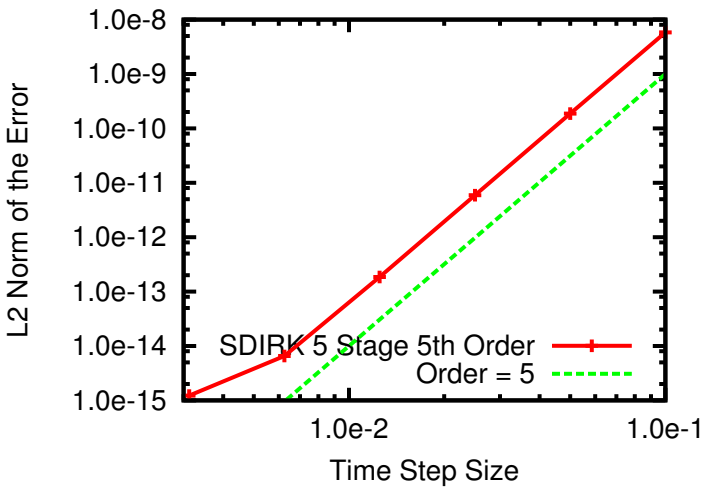


Figure 24: Order of accuracy for the SinCos Problem (Section 6.2) for SDIRK 5 Stage 5 Order.

4.3.9 DIRK 2 Stage 3 Order

a)

0	0		
2/3	1/3	1/3	
	1/4	3/4	

b)

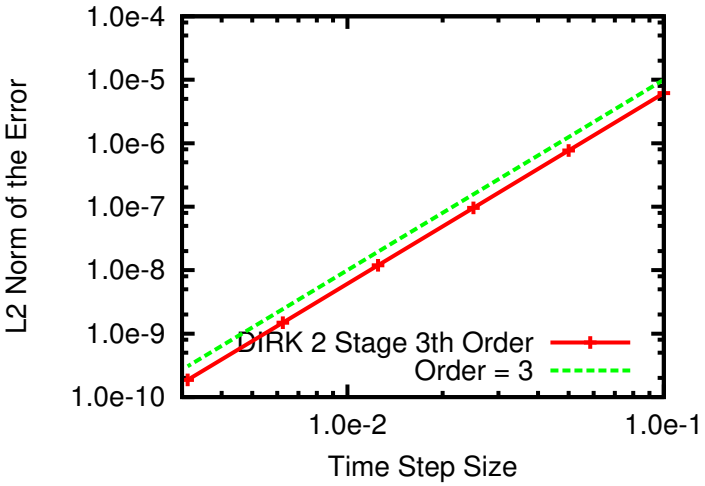


Figure 25: a) Butcher Tableau and b) Order of accuracy for the SinCos Problem (Section 6.2) for DIRK 2 Stage 3 Order.

Part II
User's Manual

5 ParameterList Description

The following description of the Rythmos ParameterList was auto-generated from the ParameterList itself, thus allowing easy refreshing of the input description and providing up-to-date information. Fig. 26 is a hyperlinked, indented heirarchy of the Rythmos ParameterList, that provides an overall structure of the Rythmos ParameterList. The hyerlinks are connected to subsequent sections, which contain additional hyperlinks to parent and children sublists, and descriptions of the sublist and parameters. All these hyperlinks provide an easy way to hop around the parameters in order to setup the time-integrator.

The basic structure starts with the "Integrator Base", and is broken down into the Integrator and the Stepper. Each of these contain several sublists giving the details of the Integrator and Stepper, e.g., IntegrationControlStrategy, InterpolationBuffer, StepControlStrategy, and error weight vector.

An easy way to create a Rythmos ParameterList is to get a valid ParameterList from the IntegatorBuilder, and manipulate the parameters from there. In this way, one able to correctly setup the Integrator and Stepper for a transient soution. For example,

```
RCP<IntegratorBuilder<double> > ib = integratorBuilder<double>();
RCP<ParameterList> pl = Teuchos::parameterList();
pl->setParameters(*(ib->getValidParameters()));
pl->sublist("Integrator Settings").set("Final Time", 1.0e-0);
```

Obviously, most software will get its input specification from the XML version of the ParameterLists, and then pass them onto Rythmos. The description provided here can also help in setting up those XML files.

- Integrator Base (Section 5.1)
  - Integrator Settings (Section 5.2)
    - Integrator Selection (Section 5.3)
      - Default Integrator (Section 5.4)
        - VerboseObject (Section 5.5)
    - Integration Control Strategy Selection (Section 5.6)
      - Simple Integration Control Strategy (Section 5.7)
      - Ramping Integration Control Strategy (Section 5.8)
    - Stepper Settings (Section 5.9)
      - Stepper Selection (Section 5.10)
        - Forward Euler (Section 5.11)
        - Backward Euler (Section 5.12)
        - Implicit BDF (Section 5.13)
        - Explicit RK (Section 5.14)
        - Implicit RK (Section 5.15)
      - Step Control Settings (Section 5.16)
        - Step Control Strategy Selection (Section 5.17)
          - Fixed Step Control Strategy (Section 5.18)
          - Simple Step Control Strategy (Section 5.19)
          - First Order Error Step Control Strategy (Section 5.20)
          - Implicit BDF Stepper Step Control Strategy (Section 5.21)
            - magicNumbers (Section 5.22)
          - Implicit BDF Stepper Ramping Step Control Strategy (Section 5.23)
        - Error Weight Vector Calculator Selection (Section 5.24)
          - Implicit BDF Stepper Error Weight Vector Calculator (Section 5.25)
      - Interpolator Selection (Section 5.26)
        - Linear Interpolator (Section 5.27)
        - Hermite Interpolator (Section 5.28)
        - Cubic Spline Interpolator (Section 5.29)
      - Runge Kutta Butcher Tableau Selection (Section 5.30)
        - Forward Euler (Section 5.31)
        - Explicit 2 Stage 2nd order by Runge (Section 5.32)
        - Explicit Trapezoidal (Section 5.33)
        - Explicit 3 Stage 3rd order (Section 5.34)
        - Explicit 3 Stage 3rd order by Heun (Section 5.35)
        - ...
    - Interpolation Buffer Settings (Section 5.69)
      - Trailing Interpolation Buffer Selection (Section 5.70)
        - Interpolation Buffer (Section 5.71)
      - Interpolation Buffer Appender Selection (Section 5.72)
        - Pointwise Interpolation Buffer Appender (Section 5.73)
      - Interpolator Selection (Section 5.74)
        - Linear Interpolator (Section 5.75)
        - Hermite Interpolator (Section 5.76)
        - Cubic Spline Interpolator (Section 5.77)

Figure 26: Schematic of ParameterList heirarchy.

5.1 Integrator Base

Description: The root ParameterList for the Integrator Builder.

Parent(s): ROOT

Child(ren): Integrator Settings (Section 5.2)
Integration Control Strategy Selection (Section 5.6)
Stepper Settings (Section 5.9)
Interpolation Buffer Settings (Section 5.69)

Parameters: None.

5.2 Integrator Settings

Description: These parameters are used directly in setting up the Integrator.

Parent(s): Integrator Base (Section 5.1)

Child(ren): Integrator Selection (Section 5.3)

Parameters: **Initial Time = 0** The initial time to start integration.

Final Time = 1 The final time to end integration.

Land On Final Time = 1 Exactly land on the final time; do not step past final time and interpolate.

5.3 Integrator Selection

Description: Select the Integrator to be used.

Parent(s): Integrator Settings (Section 5.2)

Child(ren): Default Integrator (Section 5.4)

Parameters: **Integrator Type = Default Integrator** Determines the type of Rythmos::Integrator object that will be built. The parameters for each Integrator Type are specified in this sublist

Valid std::string values:
"None"
"Default Integrator"

5.4 Default Integrator

Description: This Integrator will accept an IntergationControlStrategy, and can have an IntegrationObserver. The client can specify the maximum number of time steps allowed. The Integrator will loop over the Stepper until it reaches the requested time. For each step, the step size will be determined through a couple mechanisms/filters. If an Integration Control Strategy has been specified, the step size and the step type (fixed or variable) will be determined by it. Otherwise the step size will be set to the maximum real value and the step type will be variable. Next if the step size is beyond the final time and the 'Land On Final Time' is specified, the step size is adjusted to advance the state to the final time. The Stepper is passed this step size and type to advance the state. The DefaultIntegrator determines the step size and type taken by the Stepper, and if the step has failed. If the IntegrationControlStrategy handles failures, it can suggest another step size and retry with the Stepper. Otherwise, the Integrator will fall through with a failure. With a successful step of the Stepper, the Integrator repeats the above until it reaches the requested time. Multiple requested times can be passed to the Integrator.

Parent(s): Integrator Selection (Section 5.3)

Child(ren): VerboseObject (Section 5.5)

Parameters: **Max Number Time Steps = 2147483647** Set the maximum number of integration time-steps allowed.

