Self-scaling Collocation Methods Preserving Constants of Motion

Marco Sagliano*
German Aerospace Center, Bremen, 28359, Germany

Matthew Hölzel †
Teraki GmbH, Berlin, 10785, Germany

We present a class of self-scaling collocation integrators which, unlike explicit Runge-Kutta integrators, will automatically preserve quadratic constants of motion such as energy, linear momentum, and angular momentum, without the need for small stepsizes. However, collocation integrators in general require numerical optimization to solve an implicit set of equations which are difficult and computationally expensive to solve when the problem's Jacobian becomes ill-conditioned. Therefore, the integrators we present automatically scale the collocation conditions using a generalized eigenvalue problem (GEVP) approach, and exploit the Jacobian's structure to improve the optimization algorithm's precision and speed. Numerical results show that the presented integrators preserve invariants in a broader class of integration problems, with an improvement of approximately one order of magnitude over traditional Runge-Kutta 4(5) schemes.

I. Nomenclature

A	=	amplitude of oscillation
a	=	cylinder diameter
C_p	=	pressure coefficient
Cx	=	force coefficient in the x direction
Cy	=	force coefficient in the y direction
c	=	chord
dt	=	time step
Fx	=	X component of the resultant pressure force acting on the vehicle
Fy	=	Y component of the resultant pressure force acting on the vehicle
f, g	=	generic functions
h	=	height
i	=	time index during navigation
j	=	waypoint index
K	=	trailing-edge (TE) nondimensional angular deflection rate

II. Introduction

Despite being developed over 100 years ago, explicit Runge-Kutta methods are still among the most widely used numerical integration techniques [1–4], with many well-known integration schemes, such as Euler integration, falling into the broad class of Runge-Kutta integrators. Part of this success is due to the reasonably high orders of accuracy that simple Runge-Kutta methods can achieve. For example, the 1-stage Euler integration method is first-order accurate [5], the classic 4-stage Runge-Kutta method is 4^{th}-order accurate [5], and the 17-stage method presented in [6] is 10^{th}-order accurate. However, the number of stages required to achieve higher orders of accuracy increases rapidly, and since a general parameterization of such integrators is lacking, even calculating the parameters of higher order integrators becomes a difficult task [1]. In addition, explicit Runge-Kutta integration methods can become unstable for large

*Research Engineer, GNC Department, AIAA Senior Member
†Senior 3D Data Scientist.

integration timesteps, and since a proper timestep cannot be determined *a priori* without first analyzing the differential equation to be integrated, such methods are typically a poor choice for general-purpose numerical integration.

Conversely, variable timestep explicit Runge-Kutta integrators largely avoid instability and mimic the behavior of higher-order integrators by producing an estimate of the integration error, and decreasing the integration timestep if the estimated error is above a predefined threshold. For example, embedded Runge-Kutta pairs produce an estimate of the integration error by comparing the estimated solution of two different Runge-Kutta integrators with different orders of accuracy. However, although one can choose the tolerance of a variable timestep integrator to be arbitrarily small, the internal timestep required to achieve such a tolerance will also be small, and hence the method will require many function evaluations to integrate a differential equation. In addition, despite being able to choose arbitrarily small tolerances, explicit Runge-Kutta methods are fundamentally incapable of preserving quadratic constants of motion, such as energy, linear momentum, and angular momentum, and hence the errors in such quantities will grow at a rate proportional to the chosen tolerance [7].

Gauss collocation integrators [4], on the other hand, are inherently capable of preserving quadratic constants of motion, and can easily be chosen to have an arbitrarily high order of accuracy since the coefficients of the method are the roots of a polynomial. The disadvantage of these methods is, however, that they require a numerical optimization method to solve an implicit set of equations at each time step. Hence whereas variable timestep explicit Runge-Kutta methods can be used off-the-shelf, collocation methods require one to choose a numerical optimization method, where the accuracy of the collocation method is dependent on the accuracy of the optimization method. Furthermore, since optimization solvers are generally sensitive to scaling, one must ensure that the problem is properly scaled and conditioned before using a collocation integrator, which is a step that is not required by explicit Runge-Kutta methods.

