

Obtaining Accurate Derivatives Across Adaptive-Step Integration

Kaushik Ponnappalli* and Robert D Falck†
NASA Glenn Research Center, Cleveland, OH, 44233

Obtaining derivatives through the integration of a set of ordinary differential equations is of interest to a number of fields of research. Approaches for doing so that are available in the literature, however, do not account for details associated with variable step integrator that cause inaccuracies in the total derivatives. In this paper, we explore the causes of the accumulation of derivative error in adaptive-step Runge-Kutta integration schemes and propose a solution to eliminate them. The new algorithm proposed is a transformation of each step of a Runge-Kutta integration into a set of residual equations. This work demonstrates the improvement in accuracy of total derivatives when the new algorithm is utilized.

Nomenclature

f	=	ODE function
x	=	state vector
t	=	time
θ	=	ODE parameters
$\mathcal{I}$	=	integration function
z	=	integration parameters
h	=	step size
N	=	number of integration steps
X	=	state at intermediate stages
T	=	time at intermediate stages
k	=	state rate at intermediate stages
a	=	Runge-Kutta matrix
b	=	Runge-Kutta weights
c	=	Runge-Kutta nodes
Δ	=	error at each iteration of the integration
ϵ	=	integration error tolerance
$\mathcal{R}_*$	=	residual of the specified variable
u	=	outputs of the residuals
y	=	inputs to the residuals
r_x	=	position in x-coordinate
r_y	=	position in y-coordinate
v_x	=	velocity in x-coordinate
v_y	=	velocity in y-coordinate
μ	=	standard gravitational parameter
e	=	eccentricity
ε	=	error in integration results or derivatives

I. Introduction

THE notion of obtaining derivatives through the integration of a set of ordinary differential equations (ODEs) has gained more interest in recent years due to the rise in machine learning. Obtaining derivatives of such a process is

*Aerospace Engineer, HX5 LLC, kaushik.s.ponnappalli@nasa.gov, AIAA Member.

†Aerospace Engineer, Mission Analysis and Architecture Branch, robert.d.falck@nasa.gov, AIAA Member.

also of interest in trajectory optimization, whether as a stand-alone problem or as a part of multidisciplinary optimization. Here the derivatives may be used to guide gradient-based optimization of a solution [1].

Numerous approaches for computing the derivatives have been put forth in the literature [2], [3], [4]. Reference [2] utilizes a *direct* approach while references [3] and [4] use an *adjoint* approach. A direct approach involves integrating ODE and simultaneously computing the derivatives of the outputs. Whereas an adjoint approach involves back propagation through time to determine the derivatives. In [5], the authors demonstrate accuracy issues in calculating the gradients due to the application of a so-called "continuous adjoint" or "optimize then discretize" approach. Applying a continuous approximation to the calculation of gradients of a discrete process, such as numerical integration, can introduce discrepancies that may impede numerical optimization. These discrepancies are often small and inconsequential, but it is not easy to know a priori if they will be problematic in a given application.

In reference [2], Pellegrini and Russell account for this discrepancy but note that using a fixed-step integration yields more accurate derivatives than using an adaptive step algorithm. Many applications can accommodate fixed-step integration methods, but the propagation of spacecraft trajectories is typically not one of those applications. Limiting integration to fixed-step integration will result in analysis times that are much longer than necessary, inaccurate results, or both. To account for this, some have turned to time regularization techniques, such as the Sundman transformation [6], for astrodynamics applications [7]. The Sundman transformations allows accuracy to be maintained across a fixed-step integration technique such that accurate derivatives can be maintained. Such transformations are not necessarily available for general systems, and in the case of multidisciplinary optimization may require repeated transformations between time and the new independent variable.

In this paper, we will explore the causes of the accumulation of derivative error in adaptive-step Runge-Kutta integration schemes. We will then discuss a solution that significantly improves the accuracy of the derivatives when computed via a direct approach.

II. Theory

A. The Propagation of Derivatives across an ODE Integration

Suppose we would like to integrate some ordinary differential equation that provides the rates of change of the state variables $\dot{x}$ as a function of the current state x , the current time t , and some set of parameters θ .

$$\dot{x} = f(x, t, \theta) \quad (1)$$

In the course of our analysis, we need to integrate this equation from some initial time t_0 for some duration t_d .

$$x_f = \int_{t_0}^{t_0+t_d} f(x, t, \theta) dt = \mathcal{I}(z) \quad (2)$$

Here we treat the integration of the ODE as a function $\mathcal{I}$ and all the parameters of the integration function are packaged into some single array-valued argument z .

$$z = \begin{bmatrix} x_0 & t_0 & t_d & \theta \end{bmatrix} \quad (3)$$

Typically, x_f is obtained by numerically integrating (1). There are a variety of options available for numerical integration schemes but in this paper we focus on Runge-Kutta methods. For a general fixed-step Runge-Kutta (RK) method, the equations of the step update are

$$x_q = x_{q-1} + h_{q-1} \sum_{i=0}^s b_i k_{qi} \quad (4)$$

$$t_q = t_{q-1} + h_{q-1} \quad (5)$$

Where, s is the number of stages the integration is split into and k , the state rate at each stage of the given RK method is computed as

$$T_i = t_{q-1} + c_i h_{q-1} \quad (6)$$

$$X_i = x_{q-1} + h_{q-1} \sum_{j=0}^{i-1} a_{ij} k_{qj} \quad (7)$$

$$k_{qi} = f(T_i, X_i, \theta) \quad (8)$$

with T_i and X_i being the time and state respectively at the intermediate stages. The step size is computed as

$$h(t_d) = \frac{t_d}{N} \quad (9)$$

where N is the pre-specified number of steps.