5.5 VerboseObject

Description:

Parent(s): Default Integrator (Section 5.4)
Simple Integration Control Strategy (Section 5.7)
Ramping Integration Control Strategy (Section 5.8)
Forward Euler (Section 5.11)
Backward Euler (Section 5.12)
Implicit BDF (Section 5.13)
Explicit RK (Section 5.14)
Implicit RK (Section 5.15)
Fixed Step Control Strategy (Section 5.18)
Simple Step Control Strategy (Section 5.19)
First Order Error Step Control Strategy (Section 5.20)
magicNumbers (Section 5.22)
Implicit BDF Stepper Error Weight Vector Calculator (Section 5.25)
Linear Interpolator (Section 5.27)
Hermite Interpolator (Section 5.28)
Cubic Spline Interpolator (Section 5.29)
Forward Euler (Section 5.31)
Explicit 2 Stage 2nd order by Runge (Section 5.32)
Explicit Trapezoidal (Section 5.33)
Explicit 3 Stage 3rd order (Section 5.34)
Explicit 3 Stage 3rd order by Heun (Section 5.35)
Explicit 3 Stage 3rd order TVD (Section 5.36)
Explicit 4 Stage 3rd order by Runge (Section 5.37)
Explicit 4 Stage (Section 5.38)
Explicit 3/8 Rule (Section 5.39)
Backward Euler (Section 5.40)
IRK 1 Stage Theta Method (Section 5.41)
IRK 2 Stage Theta Method (Section 5.42)
Singly Diagonal IRK 2 Stage 2nd order (Section 5.43)
Singly Diagonal IRK 2 Stage 3rd order (Section 5.44)
Singly Diagonal IRK 3 Stage 4th order (Section 5.45)
Singly Diagonal IRK 5 Stage 4th order (Section 5.46)
Singly Diagonal IRK 5 Stage 5th order (Section 5.47)
Diagonal IRK 2 Stage 3rd order (Section 5.48)
Implicit 1 Stage 2nd order Gauss (Section 5.49)
Implicit 2 Stage 4th order Gauss (Section 5.50)
Implicit 3 Stage 6th order Gauss (Section 5.51)
Implicit 2 Stage 4th Order Hammer & Hollingsworth (Section 5.52)
Implicit 3 Stage 6th Order Kuntzmann & Butcher (Section 5.53)
Implicit 1 Stage 1st order Radau left (Section 5.54)
Implicit 2 Stage 3rd order Radau left (Section 5.55)
Implicit 3 Stage 5th order Radau left (Section 5.56)
Implicit 1 Stage 1st order Radau right (Section 5.57)
Implicit 2 Stage 3rd order Radau right (Section 5.58)

Implicit 3 Stage 5th order Radau right (Section 5.59)
Implicit 2 Stage 2nd order Lobatto A (Section 5.60)
Implicit 3 Stage 4th order Lobatto A (Section 5.61)
Implicit 4 Stage 6th order Lobatto A (Section 5.62)
Implicit 2 Stage 2nd order Lobatto B (Section 5.63)
Implicit 3 Stage 4th order Lobatto B (Section 5.64)
Implicit 4 Stage 6th order Lobatto B (Section 5.65)
Implicit 2 Stage 2nd order Lobatto C (Section 5.66)
Implicit 3 Stage 4th order Lobatto C (Section 5.67)
Implicit 4 Stage 6th order Lobatto C (Section 5.68)
Interpolation Buffer (Section 5.71)
Pointwise Interpolation Buffer Appender (Section 5.73)
Linear Interpolator (Section 5.75)
Hermite Interpolator (Section 5.76)
Cubic Spline Interpolator (Section 5.77)

Child(ren): None.

Parameters: **Verbosity Level = default** The verbosity level to use to override whatever is set in code. The value of "default" will allow the level set in code to be used.

Valid std::string values:

"default"	Use level set in code
"none"	Produce no output
"low"	Produce minimal output
"medium"	Produce a little more output
"high"	Produce a higher level of output
"extreme"	Produce the highest level of output

Output File = none The file to send output to. If the value "none" is used, then whatever is set in code will be used. However, any other std::string value will be used to create an std::ofstream object to a file with the given name. Therefore, any valid file name is a valid std::string value for this parameter.

5.6 Integration Control Strategy Selection

Description: Note that some settings conflict between step control and integration control. In general, the integration control decides which steps will be fixed or variable, not the stepper. When the integration control decides to take variable steps, the step control is then responsible for choosing appropriate step-sizes.

Parent(s): Integrator Base (Section 5.1)

Child(ren): Simple Integration Control Strategy (Section 5.7)
Ramping Integration Control Strategy (Section 5.8)

Parameters: **Integration Control Strategy Type = None** Determines the type of Rythmos::IntegrationControlStrategy object that will be built. The parameters for each Integration Control Strategy Type are specified in this sublist

Valid std::string values:

"None"
"Simple Integration Control Strategy"
"Ramping Integration Control Strategy"

5.7 Simple Integration Control Strategy

Description: This Integration Control Strategy is meant to be simple with very few parameters controlling it. Basically the client can select fixed step type (the Stepper can only take the requested step size) or variable step type (the Stepper can adjust the step size to meet accuracy, order, or other criteria). For fixed step type, the client can specify the step size and number of steps. For variable step type, the client can set the maximum step size allowable.

Parent(s): Integration Control Strategy Selection (Section 5.6)

Child(ren): None.

Parameters:

Take Variable Steps = 1 Take variable time steps or fixed time steps. If set to false, then the parameter "Fixed dt" or "Number of Time Steps" must be set!

Max dt = 1.79769e+308 Gives the max size of the variable time steps. This is only read and used if "Take Variable Steps" is set to true.

Number of Time Steps = -1 Gives the number of fixed time steps. The actual step size gets computed on the fly given the size of the time domain. This is only read and used if "Take Variable Steps" is set to false and "Fixed dt" is set to < 0.0.

Fixed dt = -1 Gives the size of the fixed time steps. This is only read and used if "Take Variable Steps" is set to false.

5.8 Ramping Integration Control Strategy

Description: This Integration Control Strategy is very similar to 'Simple Integration Control Strategy' except for handling an initial constant-sized steps followed by a ramped-fixed-sized steps, and finally variable- or fixed-sized steps. The client needs to additionally set the initial step size and the maximum number of step failures allowed.

Parent(s): Integration Control Strategy Selection (Section 5.6)

Child(ren): None.

Parameters:

Take Variable Steps = 1 Take variable time steps after 'Number of Initial Constant Steps' plus 'Number of Ramping Steps' steps. Variable time stepping allows the Stepper to adjust the time step through a StepControlStrategy after fixed time steps during initial constant steps and ramping steps. If false, fixed-time steps are taken after ramping. Fixed time stepping requires the Stepper to take the time step set by this IntegrationControlStrategy.

Number of Initial Constant Steps = 0 Number of initial constant steps to take before starting the ramping.

Number of Ramping Steps = 6 Number of ramping steps to take before handing control to variable stepper if 'Take Variable Steps' is set to true. Otherwise take fixed-time steps.

Initial dt = -1 Initial time step.

Min dt = 2.22507e-308 Minimum time step.

Max dt = 1.79769e+308 Maximum time step.

Ramping Factor = 1 Time step growth factor used during ramping phase. dt_n+1 = (ramping factor) * dt_n

Maximum Number of Step Failures = 100 The maximum number of step failures before exiting with error.

5.9 Stepper Settings

Description: This parameter list sets various parameters for the Stepper.

Parent(s): Integrator Base (Section 5.1)

Child(ren): Stepper Selection (Section 5.10)
Step Control Settings (Section 5.16)
Interpolator Selection (Section 5.26)
Runge Kutta Butcher Tableau Selection (Section 5.30)

Parameters: None.

5.10 Stepper Selection

Description: Selects the Stepper for the time integration. It should be that some time integrators can be accessed through different Steppers, e.g., Backward Euler can be obtained through the 'Backward Euler', a first-order 'Implicit BDF', or a one-stage 'Implicit RK' Stepper.Special note for 'Implicit RK' Stepper: If a fully implicit RK Butcher tableau is chosen, then the stepper will not be fully initialized unless a W factory object is set on the IntegratorBuilder through setWFactoryObject.

Parent(s): Stepper Settings (Section 5.9)

Child(ren): Forward Euler (Section 5.11)
Backward Euler (Section 5.12)
Implicit BDF (Section 5.13)
Explicit RK (Section 5.14)
Implicit RK (Section 5.15)

Parameters: **Stepper Type = Backward Euler** Determines the type of Rythmos::Stepper object that will be built. The parameters for each Stepper Type are specified in this sublist

Valid std::string values:
"None"
"Forward Euler"
"Backward Euler"
"Implicit BDF"
"Explicit RK"
"Implicit RK"

5.11 Forward Euler

Description: This is the basic Forward Euler method: x_n = x_n-1 + dt*x_dot_n-1

Parent(s): Stepper Selection (Section 5.10)

Child(ren): None.

Parameters: None.

5.12 Backward Euler

Description: This is the basic Backward Euler method: x_n = x_n-1 + dt*x_dot_n

Parent(s): Stepper Selection (Section 5.10)

Child(ren): None.

Parameters: None.

5.13 Implicit BDF

Description: This Stepper provides a re-implementation of the algorithms in the LLNL Sundials code IDA. This is an implicit BDF integrator for DAEs which uses variable step-sizes and variable-orders first through fourth.

Parent(s): Stepper Selection (Section 5.10)

Child(ren): None.

Parameters: None.

5.14 Explicit RK

Description: This Stepper has many explicit time-integrators using Runge-Kutta formulation and the Butcher Tableau specification. See 'Runge Kutta Butcher Tableau Selection' ParameterList for available options.

Parent(s): Stepper Selection (Section 5.10)

Child(ren): None.

Parameters: None.

5.15 Implicit RK

Description: This Stepper has many implicit time-integrators using Runge-Kutta formulation and the Butcher Tableau specification. See 'Runge Kutta Butcher Tableau Selection' ParameterList for available options.

Parent(s): Stepper Selection (Section 5.10)

Child(ren): None.

Parameters: None.

5.16 Step Control Settings

Description: Not all step control strategies are compatible with each stepper. If the strategy has the name of a stepper in its name, then it only works with that stepper.

Parent(s): Stepper Settings (Section 5.9)

Child(ren): Step Control Strategy Selection (Section 5.17)
Error Weight Vector Calculator Selection (Section 5.24)

Parameters: None.

5.17 Step Control Strategy Selection

Description: Used to select the Control Strategy for the stepper.