The a , b , and c terms are coefficients of the given RK scheme. The coefficients for any given RK scheme are often laid out in a *Butcher tableau*.

$$\begin{array}{c|cccc} c_0 & a_{11} & a_{12} & \dots & a_{1s} \\ c_1 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \quad (10)$$

We seek the derivatives of the final state vector x_f and final time t_f with respect to the parameters of the integration z . Differentiating (4) with respect to z yields

$$\left[\frac{dx_q}{dz} \right] = \left[\frac{\partial x_q}{\partial z} \right] + \left[\frac{\partial x_q}{\partial x_{q-1}} \right] \left[\frac{dx_{q-1}}{dz} \right] + \left[\frac{\partial x_q}{\partial h_{q-1}} \right] \left[\frac{dh_{q-1}}{dz} \right] + \sum_{i=0}^s \left[\frac{\partial x_q}{\partial k_{qi}} \right] \left[\frac{dk_{qi}}{dz} \right] \quad (11)$$

Substituting the partial derivatives of (4) simplifies this to

$$\left[\frac{dx_q}{dz} \right] = \left[\frac{dx_{q-1}}{dz} \right] + \left[\sum_{i=0}^s b_i k_{qi} \right] \left[\frac{dh_{q-1}}{dz} \right] + h_{q-1} \sum_{i=0}^s b_i \left[\frac{dk_{qi}}{dz} \right] \quad (12)$$

Derivatives of the k_{qi} terms with respect to the integration inputs are

$$\left[\frac{dk_{qi}}{dz} \right] = \left[\frac{\partial k_{qi}}{\partial z} \right] + \left[\frac{\partial k_{qi}}{\partial T_i} \right] \left[\frac{dT_i}{dz} \right] + \left[\frac{\partial k_{qi}}{\partial X_i} \right] \left[\frac{dX_i}{dz} \right] + \left[\frac{\partial k_{qi}}{\partial \theta} \right] \left[\frac{d\theta}{dz} \right] \quad (13)$$

$$\left[\frac{dk_{qi}}{dz} \right] = f_t(T_i, X_i, \theta) \left[\frac{dT_i}{dz} \right] + f_x(T_i, X_i, \theta) \left[\frac{dX_i}{dz} \right] + f_\theta(T_i, X_i, \theta) \left[\frac{d\theta}{dz} \right] \quad (14)$$

The derivative of the intermediate time with respect to the integration parameters is

$$\left[\frac{dT_i}{dz} \right] = \left[\frac{\partial T_i}{\partial z} \right] + \left[\frac{\partial T_i}{\partial t_{q-1}} \right] \left[\frac{dt_{q-1}}{dz} \right] + \left[\frac{\partial T_i}{\partial h_{q-1}} \right] \left[\frac{dh_{q-1}}{dz} \right] \quad (15)$$

$$\left[\frac{dT_i}{dz} \right] = [0] + [1] \left[\frac{dt_{q-1}}{dz} \right] + c_i \left[\frac{dh_{q-1}}{dz} \right] \quad (16)$$

and the derivatives of the intermediate states with respect to the integration parameters are

$$\left[\frac{dX_i}{dz} \right] = \left[\frac{\partial X_i}{\partial z} \right] + \left[\frac{\partial X_i}{\partial x_{q-1}} \right] \left[\frac{dx_{q-1}}{dz} \right] + \left[\frac{\partial X_i}{\partial h_{q-1}} \right] \left[\frac{dh_{q-1}}{dz} \right] + \left[\frac{\partial X_i}{\partial k_{qj}} \right] \left[\frac{dk_{qj}}{dz} \right] \quad (17)$$

$$\left[\frac{dX_i}{dz} \right] = [0] + [I_n] \left[\frac{dx_{q-1}}{dz} \right] + \sum_{j=0}^{i-1} a_{ij} k_{qj} \left[\frac{dh_{q-1}}{dz} \right] + h_{q-1} \sum_{j=0}^{i-1} a_{ij} \left[\frac{dk_{qj}}{dz} \right] \quad (18)$$

Repeating a similar procedure and differentiating (5) yields

$$\left[\frac{d t_q}{d z} \right] = \left[\frac{\partial t_q}{\partial z} \right] + \left[\frac{\partial t_q}{\partial t_{q-1}} \right] \left[\frac{d t_{q-1}}{d z} \right] + \left[\frac{\partial t_q}{\partial h_{q-1}} \right] \left[\frac{d h_{q-1}}{d z} \right] \quad (19)$$

which can be simplified to

$$\left[\frac{d t_q}{d z} \right] = \left[\frac{d t_{q-1}}{d z} \right] + \left[\frac{d h_{q-1}}{d z} \right] \quad (20)$$

by substituting the partial derivatives of (5).

The equations in (12) and (20) serve as update equations to the derivatives of x and t . The initial conditions for these derivatives are

$$\left[\frac{d x_0}{d z} \right] = \underbrace{\begin{bmatrix} 0 & 0 & \dots & 0 \\ [I] & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}}_{n \times (n+2+m)} \quad (21)$$

$$\left[\frac{d t_0}{d z} \right] = \underbrace{\begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}}_{1 \times (n+2+m)} \quad (22)$$

for a system with n states and m parameters. The identity matrix is of size $n \times n$ and corresponds to the derivative of x_0 with respect to x_0 . Similarly, in 22, the one corresponds to derivative of t_0 with respect to itself

The final remaining unknown, the derivative of the step size, is of specific interest. For a fixed-step method this derivative is obtained by differentiating (9) with respect to z

$$\left[\frac{d h_{q-1}}{d z} \right] = \underbrace{\begin{bmatrix} 0 & \dots & 0 & 1/N & 0 & \dots & 0 \end{bmatrix}}_{1 \times (n+2+m)} \quad (23)$$

with the nonzero element corresponding to the derivative of h_{q-1} with respect to t_d .

This procedure is correct for fixed-step integration techniques, but for adaptive-step integration methods it misses the nuance that the integrator is changing h over the course of the integration. Some error will accumulate in the derivatives due to changing step size. With adaptive-step techniques, h is itself a function of z and thus we cannot treat the terms in (23) as constants.

B. The Implicit Behavior of Adaptive-Step Runge-Kutta Methods

Adaptive-step Runge-Kutta techniques seek to find some "optimal" step-size h^* such that the step-size incurs some amount of relative error provided by the user, ϵ [8]. Steps larger than h^* incur an integration error that is deemed unacceptably large and are rejected. Steps smaller than h^* are smaller than necessary and thus result in an unnecessarily large number of integration steps. This is done by computing the step using two methods, one of order p and one of order $p + 1$, and the error is then computed as the difference between the two steps. In order to achieve this, the Butcher Tableau is augmented with an additional coefficient, b^* , corresponding to the higher-order step

$$\begin{array}{c|cccc} c_0 & a_{11} & a_{12} & \dots & a_{1s} \\ c_1 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \\ & b_1^* & b_2^* & \dots & b_s^* \end{array} \quad (24)$$

In addition, to the step computed in (4), the additional step is computed as

$$x_q^* = x_{q-1} + h_{q-1} \sum_{i=0}^s b_i^* k_{qi} \quad (25)$$

and the error at the step is computed as the magnitude of the difference of (4) and (25)

$$\Delta = \left| \sum_{i=0}^s (b_i^* - b_i) k_{qi} \right| \quad (26)$$

In this paper, we use the equivalent expression

$$\Delta = \sqrt{\left((b^* - b)^T k_q \right)^2} \quad (27)$$

At each step, the computed error must be below some specified tolerance ϵ . If the error is larger than the tolerance, then the step is rejected. Regardless of whether the step is rejected, the step size is adjusted based on the error. The step update considered here is

$$h_q = 0.84 h_{q-1} \left(\frac{\epsilon}{\Delta} \right)^{1/4} \quad (28)$$

Differentiating (28) yields the derivatives

$$\left[\frac{d h_q}{d z} \right] = \left[\frac{\partial h_q}{\partial z} \right] + \left[\frac{\partial h_q}{\partial h_{q-1}} \right] \left[\frac{d h_{q-1}}{d z} \right] + \left[\frac{\partial h_q}{\partial k_{qi}} \right] \left[\frac{d k_{qi}}{d z} \right] \quad (29)$$

where

$$\left[\frac{\partial h_q}{\partial h_{q-1}} \right] = 0.84 \left(\frac{\epsilon}{\Delta} \right)^{1/4} \quad (30)$$

$$\left[\frac{\partial h_q}{\partial k_{qi}} \right] = \frac{-0.21 \epsilon^{1/4} h_q (b^* - b)^2 k_{qi}}{((b^* - b) k_{qi})^{9/8}} \quad (31)$$

In cases where, the step size from the previous iteration, h_{q-1} , is small enough to meet the required error tolerance, this computation is purely explicit. So the partials derivatives as computed in (30) and (31) may be applied. However, in the cases where the step size is too large and the step update must be rejected, the computation of h is implicit since it must be iterated upon to converge the error to the tolerance. In such cases, it would be incorrect to use the derivatives in (30) and (31), since explicit behavior is assumed in their computation. This alternation between explicit and implicit behavior creates a challenge to finding a single expression for the derivative of the step size. As such, this paper proposes a reformulation of adaptive-step RK methods with the computation of the step size at each iteration as a series of residual equations that must be driven to zero.

C. The Residual Equations

For the integration across the specified duration, the inputs are x_0 , t_0 , t_d , and θ , and the outputs are x_f and t_f . We can separate this by treating each iteration as a separate computation. Now the q -th iteration has inputs x_{q-1} , t_{q-1} , t_0 , t_d , and θ , and outputs x_q and t_q . The inputs are used to first compute h_q implicitly with the following residual.

$$\mathcal{R}_h(h_q, z) = \begin{cases} \Delta - \epsilon = 0 & \text{for } t + h_q^* < t_f \\ t + h_q - t_f = 0 & \text{for } t + h_q^* \geq t_f \end{cases} \quad (32)$$