Parent(s): Step Control Settings (Section 5.16)

Child(ren): Fixed Step Control Strategy (Section 5.18)
Simple Step Control Strategy (Section 5.19)
First Order Error Step Control Strategy (Section 5.20)
Implicit BDF Stepper Step Control Strategy (Section 5.21)
Implicit BDF Stepper Ramping Step Control Strategy (Section 5.23)

Parameters: **Step Control Strategy Type = None** Determines the type of Rythmos::StepControlStrategy object that will be built. The parameters for each Step Control Strategy Type are specified in this sublist

Valid std::string values:
"None"
"Fixed Step Control Strategy"
"Simple Step Control Strategy"
"First Order Error Step Control Strategy"
"Implicit BDF Stepper Step Control Strategy"
"Implicit BDF Stepper Ramping Step Control Strategy"

5.18 Fixed Step Control Strategy

Description: This Step Control Strategy can be used for Steppers setup for variable step type (a stepper that can adjust its step size based on accuracy, order or other criteria), but would like to make fixed step sizes or used fixed step size as its default.

Parent(s): Step Control Strategy Selection (Section 5.17)

Child(ren): None.

Parameters: None.

5.19 Simple Step Control Strategy

Description: This Step Control Strategy starts with the initial step size, and simply increases or decreases the step size by the appropriate factor which is based on the change in the solution relative to the specified relative and absolute tolerances ($|dx| < r^*|x| + a$) and if solution status from the solver passes. Additionally the step size is bounded by the miminum and maximum step size, and the stepper will fail if the step size fails more than the specified value.

Parent(s): Step Control Strategy Selection (Section 5.17)

Child(ren): None.

Parameters: **Initial Step Size = 2.22507e-308** Initial step size.

Min Step Size = 2.22507e-308 Minimum step size.

Max Step Size = 1.79769e+308 Maximum step size.

Step Size Increase Factor = 1.5 Factor used to increase the step size after a successful step.

Step Size Decrease Factor = 0.5 Factor used to decrease the step size after a step failure.

Maximum Number of Step Failures = 100 The maximum number of step failures before exiting with an error. The number of failure steps are carried between successful steps.

Solution Change Relative Tolerance = 1e-06 The allowable relative change in the solution for each step to pass. The stepper solution status is also used to determine pass/fail.

Solution Change Absolute Tolerance = 1e-12 The allowable absolute change in the solution for each step to pass. The stepper solution status is also used to determine pass/fail.

5.20 First Order Error Step Control Strategy

Description: This Step Control Strategy produces a step size based on a first-order predictor (Forward Euler) and a first-order solution (Backward Euler) by by using a weight norm of the difference between the predicted and solution. See Gresho and Sani, 'Incompressible Flow and the Finite Element Method', Vol. 1, 1998, p. 268.

Parent(s): Step Control Strategy Selection (Section 5.17)

Child(ren): None.

Parameters:

Initial Step Size = 1 Initial step size.

Min Step Size = 2.22507e-308 Minimum step size.

Max Step Size = 1.79769e+308 Maximum step size.

Max Step Size Increase Factor = 1.5 The maximum factor to increase the step size after a successful step.

Min Step Size Decrease Factor = 0.5 The minimum allowable factor to decrease the step size. If the stepSizeFactor_ is below this, the current step is considered a failure and retried with 'Min Step Size Decrease Factor' applied.

Maximum Number of Step Failures = 100 The maximum number of step failures before exiting with an error. The number of failure steps are carried between successful steps.

Error Relative Tolerance = 1e-06 The allowable relative change in the error (the difference between the solution at the end of the step and the predicted solution) for each step to pass. The stepper solution status is also used to determine pass/fail.

Error Absolute Tolerance = 1e-12 The allowable absolute change in the error (the difference between the solution at the end of the step and the predicted solution) for each step to pass. The stepper solution status is also used to determine pass/fail.

5.21 Implicit BDF Stepper Step Control Strategy

Description: This Step Control Strategy is specifically for use with the 'Implicit BDF' Stepper. The parameters in this list and sublist are directly related to those available in SUNDAILS/IDA. See Hindmarsh, 'The PVODE and IDA Algorithms', 2000 for more details.

Parent(s): Step Control Strategy Selection (Section 5.17)

Child(ren): magicNumbers (Section 5.22)

Parameters:

minOrder = 1 lower limit of order selection, guaranteed

maxOrder = 5 upper limit of order selection, does not guarantee this order

relErrTol = 0.0001 Relative tolerance value used in WRMS calculation.

absErrTol = 1e-06 Absolute tolerance value used in WRMS calculation.

constantStepSize = 0 Use constant (=0) or variable (=1) step sizes during BDF steps.

stopTime = 10 The final time for time integration. This may be in conflict with the Integrator's final time.

failStepIfNonlinearSolveFails = 0 Power user command. Will force the function acceptStep() to return false even if the LET is acceptable. Used to run with loose tolerances but enforce a correct nonlinear solution to the step.

5.22 magicNumbers

Description: These are knobs in the algorithm that have been set to reasonable values using lots of testing and heuristics and some theory.

Parent(s): Implicit BDF Stepper Step Control Strategy (Section 5.21)

Child(ren): None.

Parameters:

h0_safety = 2 Safety factor on the initial step size for time-dependent DAEs. Larger values will tend to reduce the initial step size.

h0_max_factor = 0.001 Factor multipler for the initial step size for non-time-dependent DAEs.

h_phase0_incr = 2 Initial ramp-up in variable mode (stepSize multiplier).

h_max_inv = 0 The inverse of the maximum time step (maxTimeStep). (Not used?)

Tkm1_Tk_safety = 2 Used to help control the decrement of the order when the order is less than or equal to 2. Larger values will tend to reduce the order from one step to the next.

Tkp1_Tk_safety = 0.5 Used to control the increment of the order when the order is one. Larger values tend to increase the order for the next step.

r_factor = 0.9 Used in rejectStep: time step ratio multiplier.

r_safety = 2 Local error multiplier as part of time step ratio calculation.

r_fudge = 0.0001 Local error addition as part of time step ratio calculation.

r_min = 0.125 Used in rejectStep: How much to cut step and lower bound for time step ratio.

r_max = 0.9 Upper bound for time step ratio.

r_hincr_test = 2 Used in completeStep: If time step ratio > this then set time step ratio to r_hincr.

r_hincr = 2 Used in completeStep: Limit on time step ratio increases, not checked by r_max.

max_LET_fail = 15 Max number of rejected steps

minTimeStep = 0 Bound on smallest time step in variable mode.

maxTimeStep = 10 bound on largest time step in variable mode.

5.23 Implicit BDF Stepper Ramping Step Control Strategy

Description: This Step Control Strategy is specifically for use with the 'Implicit BDF' Stepper, and has a two-phase approach: constant step sizes and followed by variable step sizes. The step size is adjusted based on the WRMS, see Implicit BDF Stepper Error Weight Vector Calculator

Parent(s): Step Control Strategy Selection (Section 5.17)

Child(ren): None.

Parameters:

Number of Constant First Order Steps = 10 Number of constant steps to take before handing control to variable stepper.

Initial Step Size = 0.001 Initial time step size and target step size to take during the initial constant step phase (could be reduced due to step failures).

Min Step Size = 1e-07 Minimum time step size.

Max Step Size = 1 Maximum time step size.

Step Size Increase Factor = 1.2 Time step growth factor used after a successful time step. dt_n+1 = (increase factor) * dt_n

Step Size Decrease Factor = 0.5 Time step reduction factor used for a failed time step. dt_n+1 = (decrease factor) * dt_n

Min Order = 1 Minimum order to run at.

Max Order = 5 Maximum order to run at.

Absolute Error Tolerance = 1e-05 abstol value used in WRMS calculation.

Relative Error Tolerance = 0.001 reltol value used in WRMS calculation.

Use LET To Determine Step Acceptance = FALSE If set to TRUE, then acceptance of step dependes on LET in addition to Nonlinear solver converging.

Valid std::string values:
"TRUE"
"FALSE"

5.24 Error Weight Vector Calculator Selection

Description: Not all ErrWtVec calculators are compatible with each step control strategy. If the calculator has the name of a stepper or another step control strategy in its name, then it only works with that step control strategy.

Parent(s): Step Control Settings (Section 5.16)

Child(ren): Implicit BDF Stepper Error Weight Vector Calculator (Section 5.25)

Parameters: **Error Weight Vector Calculator Type = None** Determines the type of Rythmos::ErrWtVecCalc object that will be built. The parameters for each Error Weight Vector Calculator Type are specified in this sublist

Valid std::string values:
"None"
"Implicit BDF Stepper Error Weight Vector Calculator"

5.25 Implicit BDF Stepper Error Weight Vector Calculator

Description: This Error Weight Vector Calculator is specifically for use with the 'Implicit BDF' Stepper.

Parent(s): Error Weight Vector Calculator Selection (Section 5.24)

Child(ren): None.

Parameters: None.

5.26 Interpolator Selection

Description: Note all Steppers accept an interpolator. Currently, only the BackwardEuler stepper does.

Parent(s): Stepper Settings (Section 5.9)

Child(ren): Linear Interpolator (Section 5.27)
Hermite Interpolator (Section 5.28)
Cubic Spline Interpolator (Section 5.29)

Parameters: **Interpolator Type = None** Determines the type of Rythmos::Interpolator object that will be built. The parameters for each Interpolator Type are specified in this sublist

Valid std::string values:
"None"
"Linear Interpolator"
"Hermite Interpolator"
"Cubic Spline Interpolator"

5.27 Linear Interpolator

Description: This provides a simple linear interpolation between time nodes.

Parent(s): Interpolator Selection (Section 5.26)

Child(ren): None.

Parameters: None.

5.28 Hermite Interpolator

Description: This provides a piecewise cubic Hermite interpolation on each interval where the data is the solution and its time derivatives at the end points of the interval. It will match 3rd degree polynomials exactly with both function values and derivatives.

Parent(s): Interpolator Selection (Section 5.26)

Child(ren): None.

Parameters: None.

5.29 Cubic Spline Interpolator

Description: This provides a cubic spline interpolation between time nodes.