The step size that is computed from solving this residual is used to compute the step update as

$$x_q = x_{q-1} + h_q \sum_{i=0}^s b_i k_{qi} \quad (33)$$

$$t_q = t_{q-1} + h_q \quad (34)$$

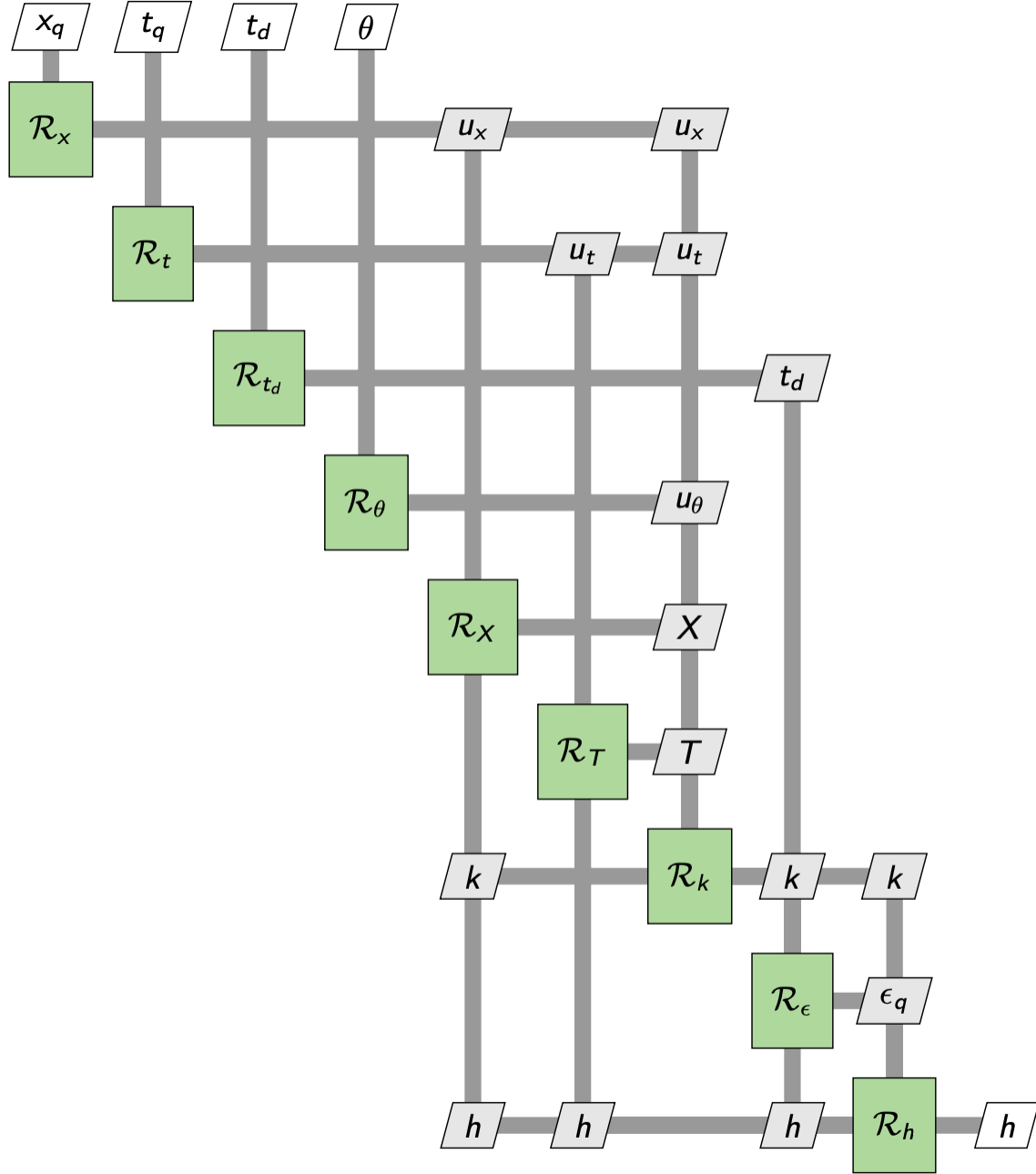


Fig. 1 Extended Design Structure Matrix (XDSM) of the reformulated equations to compute step size.

Note that these equations differ from (4) and (5) in the iteration index of the step size. This is to indicate that the new step size is computed prior to the step update rather than after.

Using the *implicit function theorem*, the total derivative of (32) is obtained by the procedure laid out by Martins and Hwang [9].

$$\left[\frac{d\mathcal{R}_h(h, z)}{dz} \right] = \left[\frac{\partial \mathcal{R}_h}{\partial z} \right] + \left[\frac{\partial \mathcal{R}_h}{\partial h} \right] \left[\frac{dh}{dz} \right] \quad (35)$$

If this residual is reliably eliminated for each value of z , then the total derivative of the residual with respect to z is zero. Thus, the total derivatives of the implicitly calculated step-size with respect to z is obtained from:

$$\left[\frac{\partial \mathcal{R}_h}{\partial z} \right] + \left[\frac{\partial \mathcal{R}_h}{\partial h} \right] \left[\frac{dh}{dz} \right] = 0 \quad (36)$$

$$\left[\frac{dh}{dz} \right] = - \left[\frac{\partial \mathcal{R}_h}{\partial h} \right]^{-1} \left[\frac{\partial \mathcal{R}_h}{\partial z} \right] \quad (37)$$

This is the so-called forward-mode of differentiation, because this linear system needs to be solved once for each value in z . The derivative of the end-states across a single step of the integration can also be computed using a reverse or adjoint differentiation.

$$\left[\frac{\partial \mathcal{R}_h}{\partial z} \right]^T + \left[\frac{dh}{dz} \right]^T \left[\frac{\partial \mathcal{R}_h}{\partial h} \right]^T = 0 \quad (38)$$

In this paper, we will focus on applying the forward-mode.

Directly computing the partials listed in (36) can be challenging due to the number of embedded calculations present. To simplify the computation, we separate $\mathcal{R}$ into a system of implicit equations. The structure of this series of calculations is laid out in Fig 1 This is achieved by transforming the explicit computations in equations (6)-(8) into residuals that must be converged,

$$\mathcal{R}_T(T, t_{q-1}, h_q) = T - t_{q-1} - c_i h \quad (39)$$

$$\mathcal{R}_X(X, x_{q-1}, k, h_q) = X - x_{q-1} - h_q \sum_{i=0}^s b_i k_{qi} \quad (40)$$

$$\mathcal{R}_k(k_q, X, T, \theta) = k_q - f(X, T, \theta) \quad (41)$$

The residual for the step size may now be written as

$$\mathcal{R}_h(h, k_q) = \sqrt{\left((b^* - b)^T k_q \right)^2} - \epsilon \quad (42)$$

An additional transformation that we apply an implicit transformation to the input variables contained in z . This defines the new residuals

$$\mathcal{R}_x(u_x, x_{q-1}) = u_x - x_{q-1} \quad (43)$$

$$\mathcal{R}_t(u_t, t_{q-1}) = u_t - t_{q-1} \quad (44)$$

$$\mathcal{R}_{t_d}(u_{t_d}, t_d) = u_{t_d} - t_d \quad (45)$$

$$\mathcal{R}_\theta(u_\theta, \theta) = u_\theta - \theta \quad (46)$$

where u_x , u_t , u_{t_d} and u_θ are the implicit versions of x_{q-1} , t_{q-1} , t_d and θ . The initial time, t_0 has been excluded since it does not appear explicitly at each iteration. This now allows equations (39)-(41) to be re-written as

$$\mathcal{R}_T(T, u_t, h_q) = T - u_t - c_i h \quad (47)$$

$$\mathcal{R}_X(X, u_x, k, h_q) = X - u_x - h_q \sum_{i=0}^s b_i k_{qi} \quad (48)$$

$$\mathcal{R}_k(k_q, X, T, u_\theta) = k_q - f(X, T, u_\theta) \quad (49)$$

These equations can be packaged into array valued arguments $\mathcal{R}$, y , and u as

$$\mathcal{R} = \begin{bmatrix} \mathcal{R}_x & \mathcal{R}_t & \mathcal{R}_{t_d} & \mathcal{R}_\theta & \mathcal{R}_X & \mathcal{R}_T & \mathcal{R}_k & \mathcal{R}_h \end{bmatrix} \quad (50)$$

$$u = \begin{bmatrix} u_x & u_t & u_{t_d} & u_\theta & X & T & k & h \end{bmatrix} \quad (51)$$

$$y = \begin{bmatrix} x_q & t_q & t_d & \theta \end{bmatrix} \quad (52)$$

where $\mathcal{R}$ is the array of residuals, u is array of outputs, and y is array of inputs.

Applying the implicit value theorem as we did in (35) yields

$$\left[\frac{du}{dy} \right] = - \left[\frac{\partial \mathcal{R}}{\partial u} \right]^{-1} \left[\frac{\partial \mathcal{R}}{\partial y} \right] \quad (53)$$

Solving this system of linear equations with converged values yields the derivatives of h and the other outputs with respect to x_{q-1} , $t_q - 1$, and θ . These derivatives with respect to z may be obtained as

$$\left[\frac{dh}{dz} \right] = \left[\frac{dh}{dx_{q-1}} \right] \left[\frac{dx_{q-1}}{dz} \right] + \left[\frac{dh}{dt_{q-1}} \right] \left[\frac{dt_{q-1}}{dz} \right] + \left[\frac{dh}{dt_d} \right] \left[\frac{dt_d}{dz} \right] + \left[\frac{dh}{d\theta} \right] \left[\frac{d\theta}{dz} \right] \quad (54)$$

D. Adaptive Tolerance

The approach detailed in II.C yields accurate derivatives but does so at the cost of integration accuracy as will be shown in Sec III. This is due to the fact that the error at each step is required to be at the tolerance rather than at or below the tolerance. To correct for this, we adjust the tolerance at each step based on an approach detailed in [10]. The residual in (42) is modified to

$$\mathcal{R}_h(h, k_q, \epsilon_q) = \sqrt{\left((b^* - b)^T k_q \right)^2} - \epsilon_q \quad (55)$$

where ϵ_q is computed from the residual

$$\mathcal{R}_\epsilon(h, t_d, \epsilon_q) = \epsilon_q - \epsilon \sqrt{\frac{h_q}{t_d}} \quad (56)$$

This approach allows for bounding the total error across the integration using ϵ .

III. Applications

To demonstrate the improvements achieved by the method presented in II.C and II.D, we will examine a test case. We will look at the propagation of a spacecraft under the two-body model. The desired derivatives of the outputs are compared against two other methods, the traditional variable step approach as described in II.B and complex step differentiation (CSD) [11], with CSD treated as the true values.

For both adaptive step approaches, a Runge-Kutta-Fehlberg with order seven and embedded order 8 (RK78) is used. The variable step integrators use $\epsilon = 10^{-9}$. For the new approach, the residuals are converged using a standard Newton solver with a tolerance of 10^{-14} .