Parent(s): Interpolator Selection (Section 5.26)

Child(ren): None.

Parameters: None.

5.30 Runge Kutta Butcher Tableau Selection

Description: Only the Explicit RK Stepper and the Implicit RK Stepper accept an RK Butcher Tableau.

Parent(s): Stepper Settings (Section 5.9)

Child(ren): Forward Euler (Section 5.31)
Explicit 2 Stage 2nd order by Runge (Section 5.32)
Explicit Trapezoidal (Section 5.33)
Explicit 3 Stage 3rd order (Section 5.34)
Explicit 3 Stage 3rd order by Heun (Section 5.35)
Explicit 3 Stage 3rd order TVD (Section 5.36)
Explicit 4 Stage 3rd order by Runge (Section 5.37)
Explicit 4 Stage (Section 5.38)
Explicit 3/8 Rule (Section 5.39)
Backward Euler (Section 5.40)

IRK 1 Stage Theta Method (Section 5.41)
IRK 2 Stage Theta Method (Section 5.42)
Singly Diagonal IRK 2 Stage 2nd order (Section 5.43)
Singly Diagonal IRK 2 Stage 3rd order (Section 5.44)
Singly Diagonal IRK 3 Stage 4th order (Section 5.45)
Singly Diagonal IRK 5 Stage 4th order (Section 5.46)
Singly Diagonal IRK 5 Stage 5th order (Section 5.47)
Diagonal IRK 2 Stage 3rd order (Section 5.48)
Implicit 1 Stage 2nd order Gauss (Section 5.49)
Implicit 2 Stage 4th order Gauss (Section 5.50)
Implicit 3 Stage 6th order Gauss (Section 5.51)
Implicit 2 Stage 4th Order Hammer & Hollingsworth (Section 5.52)
Implicit 3 Stage 6th Order Kuntzmann & Butcher (Section 5.53)
Implicit 1 Stage 1st order Radau left (Section 5.54)
Implicit 2 Stage 3rd order Radau left (Section 5.55)
Implicit 3 Stage 5th order Radau left (Section 5.56)
Implicit 1 Stage 1st order Radau right (Section 5.57)
Implicit 2 Stage 3rd order Radau right (Section 5.58)
Implicit 3 Stage 5th order Radau right (Section 5.59)
Implicit 2 Stage 2nd order Lobatto A (Section 5.60)
Implicit 3 Stage 4th order Lobatto A (Section 5.61)
Implicit 4 Stage 6th order Lobatto A (Section 5.62)
Implicit 2 Stage 2nd order Lobatto B (Section 5.63)
Implicit 3 Stage 4th order Lobatto B (Section 5.64)
Implicit 4 Stage 6th order Lobatto B (Section 5.65)
Implicit 2 Stage 2nd order Lobatto C (Section 5.66)
Implicit 3 Stage 4th order Lobatto C (Section 5.67)
Implicit 4 Stage 6th order Lobatto C (Section 5.68)

Parameters: **Runge Kutta Butcher Tableau Type = None** Determines the type of Rythmos::RKButcherTableau object that will be built. The parameters for each Runge Kutta Butcher Tableau Type are specified in this sublist

Valid std::string values:

"None"
"Forward Euler"
"Explicit 2 Stage 2nd order by Runge"
"Explicit Trapezoidal"
"Explicit 3 Stage 3rd order"
"Explicit 3 Stage 3rd order by Heun"
"Explicit 3 Stage 3rd order TVD"
"Explicit 4 Stage 3rd order by Runge"
"Explicit 4 Stage"
"Explicit 3/8 Rule"
"Backward Euler"
"IRK 1 Stage Theta Method"
"IRK 2 Stage Theta Method"
"Singly Diagonal IRK 2 Stage 2nd order"
"Singly Diagonal IRK 2 Stage 3rd order"
"Singly Diagonal IRK 3 Stage 4th order"
"Singly Diagonal IRK 5 Stage 4th order"
"Singly Diagonal IRK 5 Stage 5th order"
"Diagonal IRK 2 Stage 3rd order"
"Implicit 1 Stage 2nd order Gauss"
"Implicit 2 Stage 4th order Gauss"
"Implicit 3 Stage 6th order Gauss"
"Implicit 2 Stage 4th Order Hammer & Hollingsworth"
"Implicit 3 Stage 6th Order Kuntzmann & Butcher"
"Implicit 1 Stage 1st order Radau left"
"Implicit 2 Stage 3rd order Radau left"
"Implicit 3 Stage 5th order Radau left"
"Implicit 1 Stage 1st order Radau right"
"Implicit 2 Stage 3rd order Radau right"
"Implicit 3 Stage 5th order Radau right"
"Implicit 2 Stage 2nd order Lobatto A"
"Implicit 3 Stage 4th order Lobatto A"
"Implicit 4 Stage 6th order Lobatto A"
"Implicit 2 Stage 2nd order Lobatto B"
"Implicit 3 Stage 4th order Lobatto B"
"Implicit 4 Stage 6th order Lobatto B"
"Implicit 2 Stage 2nd order Lobatto C"
"Implicit 3 Stage 4th order Lobatto C"
"Implicit 4 Stage 6th order Lobatto C"

5.31 Forward Euler

Description: `Forward Euler`

```
c = [ 0 ]'
A = [ 0 ]
b = [ 1 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.32 Explicit 2 Stage 2nd order by Runge

Description: `Explicit 2 Stage 2nd order by Runge`

```
Also known as Explicit Midpoint
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S.P. Norsett, G. Wanner
Table 1.1, pg 135
c = [  0  1/2 ]'
A = [  0      ]
    [ 1/2  0  ]
b = [  0   1  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.33 Explicit Trapezoidal

Description:
```
Explicit Trapezoidal
c = [  0   1  ]'
A = [  0      ]
    [  1   0  ]
b = [ 1/2 1/2 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.34 Explicit 3 Stage 3rd order

Description:
```
Explicit 3 Stage 3rd order
c = [  0  1/2  1  ]'
A = [  0          ]
    [ 1/2  0      ]
    [ -1   2   0  ]
b = [ 1/6 4/6 1/6 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.35 Explicit 3 Stage 3rd order by Heun

Description:
```
Explicit 3 Stage 3rd order by Heun
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S.P. Norsett, G. Wanner
Table 1.1, pg 135
c = [  0  1/3 2/3 ]'
A = [  0          ]
    [ 1/3  0      ]
    [  0  2/3  0  ]
b = [ 1/4  0  3/4 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.36 Explicit 3 Stage 3rd order TVD

Description:

```
Explicit 3 Stage 3rd order TVD
Sigal Gottlieb and Chi-Wang Shu
'Total Variation Diminishing Runge-Kutta Schemes'
Mathematics of Computation
Volume 67, Number 221, January 1998, pp. 73-85
c = [  0   1  1/2 ]'
A = [  0          ]
    [  1   0      ]
    [ 1/4 1/4  0  ]
b = [ 1/6 1/6 4/6 ]'
This is also written in the following set of updates.
u1 = u^n + dt L(u^n)
u2 = 3 u^n/4 + u1/4 + dt L(u1)/4
u^(n+1) = u^n/3 + 2 u2/2 + 2 dt L(u2)/3
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.37 Explicit 4 Stage 3rd order by Runge

Description:

```
Explicit 4 Stage 3rd order by Runge
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S.P. Norsett, G. Wanner
Table 1.1, pg 135
c = [  0  1/2  1   1  ]'
A = [  0              ]
    [ 1/2  0          ]
    [  0   1   0      ]
    [  0   0   1   0  ]
b = [ 1/6 2/3  0  1/6 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.38 Explicit 4 Stage

Description:

```
Explicit 4 Stage
"The" Runge-Kutta Method (explicit):
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S.P. Norsett, G. Wanner
Table 1.2, pg 138
```

```
c = [  0  1/2 1/2  1  ]'
A = [  0              ]
    [ 1/2  0          ]
    [  0  1/2  0      ]
    [  0   0   1   0  ]
b = [ 1/6 1/3 1/3 1/6 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.39 Explicit 3/8 Rule

Description:

```
Explicit 3/8 Rule
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S.P. Norsett, G. Wanner
Table 1.2, pg 138
c = [  0  1/3 2/3  1  ]'
A = [  0              ]
    [ 1/3  0          ]
    [-1/3  1   0      ]
    [  1  -1   1   0  ]
b = [ 1/8 3/8 3/8 1/8 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.40 Backward Euler

Description:

```
Backward Euler
c = [ 1 ]'
A = [ 1 ]
b = [ 1 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.41 IRK 1 Stage Theta Method

Description:

```
IRK 1 Stage Theta Method
Non-standard finite-difference methods
in dynamical systems, P. Kama,
Dissertation, University of Pretoria, pg. 49.
Comment:  Generalized Implicit Midpoint Method
c = [ theta ]'
A = [ theta ]
b = [  1  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: **theta = 0.5** Valid values are 0 <= theta <= 1, where theta = 0 implies Forward Euler, theta = 1/2 implies midpoint method, and theta = 1 implies Backward Euler. For theta != 1/2, this method is first-order accurate, and with theta = 1/2, it is second-order accurate. This method is A-stable, but becomes L-stable with theta=1.

5.42 IRK 2 Stage Theta Method

Description:
```
IRK 2 Stage Theta Method
Computer Methods for ODEs and DAEs
U. M. Ascher and L. R. Petzold
p. 113
c = [  0       1     ]'
A = [  0       0     ]
    [ 1-theta  theta ]
b = [ 1-theta  theta ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: **theta = 0.5** Valid values are 0 < theta <= 1, where theta = 0 implies Forward Euler, theta = 1/2 implies trapezoidal method, and theta = 1 implies Backward Euler. For theta != 1/2, this method is first-order accurate, and with theta = 1/2, it is second-order accurate. This method is A-stable, but becomes L-stable with theta=1.