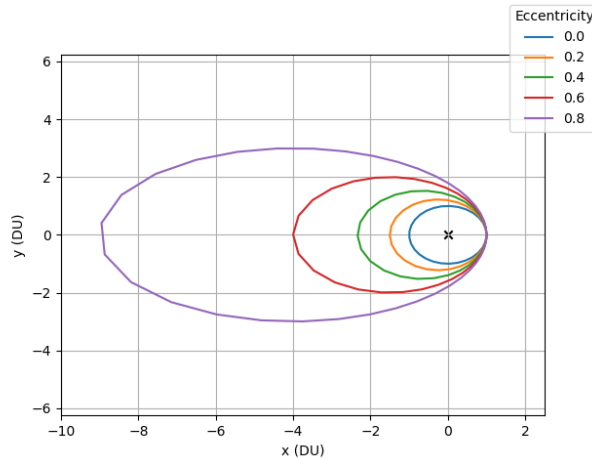


Fig. 2 Orbits propagated in the two-body gravitational model

A. Two-body Orbit Propagation

We are interested in the propagation of a two-dimensional orbit under the two body gravity model across multiple eccentricities. The equations of motion are

$$\begin{bmatrix} \dot{r}_x & \dot{r}_y & \dot{v}_x & \dot{v}_y \end{bmatrix} = \begin{bmatrix} v_x & v_y & -\frac{\mu}{(r_x^2+r_y^2)^{3/2}}r_x & -\frac{\mu}{(r_x^2+r_y^2)^{3/2}}r_y \end{bmatrix} \quad (57)$$

The initial conditions for the integration in normalized Length Units (LU) and Time Units (TU) are:

$$\begin{bmatrix} r_{x0} & r_{y0} & v_{x0} & v_{y0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \sqrt{1+e} \end{bmatrix} \quad (58)$$

where e is the eccentricity. The orbit is propagated for one revolution, that is, $t_d = 2\pi/\sqrt{(1-e)^3}$ TU. The standard gravitational parameter is $\mu = 1\text{LU}^3/\text{TU}^2$. This problem is run for the different eccentricities 0.0, 0.2, 0.4, 0.6, 0.8. The resulting orbits can be seen in Fig 2.

1. Accuracy of Integration

The results of the integration may be seen in Fig 3. The first subplot in Fig 3 shows the integration results of the new algorithm with a fixed tolerance compared to the new algorithms, one with a fixed tolerance and one with an adaptive tolerance. The accuracy of the final state value is determined by comparing against the known solution. Since the duration of propagation was selected to be the orbital period, the final state is expected to be same as the initial condition. As expected, we observe that all three approaches are more accurate at lower eccentricities and lose some precision at the more eccentric orbits. We observe that the traditional adaptive-step approach is slightly more accurate than the new algorithm when the adaptive tolerance is not used. The likely cause of this difference in accuracy is the fact that the new approach must be at the error tolerance at each iteration while the traditional approach may be at or below the error tolerance. However, when an adaptive tolerance is used, the new algorithm is more accurate than the traditional approach.

2. Accuracy of Derivatives

As a result of the integration, eight derivatives are computed; the derivatives of the two outputs, t_f and x_f , with respect to the four inputs x_0 , t_0 , t_d , θ . For the purpose of this paper will focus only on the derivatives of x_f . The accuracy of the different approaches may be seen Fig 3. We can observe that both the new approaches yield significantly more accurate derivatives than the traditional adaptive-step approach in most cases. The derivative of x_f with respect to t_d is effectively the same between the two methods because the duration of the integration t_d has no impact on the step size until the end of the propagation. The derivative of x_f with respect to t_0 is omitted due to the fact that it is identically zero for the time-invariant ODE being integrated. We can also note that both of the new approaches are similar in the accuracy of derivatives they yield. This indicates that the choice of whether or not to use an adaptive tolerance depends only on the desired accuracy of the integration and not the accuracy of the derivatives obtained.

3. Impact of Changing Step Size

To further examine the impact of the changing step size on the accuracy of the derivatives, we will isolate one of the orbits and examine it more closely. We will look at the case where $e = 0.6$. Fig 4 shows the error in the state derivatives at each time step for the traditional approach and the new approach with a fixed tolerance. We may note a few things from these results.