5.43 Singly Diagonal IRK 2 Stage 2nd order

Description:
```
Singly Diagonal IRK 2 Stage 2nd order
Computer Methods for ODEs and DAEs
U. M. Ascher and L. R. Petzold
p. 106
gamma = (2+-sqrt(2))/2
c = [  gamma   1     ]'
A = [  gamma   0     ]
    [ 1-gamma  gamma ]
b = [ 1-gamma  gamma ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: **gamma = 0.292893** The default value is gamma = (2-sqrt(2))/2. This will produce an L-stable 2nd order method with the stage times within the timestep. Other values of gamma will still produce an L-stable scheme, but will only be 1st order accurate.

5.44 Singly Diagonal IRK 2 Stage 3rd order

Description:
```
Singly Diagonal IRK 2 Stage 3rd order
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S. P. Norsett, and G. Wanner
Table 7.2, pg 207
```

```
gamma = (3+-sqrt(3))/6 -> 3rd order and A-stable
gamma = (2+-sqrt(2))/2 -> 2nd order and L-stable
c = [  gamma     1-gamma  ]'
A = [  gamma     0        ]
    [ 1-2*gamma  gamma    ]
b = [ 1/2        1/2      ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: **3rd Order A-stable = 1** If true, set gamma to gamma = (3+sqrt(3))/6 to obtain a 3rd order A-stable scheme. '3rd Order A-stable' and '2nd Order L-stable' can not both be true.

2nd Order L-stable = 0 If true, set gamma to gamma = (2+sqrt(2))/2 to obtain a 2nd order L-stable scheme. '3rd Order A-stable' and '2nd Order L-stable' can not both be true.

gamma = 0.788675 If both '3rd Order A-stable' and '2nd Order L-stable' are false, gamma will be used. The default value is the '3rd Order A-stable' gamma value, (3+sqrt(3))/6.

5.45 Singly Diagonal IRK 3 Stage 4th order

Description:

```
Singly Diagonal IRK 3 Stage 4th order
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
pg100
gamma = (1/sqrt(3))*cos(pi/18)+1/2
delta = 1/(6*(2*gamma-1)^2)
c = [ gamma      1/2        1-gamma ]'
A = [ gamma                         ]
    [ 1/2-gamma  gamma              ]
    [ 2*gamma    1-4*gamma  gamma   ]
b = [ delta      1-2*delta  delta   ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.46 Singly Diagonal IRK 5 Stage 4th order

Description:

```
Singly Diagonal IRK 5 Stage 4th order
L-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
pg100
c  = [ 1/4       3/4        11/20   1/2     1   ]'
A  = [ 1/4                                      ]
     [ 1/2       1/4                            ]
```

```
     [ 17/50     -1/25      1/4                ]
     [ 371/1360  -137/2720  15/544  1/4        ]
     [ 25/24     -49/48     125/16  -85/12  1/4 ]
b  = [ 25/24     -49/48     125/16  -85/12  1/4 ]'
b' = [ 59/48     -17/96     225/32  -85/12  0   ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.47 Singly Diagonal IRK 5 Stage 5th order

Description:

```
Singly Diagonal IRK 5 Stage 5th order
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
pg101
c = [ (6-sqrt(6))/10   ]
    [ (6+9*sqrt(6))/35 ]
    [ 1                ]
    [ (4-sqrt(6))/10   ]
    [ (4+sqrt(6))/10   ]
A = [ A1 A2 A3 A4 A5 ]
      A1 = [ (6-sqrt(6))/10                ]
           [ (-6+5*sqrt(6))/14             ]
           [ (888+607*sqrt(6))/2850        ]
           [ (3153-3082*sqrt(6))/14250     ]
           [ (-32583+14638*sqrt(6))/71250 ]
      A2 = [ 0                            ]
           [ (6-sqrt(6))/10               ]
           [ (126-161*sqrt(6))/1425       ]
           [ (3213+1148*sqrt(6))/28500    ]
           [ (-17199+364*sqrt(6))/142500 ]
      A3 = [ 0                          ]
           [ 0                          ]
           [ (6-sqrt(6))/10             ]
           [ (-267+88*sqrt(6))/500      ]
           [ (1329-544*sqrt(6))/2500 ]
      A4 = [ 0                         ]
           [ 0                         ]
           [ 0                         ]
           [ (6-sqrt(6))/10            ]
           [ (-96+131*sqrt(6))/625 ]
      A5 = [ 0                 ]
           [ 0                 ]
           [ 0                 ]
           [ 0                 ]
           [ (6-sqrt(6))/10 ]
b = [               0 ]
    [               0 ]
    [             1/9 ]
```

```
[ (16-sqrt(6))/36 ]
[ (16+sqrt(6))/36 ]
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.48 Diagonal IRK 2 Stage 3rd order

Description:

```
Diagonal IRK 2 Stage 3rd order
Hammer & Hollingsworth method
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S. P. Norsett, and G. Wanner
Table 7.1, pg 205
c = [  0   2/3 ]'
A = [  0    0  ]
    [ 1/3  1/3 ]
b = [ 1/4  3/4 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.49 Implicit 1 Stage 2nd order Gauss

Description:

```
Implicit 1 Stage 2nd order Gauss
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.2, pg 72
Also:  Implicit midpoint rule
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S. P. Norsett, and G. Wanner
Table 7.1, pg 205
c = [ 1/2 ]'
A = [ 1/2 ]
b = [  1  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.50 Implicit 2 Stage 4th order Gauss

Description:

```
Implicit 2 Stage 4th order Gauss
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
```

```
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.2, pg 72
c = [ 1/2-sqrt(3)/6  1/2+sqrt(3)/6 ]'
A = [ 1/4            1/4-sqrt(3)/6 ]
    [ 1/4+sqrt(3)/6  1/4           ]
b = [ 1/2            1/2 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.51 Implicit 3 Stage 6th order Gauss

Description:

```
Implicit 3 Stage 6th order Gauss
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.2, pg 72
c = [ 1/2-sqrt(15)/10   1/2              1/2+sqrt(15)/10  ]'
A = [ 5/36              2/9-sqrt(15)/15  5/36-sqrt(15)/30 ]
    [ 5/36+sqrt(15)/24  2/9              5/36-sqrt(15)/24 ]
    [ 5/36+sqrt(15)/30  2/9+sqrt(15)/15  5/36             ]
b = [ 5/18              4/9              5/18             ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.52 Implicit 2 Stage 4th Order Hammer & Hollingsworth

Description:

```
Implicit 2 Stage 4th Order Hammer & Hollingsworth
Hammer & Hollingsworth method
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S. P. Norsett, and G. Wanner
Table 7.3, pg 207
c = [ 1/2-sqrt(3)/6  1/2+sqrt(3)/6 ]'
A = [ 1/4            1/4-sqrt(3)/6 ]
    [ 1/4+sqrt(3)/6  1/4           ]
b = [ 1/2            1/2           ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.53 Implicit 3 Stage 6th Order Kuntzmann & Butcher

Description:
```
Implicit 3 Stage 6th Order Kuntzmann & Butcher
Kuntzmann & Butcher method
Solving Ordinary Differential Equations I:
Nonstiff Problems, 2nd Revised Edition
E. Hairer, S. P. Norsett, and G. Wanner
Table 7.4, pg 209
c = [ 1/2-sqrt(15)/10   1/2              1/2-sqrt(15)/10  ]'
A = [ 5/36              2/9-sqrt(15)/15  5/36-sqrt(15)/30 ]
    [ 5/36+sqrt(15)/24  2/9              5/36-sqrt(15)/24 ]
    [ 5/36+sqrt(15)/30  2/9+sqrt(15)/15  5/36             ]
b = [ 5/18              4/9              5/18             ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.54 Implicit 1 Stage 1st order Radau left

Description:
```
Implicit 1 Stage 1st order Radau left
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.3, pg 73
c = [ 0 ]'
A = [ 1 ]
b = [ 1 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.55 Implicit 2 Stage 3rd order Radau left

Description:
```
Implicit 2 Stage 3rd order Radau left
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.3, pg 73
c = [  0    2/3 ]'
A = [ 1/4  -1/4 ]
    [ 1/4  5/12 ]
b = [ 1/4  3/4  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.56 Implicit 3 Stage 5th order Radau left

Description:

```
Implicit 3 Stage 5th order Radau left
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.4, pg 73
c = [  0   (6-sqrt(6))/10       (6+sqrt(6))/10      ]'
A = [ 1/9  (-1-sqrt(6))/18      (-1+sqrt(6))/18     ]
    [ 1/9  (88+7*sqrt(6))/360   (88-43*sqrt(6))/360 ]
    [ 1/9  (88+43*sqrt(6))/360  (88-7*sqrt(6))/360  ]
b = [ 1/9  (16+sqrt(6))/36      (16-sqrt(6))/36     ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.57 Implicit 1 Stage 1st order Radau right

Description:

```
Implicit 1 Stage 1st order Radau right
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.5, pg 74
c = [ 1 ]'
A = [ 1 ]
b = [ 1 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.58 Implicit 2 Stage 3rd order Radau right

Description:

```
Implicit 2 Stage 3rd order Radau right
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.5, pg 74
c = [ 1/3     1   ]'
A = [ 5/12  -1/12 ]
    [ 3/4    1/4  ]
b = [ 3/4    1/4  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.59 Implicit 3 Stage 5th order Radau right

Description:

```
Implicit 3 Stage 5th order Radau right
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.6, pg 74
c = [ (4-sqrt(6))/10          (4+sqrt(6))/10          1    ]'
A = [ A1 A2 A3 ]
      A1 = [ (88-7*sqrt(6))/360     ]
           [ (296+169*sqrt(6))/1800 ]
           [ (16-sqrt(6))/36        ]
      A2 = [ (296-169*sqrt(6))/1800 ]
           [ (88+7*sqrt(6))/360     ]
           [ (16+sqrt(6))/36        ]
      A3 = [ (-2+3*sqrt(6))/225 ]
           [ (-2-3*sqrt(6))/225 ]
           [ 1/9                ]
b = [ (16-sqrt(6))/36         (16+sqrt(6))/36         1/9 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.60 Implicit 2 Stage 2nd order Lobatto A

Description:

```
Implicit 2 Stage 2nd order Lobatto A
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.7, pg 75
c = [  0    1   ]'
A = [  0    0   ]
    [ 1/2  1/2  ]
b = [ 1/2  1/2  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.61 Implicit 3 Stage 4th order Lobatto A

Description:

```
Implicit 3 Stage 4th order Lobatto A
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.7, pg 75
```

```
c = [  0    1/2    1  ]'
A = [  0     0     0   ]
    [ 5/24  1/3  -1/24  ]
    [ 1/6   2/3   1/6   ]
b = [ 1/6   2/3   1/6   ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.62 Implicit 4 Stage 6th order Lobatto A

Description:

```
Implicit 4 Stage 6th order Lobatto A
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.8, pg 75
c = [ 0  (5-sqrt(5))/10  (5+sqrt(5))/10  1 ]'
A = [ A1  A2  A3  A4 ]
      A1 = [ 0               ]
           [ (11+sqrt(5)/120 ]
           [ (11-sqrt(5)/120 ]
           [ 1/12            ]
      A2 = [ 0                    ]
           [ (25-sqrt(5))/120     ]
           [ (25+13*sqrt(5))/120  ]
           [ 5/12                 ]
      A3 = [ 0                   ]
           [ (25-13*sqrt(5))/120 ]
           [ (25+sqrt(5))/120    ]
           [ 5/12                ]
      A4 = [ 0                ]
           [ (-1+sqrt(5))/120 ]
           [ (-1-sqrt(5))/120 ]
           [ 1/12             ]
b = [ 1/12  5/12  5/12   1/12 ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.63 Implicit 2 Stage 2nd order Lobatto B

Description:

```
Implicit 2 Stage 2nd order Lobatto B
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.9, pg 76
```

```
c = [  0    1   ]'
A = [ 1/2   0   ]
    [ 1/2   0   ]
b = [ 1/2  1/2  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.64 Implicit 3 Stage 4th order Lobatto B

Description:

```
Implicit 3 Stage 4th order Lobatto B
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.9, pg 76
c = [  0    1/2    1   ]'
A = [ 1/6  -1/6    0   ]
    [ 1/6   1/3    0   ]
    [ 1/6   5/6    0   ]
b = [ 1/6   2/3   1/6  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.65 Implicit 4 Stage 6th order Lobatto B

Description:

```
Implicit 4 Stage 6th order Lobatto B
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.10, pg 76
c = [ 0     (5-sqrt(5))/10       (5+sqrt(5))/10       1     ]'
A = [ 1/12  (-1-sqrt(5))/24      (-1+sqrt(5))/24      0     ]
    [ 1/12  (25+sqrt(5))/120     (25-13*sqrt(5))/120  0     ]
    [ 1/12  (25+13*sqrt(5))/120  (25-sqrt(5))/120     0     ]
    [ 1/12  (11-sqrt(5))/24      (11+sqrt(5))/24      0     ]
b = [ 1/12  5/12                 5/12                 1/12  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.66 Implicit 2 Stage 2nd order Lobatto C

Description:
```
Implicit 2 Stage 2nd order Lobatto C
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.11, pg 76
c = [  0    1   ]'
A = [ 1/2 -1/2  ]
    [ 1/2  1/2  ]
b = [ 1/2  1/2  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.67 Implicit 3 Stage 4th order Lobatto C