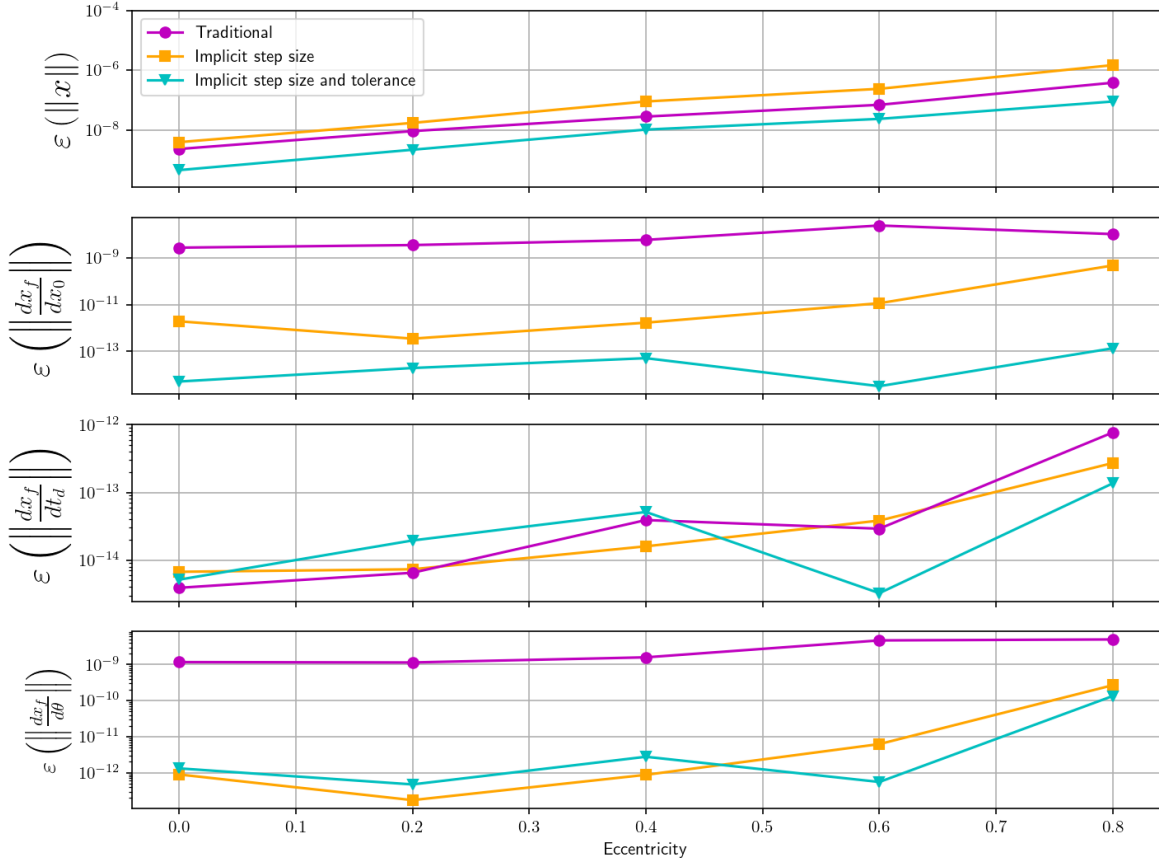


Fig. 3 Error in the integration results and derivatives.

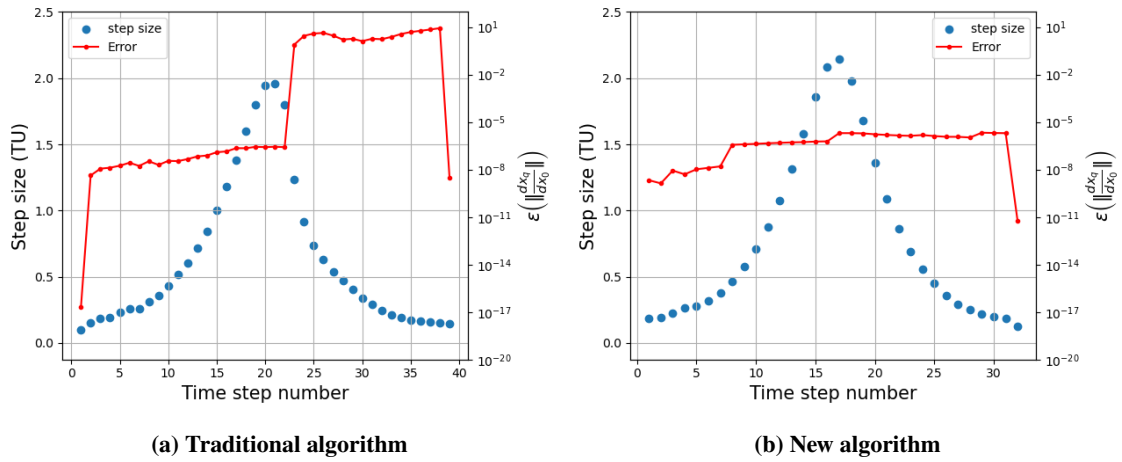


Fig. 4 Step size and derivative error at each time step.

Firstly, the change in derivatives is largest at the final step, and it dominates the overall accuracy of the computation. Secondly, as expected, the step size initially stays quite small and then increases as the spacecraft reaches apoapsis, before decreasing. And finally, the traditional approach sees a large spike in error when the step is decreased, that is when a step is rejected for being too large and is recomputed. This appears to confirm the hypothesis in II.B that inaccuracy in computing the derivatives stems from the implicit behavior that is not captured in explicit computation of

the derivatives. There is no similar spike for the new approach, showing that this issue has been successfully mitigated.

4. Computational Performance

Runtime tests, as seen in Fig 5, show that the traditional approach executes faster than either of the new approaches. This indicates that the new approaches trade runtime for derivative accuracy, which is as expected. We also note from Fig 6 that that new approach with an explicit tolerance takes fewer time steps than the traditional approach. The likely cause of this is the fact that the step size is selected to be the absolutely largest value allowed under the specified tolerance in the new approach while the traditional approach allows for a step to smaller than strictly necessary. Similarly, we note that the new approach with an implicit tolerance takes more steps than either of the other approaches. This approach is formulated such that it converges to a tighter tolerance at each step than the other approaches. Therefore, it follows that smaller step sizes will be required to converge successfully.

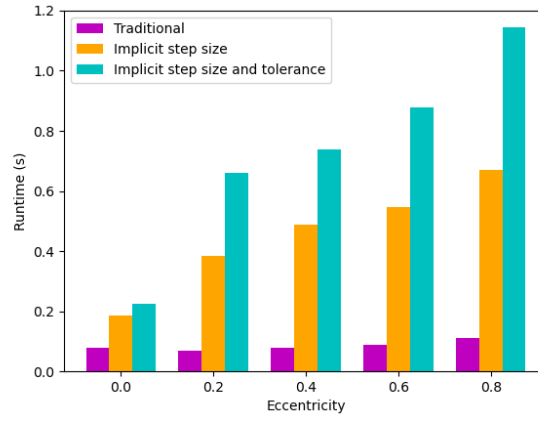


Fig. 5 Compute times.