Description:
```
Implicit 3 Stage 4th order Lobatto C
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.11, pg 76
c = [  0    1/2    1   ]'
A = [ 1/6  -1/3   1/6  ]
    [ 1/6   5/12 -1/12 ]
    [ 1/6   2/3   1/6  ]
b = [ 1/6   2/3   1/6  ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.68 Implicit 4 Stage 6th order Lobatto C

Description:
```
Implicit 4 Stage 6th order Lobatto C
A-stable
Solving Ordinary Differential Equations II:
Stiff and Differential-Algebraic Problems,
2nd Revised Edition
E. Hairer and G. Wanner
Table 5.12, pg 76
c = [ 0     (5-sqrt(5))/10       (5+sqrt(5))/10       1          ]'
A = [ 1/12  -sqrt(5)/12          sqrt(5)/12          -1/12       ]
    [ 1/12  1/4                  (10-7*sqrt(5))/60   sqrt(5)/60  ]
    [ 1/12  (10+7*sqrt(5))/60    1/4                 -sqrt(5)/60 ]
    [ 1/12  5/12                 5/12                 1/12       ]
b = [ 1/12  5/12                 5/12                 1/12       ]'
```

Parent(s): Runge Kutta Butcher Tableau Selection (Section 5.30)

Child(ren): None.

Parameters: None.

5.69 Interpolation Buffer Settings

Description: This parameter list sets various parameters for the InterpolationBuffer.

Parent(s): Integrator Base (Section 5.1)

Child(ren): Trailing Interpolation Buffer Selection (Section 5.70)
Interpolation Buffer Appender Selection (Section 5.72)
Interpolator Selection (Section 5.74)

Parameters: None.

5.70 Trailing Interpolation Buffer Selection

Description: Used to select the Interpolation Buffer.

Parent(s): Interpolation Buffer Settings (Section 5.69)

Child(ren): Interpolation Buffer (Section 5.71)

Parameters: **Interpolation Buffer Type = None** Determines the type of Rythmos::InterpolationBuffer object that will be built. The parameters for each Interpolation Buffer Type are specified in this sublist

Valid std::string values:
"None"
"Interpolation Buffer"

5.71 Interpolation Buffer

Description: Sets parameters for the Interpolation Buffer.

Parent(s): Trailing Interpolation Buffer Selection (Section 5.70)

Child(ren): None.

Parameters: **InterpolationBufferPolicy = Keep Newest Policy** Interpolation Buffer Policy for when the maximum storage size is exceeded. Static will throw an exception when the storage limit is exceeded. Keep Newest will over-write the oldest data in the buffer when the storage limit is exceeded.

Valid std::string values:
"Invalid Policy"
"Static Policy"
"Keep Newest Policy"

StorageLimit = 0 Storage limit for the interpolation buffer.

5.72 Interpolation Buffer Appender Selection

Description: Used to select the Interpolation Buffer Appender.

Parent(s): Interpolation Buffer Settings (Section 5.69)

Child(ren): Pointwise Interpolation Buffer Appender (Section 5.73)

Parameters: **Interpolation Buffer Appender Type = None** Determines the type of Rythmos::InterpolationBufferAppender object that will be built. The parameters for each Interpolation Buffer Appender Type are specified in this sublist

Valid std::string values:
"None"
"Pointwise Interpolation Buffer Appender"

5.73 Pointwise Interpolation Buffer Appender

Description: Appender that just transfers nodes without any regard for accuracy or order.

Parent(s): Interpolation Buffer Appender Selection (Section 5.72)

Child(ren): None.

Parameters: None.

5.74 Interpolator Selection

Description: Choose the interpolator to use.

Parent(s): Interpolation Buffer Settings (Section 5.69)

Child(ren): Linear Interpolator (Section 5.75)
Hermite Interpolator (Section 5.76)
Cubic Spline Interpolator (Section 5.77)

Parameters: **Interpolator Type = None** Determines the type of Rythmos::Interpolator object that will be built. The parameters for each Interpolator Type are specified in this sublist

Valid std::string values:
"None"
"Linear Interpolator"
"Hermite Interpolator"
"Cubic Spline Interpolator"

5.75 Linear Interpolator

Description: This provides a simple linear interpolation between time nodes.

Parent(s): Interpolator Selection (Section 5.74)

Child(ren): None.

Parameters: None.

5.76 Hermite Interpolator

Description: This provides a piecewise cubic Hermite interpolation on each interval where the data is the solution and its time derivatives at the end points of the interval. It will match 3rd degree polynomials exactly with both function values and derivatives.

Parent(s): Interpolator Selection (Section 5.74)

Child(ren): None.

Parameters: None.

5.77 Cubic Spline Interpolator

Description: This provides a cubic spline interpolation between time nodes.

Parent(s): Interpolator Selection (Section 5.74)

Child(ren): None.

Parameters: None.

6 Convergence Test Examples

6.1 Dahlquist Test Equation

The Dahlquist Test equation,

$$\dot{x} = \lambda x, \tag{13}$$

is used to investigate and classify the properties of integrators. Its solution is $x = ce^{\lambda t}$ where λ can be complex with positive, negative coefficients, and provides a variety of canonical problems (*e.g.*, exponentially increasing and decaying solutions, and oscillatory solutions). By implementing time integrators in the Dahlquist test equation, the amplification or stability factor, $R(z)$, can be determined by writing it in the update form

$$x_n = R(\lambda \Delta t)\, x_{n-1} = R(z)\, x_{n-1}$$

where $z = \lambda \Delta t$. A few important measures of stability are absolute stability, A-stability and L-stability and are defined by z and $R(z)$. First off, absolute stability requires the solution has no growth from one time step to the next,

$$|x_n| \leq |x_{n-1}|$$

and this true when

$$|R(z)| \leq 1$$

which defines the stability domain. A-stable methods have a stability domain for the negative half domain where $Re(z) \leq 0$. Although A-stability is useful for many non-stiff equations, stiff equations require additional stability properties to A-stability, $|R(z)| \to 0$ as $z \to -\infty$, and is known as L-stability.

6.2 SinCos Problem

This is a canonical Sine-Cosine differential equation

$$\ddot{\mathbf{x}} = -\mathbf{x}$$

with a few enhancements. We start with the exact solution to the differential equation

$$\begin{aligned} x_0(t) &= a + b * \sin((f/L) * t + \phi) \\ x_1(t) &= b * (f/L) * \cos((f/L) * t + \phi) \end{aligned}$$

then the form of the model is:

$$\begin{aligned} \frac{d}{dt} x_0(t) &= x_1(t) \\ \frac{d}{dt} x_1(t) &= \left(\frac{f}{L}\right)^2 (a - x_0(t)) \end{aligned}$$

where the default parameter values are $a = 0$, $f = 1$, and $L = 1$, and the initial conditions

$$\begin{aligned} x_0(t_0 = 0) &= \gamma_0[= 0] \\ x_1(t_0 = 0) &= \gamma_1[= 1] \end{aligned}$$

determine the remaining coefficients

$$\begin{aligned} \phi &= \arctan(((f/L)/\gamma_1) * (\gamma_0 - a)) - (f/L) * t_0[= 0] \\ b &= \gamma_1/((f/L) * cos((f/L) * t_0 + \phi))[= 1] \end{aligned}$$

The solution is shown in Fig. 27. Therefore this model has three model parameters and two initial

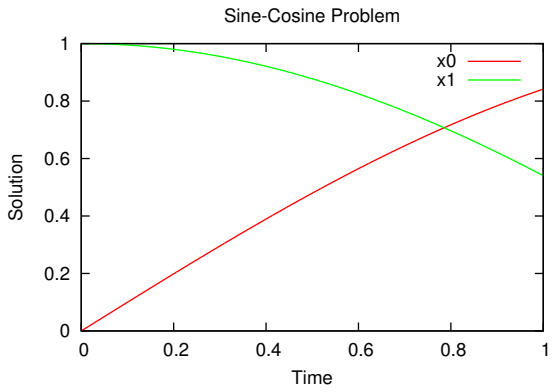


Figure 27: Exact solution to the Sine-Cosine problem.

conditions which effect the exact solution as above.

$$\mathbf{p} = (a, f, L)$$

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, t, \mathbf{p})$$

where

$$\begin{aligned} F_0 &= x_1 \\ F_1 &= \left(\frac{f}{L}\right)^2 (a - x_0) \end{aligned}$$

The exact sensitivities, $\mathbf{s} = \partial\mathbf{x}/\partial\mathbf{p}$, for the problem are specified as

$$\mathbf{s}(t) = \begin{bmatrix} 1 & 0 \\ \left(\frac{b}{L}\right) t \cos\left(\left(\frac{f}{L}\right) t + \phi\right) & \left(\frac{b}{L}\right) \cos\left(\left(\frac{f}{L}\right) t + \phi\right) - \frac{b\, f\, t}{L^2} \sin\left(\left(\frac{f}{L}\right) t + \phi\right) \\ -\frac{b\, f\, t}{L^2} \cos\left(\left(\frac{f}{L}\right) t + \phi\right) & -\frac{b\, f}{L^2} \cos\left(\left(\frac{f}{L}\right) t + \phi\right) + \frac{b\, f^2\, t}{L^3} \sin\left(\left(\frac{f}{L}\right) t + \phi\right) \end{bmatrix}$$

and for the default initial conditions, $\phi = 0$ and $b = 1$

$$\mathbf{s}(t = 0) = \begin{bmatrix} 1 & 0 \\ 0 & \frac{b}{L} \\ 0 & -\frac{f}{L^2} \end{bmatrix}$$

The time differentiated sensitivities, $\dot{\mathbf{s}} = \partial\mathbf{s}/\partial t = \partial/\partial t(\partial\mathbf{x}/\partial\mathbf{p}) = \partial/\partial\mathbf{p}(\partial\mathbf{x}/\partial t)$ are

$$\dot{\mathbf{s}}(t) = \begin{bmatrix} 0 & 0 \\ \left(\frac{b}{L}\right) \cos\left(\left(\frac{f}{L}\right) t + \phi\right) - \frac{b\, f\, t}{L^2} \sin\left(\left(\frac{f}{L}\right) t + \phi\right) & -\frac{2b\, f}{L^2} \sin\left(\left(\frac{f}{L}\right) t + \phi\right) \left(\frac{b}{L}\right) - \frac{b\, f^2\, t}{L^3} \cos\left(\left(\frac{f}{L}\right) t + \phi\right) \\ -\frac{b\, f}{L^2} \cos\left(\left(\frac{f}{L}\right) t + \phi\right) + \frac{b\, f^2\, t}{L^3} \sin\left(\left(\frac{f}{L}\right) t + \phi\right) & \frac{2b\, f^2}{L^3} \sin\left(\left(\frac{f}{L}\right) t + \phi\right) + \frac{b\, f^3\, t}{L^4} \cos\left(\left(\frac{f}{L}\right) t + \phi\right) \end{bmatrix}$$

6.3 Log-Time Problem

This problem was generated to be similar to those seen in Charon. What is different about this solution is that it evolves on logarithmic scale. As seen in Fig. 28, the solution has a very sharp gradient near $t = 10^{-9}$ and then decays over many order of magnitude until near constant at $t = 1$. From Fig. 28, one can see the challenge to capture the strong transient for $10^{-10} \leq t < 10^{-8}$ requiring step sizes at least as small as $\Delta t = 10^{-11}$, and then transition to larger and larger time steps as $t \to 1$ to reduce the overall computational costs.

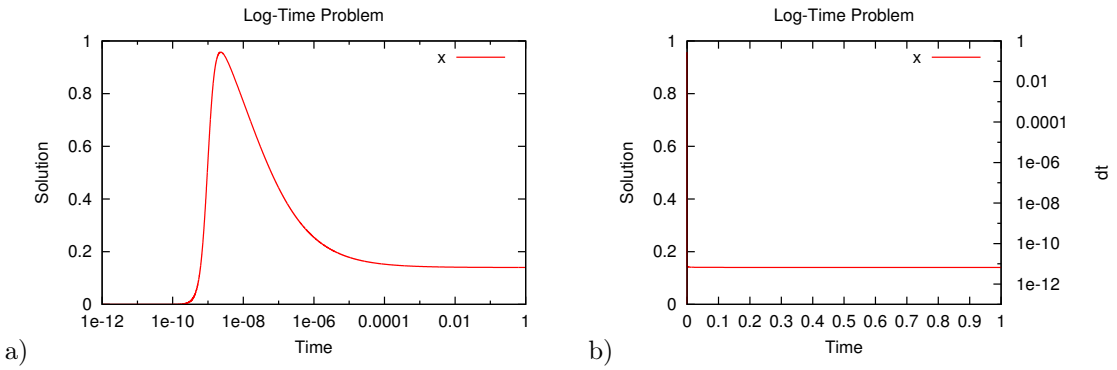


Figure 28: Exact solution to the Log-Time problem with a) logarithmic x-axis and b) linear x-axis.

The ODE for this problem is

$$x(t) = \frac{a\left(bt^4 + ct^{9/2}\right)}{\left(b + \sqrt{t}\right)\left(d + t^4\right)}$$