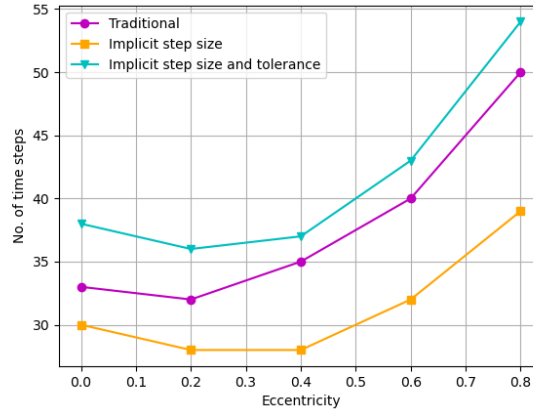


Fig. 6 Number of time steps.

Fig 7 shows the average execution time for a single time step. As before, the traditional approach is significantly faster than both the new approaches. The reasons for this are two-fold. Firstly, approximately half of the time steps in the traditional approach, where the step size is increasing, require no iteration and are purely explicit. This means that the ODE is only evaluated as many times as there are integration stages. In contrast, the new approaches iterate at every time step, thereby significantly increasing the number of time the ode is evaluated. The second factor driving the increased compute time is derivative computation. In the traditional approach, partial derivatives are computed

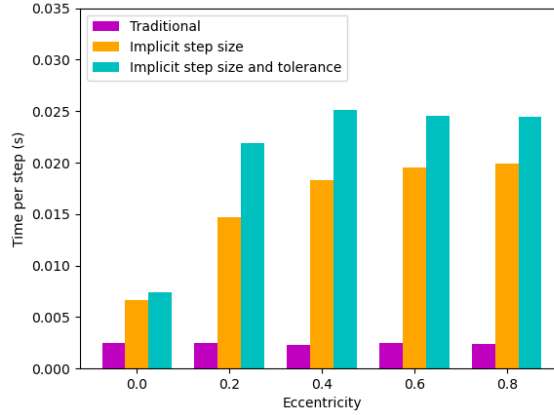


Fig. 7 Compute times at each time step.

by explicitly evaluating expressions. The new approaches obtain the partial derivatives by solving the linear system described in 53. Computing the partials in this fashion, while more expensive, is necessary to accurately evaluate partial derivatives across an implicit function. This is the driving factor behind the increase in derivative accuracy, so an increased computational cost may be accepted.

IV. Conclusions

In this work we’ve demonstrated a new method for integrating ODEs and accurately computing the derivatives of their outputs. We identified the issue in the traditional adaptive-step method of integrating ODEs that causes the derivatives computed from them to be inaccurate. Namely, the computation at each time can alternate between being explicit and implicit. A new approach that is implicit at each time step was proposed. This new approach eliminates the oscillation between explicit and implicit behavior that is observed in the traditional approach and as a result is able to compute much more accurate derivatives. Application of this algorithm in the modeling of sensitive dynamical systems should significantly improve the quality of derivatives. This in turn should enable more efficient trajectory optimization through use of algorithms like differential correction and gradient-based optimization.

V. Acknowledgements

This work was funded under NASA’s Transformational Tools and Technologies project. The authors would also like to thank Dr. Justin Gray for his input which helped to highlight common applications between the optimal control and multidisciplinary optimization communities.

References

- [1] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, Society for Industrial and Applied Mathematics, 2010.
- [2] Pellegrini, E., and Russell, R. P., “On the Computation and Accuracy of Trajectory State Transition Matrices,” *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 11, 2016, pp. 2485–2499. doi:10.2514/1.G001920, URL <https://doi.org/10.2514/1.G001920>.
- [3] Margolis, B. W. L., “A Sweeping Gradient Method for Ordinary Differential Equations with Events,” *Journal of Optimization Theory and Applications*, 2023.
- [4] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D., “Neural Ordinary Differential Equations,” 2018. doi: <https://doi.org/10.48550/arXiv.1806.07366>, URL [arXiv:1806.07366](https://arxiv.org/abs/1806.07366).
- [5] Gholami, A., Keutzer, K., and Biros, G., “ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs,” 2019. doi: <https://doi.org/10.48550/arXiv.1902.10298>, URL [arXiv:1902.10298](https://arxiv.org/abs/1902.10298).

- [6] Sundman, K. F., “Mémoire sur le problème des trois corps,” *Acta mathematica*, Vol. 36, No. 1, 1913, pp. 105–179.
- [7] Ellison, D. H., Englander, J. A., and Conway, B. A., “A Time-Regularized Multiple Gravity-Assist Low-Thrust Bounded-Impulse Model for Trajectory Optimization,” *27th AAS/AIAA Space Flight Mechanics Meeting, San Antonio, Texas*, 2017. AAS 17-429.
- [8] Fehlberg, E., *Classical Fifth-, Sixth-, Seventh-, and Eighth-order Runge-Kutta Formulas with Step-size Control*, NASA technical report, National Aeronautics and Space Administration, 1968. URL <https://books.google.com/books?id=5Uy6HLvsZRgC>.
- [9] Martins, J. R. R. A., and Hwang, J. T., “Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models,” *AIAA Journal*, Vol. 51, No. 11, 2013, pp. 2582–2599. doi:10.2514/1.J052184, URL <https://doi.org/10.2514/1.J052184>.
- [10] Fedorov, D. V., *Yet Another Introduction to Numerical Methods*, 2013.
- [11] Martins, J. R. R. A., Sturdza, P., and Alonso, J. J., “The Complex-Step Derivative Approximation,” *ACM Trans. Math. Softw.*, Vol. 29, No. 3, 2003, p. 245–262. doi:10.1145/838250.838251, URL <https://doi.org/10.1145/838250.838251>.