where $a = 1.4$, $b = 0.0001$, $c = 0.1$, $d = 10^{-36}$, and the initial condition is $x(0) = 0$. The center of the rapid increase is approximately $d^{1/4}$, the product ac sets the steady state value, and the center of the decay is approximately b^2. The time derivative is given by

$$\dot{x} = \frac{at^3\left(8b^2d + b\sqrt{t}\left((9c+7)d + (c-1)t^4\right) + 8cdt\right)}{2\left(b + \sqrt{t}\right)^2\left(d + t^4\right)^2}.$$

Part III
Developer's Guide

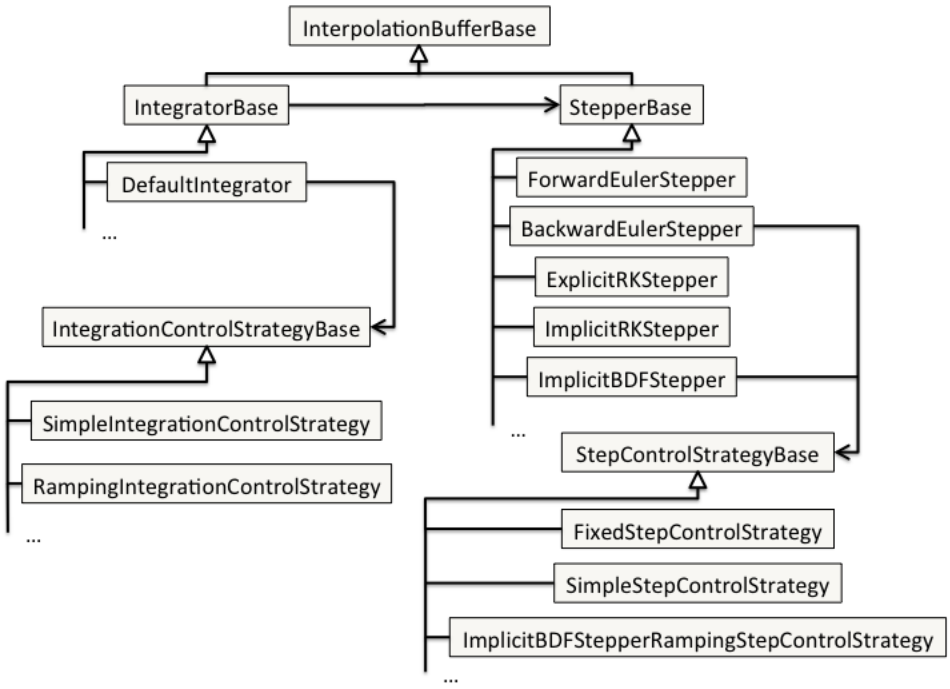


Figure 29: General relationships between the Integrators, IntegrationControlStrategies, Steppers and StepControlStrategies.

7 Introduction

The primary purpose of the Developer's Guide is to provide the high-level OO design and description of the C++ interfaces and concrete implementations for the solution of a broad class of transient ordinary differential equations (ODEs) and differential algebraic equations (DAEs) in a consistent manner. Although Doxygen is a good tool to explore the details of the C++ interfaces and classes, it rarely provides the high-level interaction of these objects and how they are intended to be used. Therefore in this Developer's Guide, the high-level description is given in order for new Rythmos developers can get the basic design of Rythmos. It is intended that this description should remain fairly stable over time, and not require many updates. However detailed information, which could change more often, is left to the Doxygen webpages, which obviously provides quick and easy document generation of software implementation.

7.1 Interpolation-Buffer Base

An InterpolationBuffer is basically a container of solutions and time derivatives ("states") at discrete times ("time nodes") over a time range in ascending order, and can be used for storing checkpoints and breakpoints of the state for later use. With an Interpolator specified, a state at desired "time points" can be returned over the time range with an accuracy of the order of the Interpolator. See Fig 30.

Part of the design here is to separate the state information (solutions and time derivatives) from how the state is integrated to that time. This can be used when separate ODEs and DAEs with different time integrators are advanced in time, and wish to share their states (e.g., forward and adjoint states, and operator-split solutions). No direct synchronizing of time steps is required between them; time steps of each state can be interpolated between for each integrator.

An important caveat here is that the interpolated solutions are not guaranteed to satisfy the ODE/DAE at the interpolated "time points". This may be important for simulations where conser-

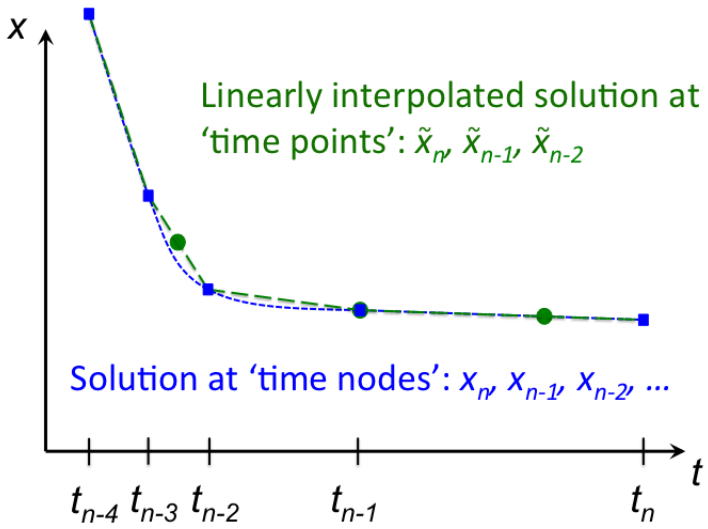


Figure 30: Illustrating the time points and time nodes in InterpolationBuffer with a linear interpolator.

vation is required, e.g., conservation of mass, momentum and energy. The InterpolationBuffer can be still useful in this situation, but the "time nodes" should be used to ensure that the state satisfies the governing ODEs/DAEs.

7.2 Integrator Base

Time integration is carried out by Integrators, which inherit from InterpolationBufferBase, and advance the state in time (step-by-step) with Steppers. Integrators therefore have a set of states at given times (see Sec. 7.1), and can provide states to client requests. If the client requests a time point within the current time range, the Integrator can simply return an interpolated state. If the client requests a point forward in time, the Integrator will advance the state to include the requested time.

The primary function of the Integrator is to manage the "high-level" time coordination required by the clients. Examples of this time marching would include output of the solution state at various times, managing checkpoints between solution states and adjoint states, coordinating states between operator-split solutions, etc. All these coordinations have very little to do with individual time steps, and more with times where the solution states are required. This helps define the difference between Integrators and Steppers when it comes to step-size selection (Integration and Step Control Strategies).

In general the Integrator will request the largest step size required to meet the next "coordination" (see Section 7.3), and leave the leave final determination of the step size to the Stepper based on client specification, e.g., fixed or variable step size, order, and accuracy (see Section 7.5). Many times the step size completed by the Stepper will not advance the state to the Integrator requested time, and the Integrator will simply request another time step from the Stepper. With that said, the Integrator has the flexibility to more directly control the time stepping, and control the step size used by the Steppers. However allowing the Stepper to choose the step size based on the Step Control Strategy and having the Integration Control Strategy select "coordination" times is the preferred approach.

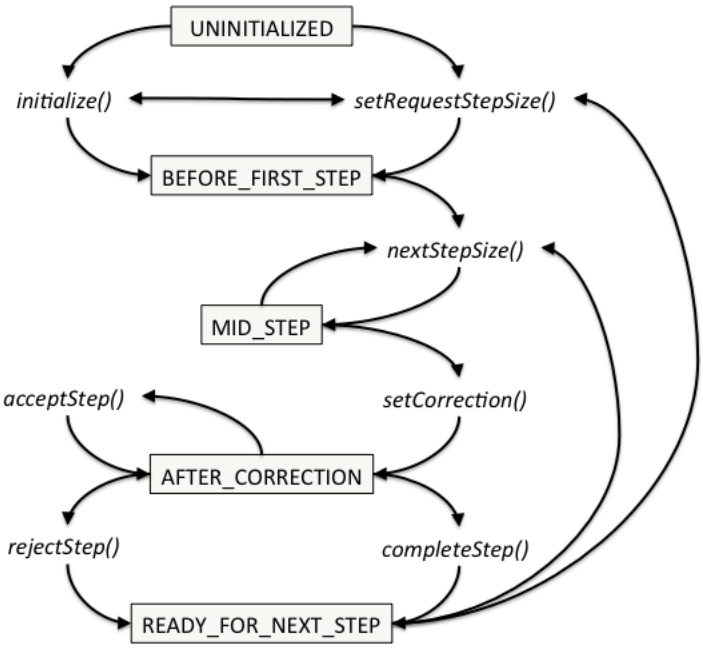


Figure 31: Progression of the ControlStrategyState through a sequence of StepControlStrategy functions by a Stepper.

7.3 Integration Control-Strategy

The Integration Control-Strategy is meant to determine the step size needed for "high-level" simulation coordination (e.g., operator splitting), solution output, and/or Stepper failure. In the Integration Control-Strategy, we are not concerned with the accuracy of the solution, as much as we are concerned with when are solutions needed. Obviously one could control the Stepper step size with the Integration control-strategy by using a *fixed* step type and requiring the desired step size be taken. This is especially useful for Steppers that can only handle *fixed* step types.

7.4 Stepper Base

A Stepper is a basic building block for DAE/ODE solvers. It advances the state by ONE time step,

$$x_{n-1} \xrightarrow[\Delta t]{} x_n,$$

that has been requested by the Integrator. This time step can be fixed or variable, where a *fixed* step size indicates the stepper will take the requested step size or fail, and where a *variable* step size allows the Stepper to adjust the step size to meet accuracy and/or order requirements, and report to the Integrator the step size taken. Most Steppers can be implemented very quickly for fixed step sizes, and allow the Integrator to control the step size through an Integration Control-Strategy (see Sec. 7.3). For variable step sizes, the Stepper needs a Step Control-Strategy (see Sec. 7.5) in order to adjust the step size based on error/order, and handle time step failures.

7.5 Step Control-Strategy

Step Control-Strategies are for adjusting the step size when a Stepper is requested to take a variable step size by an Integrator. In order to handle the correct progression of calls to Step Control-

Strategies functions (see Fig. 31), a ControlStrategyState is used, and can have the following values, `UNINITIALIZED`, `BEFORE_FIRST_STEP`, `MID_STEP`, `AFTER_CORRECTION`, and `READY_FOR_NEXT STEP`.

The functions *initialize()* and *setRequestedStepsize()* are used to prepare for the first step each time the Integrator calls the Stepper, i.e., setting the requested step size and initializing the Error Weight Vector. The *nextStepsize()* function is used to complete anything before starting the step. The *setCorrection()* functions is used to perform anything after the solver. i.e., the correction. At this point the Stepper uses the StepControlStrategy to determine if the step was acceptable with *acceptStep()*. If the step was acceptable, final computations of the solution and time derivative are completed for the next time step by *completeStep()*. If the step was not acceptable, the step size and/order could be adjusted by the StepControlStrategy function, *rejectStep()*, and the time step retried in order to obtain an acceptable time step. The *rejectStep()* function could completely reject the step and return control to the Integrator to see if the Integrator Control Strategy can do something to revive the time step.

Part IV
Back Matter

Index

References

[1] Sundials website. `http://sundials.wikidot.com/bdf-flc-derivation`, April 2007.

[2] U. M. Ascher and L. R. Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*. SIAM, Philadelphia, 1998.

[3] K. E. Brennan, S. L. Campbell, and L. R. Petzold. *Numerical solution of initial-value problems in differential-algebraic equations*, volume 14 of *Classics in Applied Mathematics*. SIAM, Philadelphia, PA, first edition, 1996.

[4] P. N. Brown, A. C. Hindmarsh, and L. R. Petzold. Consistent initial condition calculation for differential-algebraic systems. *SIAM J. Sci. Comput.*, 19:1495–1512, 1998.

[5] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation*, 67:73–85, January 1998.

[6] P. M. Gresho and R. L. Sani. *Incompressible Flow and Finite Element Method*, volume One Advection-Diffusion. John Wiley and Sons, New York, 2000.

[7] P. M. Gresho and R. L. Sani. *Incompressible Flow and Finite Element Method*, volume Two Isothermal Laminar Flow. John Wiley and Sons, New York, 2000.

[8] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Germany, second edition, 2000.

[9] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Germany, second edition, 1996.

[10] A. C. Hindmarsh. The PVODE and IDA algorithms. Technical Report UCRL-ID-141558, Lawrence Livermore National Laboratory, Livermore, CA, December 2000.

[11] K. R. Jackson and R. Sacks-Davis. An alternative implementation of variable step-size multistep formulas for stiff ODEs. *ACM Transactions on Mathematical Software*, 6(3):295–318, 1980.

[12] Phumezile Kama. *Non-standard finite-difference methods in dynamical systems*. PhD thesis, University of Pretoria, 2009.