In this paper, we present a generalized eigenvalue problem (GEVP) approach for automatically scaling the collocation conditions, thereby making their solution faster and more reliable [8]. Furthermore, we introduce a novel hybrid Jacobian computation approach which splits the computation into a component that can be computed *a priori*, and a contribution due to the differential equation that can be computed efficiently using a complex-step approximation [9]. The self-scaling collocation scheme that we propose is made significantly faster by using the hybrid Jacobian approach, while the automatic scaling improves the conditioning of the numerical problem to be solved, leading to more accurate results.

The paper is organized as follows: in Sec. III, explicit and implicit Runge-Kutta methods are introduced and compared. Section IV introduces the class of implicit Gauss collocation methods. In Sec. V the systematic hybridization of the Jacobian matrix, which exploits the structure of the collocation problem, is described in detail, while in Sec. VI the generalized eigenvalue problem (GEVP) scaling is applied to the collocation scheme. Results from attitude and orbital dynamics are shown in Sec. VII, while some final remarks are expressed in Sec. VIII.

III. Numerical Integration Techniques

In this paper, we investigate the performance of several Runge-Kutta integration methods for solving initial value problems of the form:

Problem III.1 Given the differential equation

$$\frac{dx}{dt}(t) = f\big(t, x(t)\big) \tag{1}$$

along with an initial condition $x(t_0) \in \mathbb{R}^n$ and a timestep h, compute an estimate $\hat{x}(t_0 + h)$ of $x(t_0 + h)$. □

where the family of Runge-Kutta methods is described in the following definition [1–4]:

Definition III.1 An *s-stage Runge-Kutta method* is a numerical integration method of the form

$$\begin{aligned} \hat{x}(t_0 + h) &= x(t_0) + h \sum_{j=1}^{s} b_j k_j \\ k_i &= f\left(t_0 + c_i h, x(t_0) + h \sum_{j=1}^{s} a_{i,j} k_j\right) \end{aligned} \tag{2}$$

or equivalently,

$$\begin{aligned}\hat{x}(t_0+h) &= x(t_0) + h\sum_{j=1}^{s} b_j f\left(t_0 + c_j h, X_j\right)\\ X_i &= x(t_0) + h\sum_{j=1}^{s} a_{i,j} f\left(t_0 + c_j h, X_j\right)\end{aligned} \tag{3}$$

where $a_{i,j}, b_j, c_i \in \mathbb{R}$ for $i, j \in [1, s]$, and the coefficients $a_{i,j}, b_j, c_i$ are independent of t_0, h, $x(t_0)$, and f. ☐

For example, Euler integration is a 1-stage Runge-Kutta method which produces an estimate $\hat{x}(t_0 + h)$ of the form [4]:

$$\hat{x}(t_0+h) = x(t_0) + hf\big(t_0, x(t_0)\big) \tag{4}$$

and the 4^{th}-order Runge-Kutta integrator [5] is a 4-stage integrator which produces an estimate of the form

$$\begin{aligned}\hat{x}(t_0+h) &= x(t_0) + \frac{hk_1}{6} + \frac{hk_2}{3} + \frac{hk_3}{3} + \frac{hk_4}{6}\\ k_1 &= f\left(t_0, x(t_0)\right)\\ k_2 &= f\left(t_0 + \frac{h}{2}, x(t_0) + \frac{hk_1}{2}\right)\\ k_3 &= f\left(t_0 + \frac{h}{2}, x(t_0) + \frac{hk_2}{2}\right)\\ k_4 &= f\left(t_0 + h, x(t_0) + hk_3\right)\end{aligned} \tag{5}$$

One of the distinguishing features of both Euler integration and the 4^{th}-order Runge-Kutta method is that they are *explicit* integration methods. In contrast, the implicit midpoint method [4] is an implicit Runge-Kutta method since the estimate $\hat{x}(t_0 + h)$ is the solution of the implicit equation

$$\hat{x}(t_0+h) = x(t_0) + hf\left(t_0 + \frac{h}{2}, \frac{\hat{x}(t_0+h)}{2} + \frac{x(t_0)}{2}\right) \tag{6}$$

Although implicit methods such as Eq. (6) will generally require numerical optimization to solve, and are therefore much more computationally expensive than explicit Runge-Kutta methods, as we will see, implicit Runge-Kutta methods are capable of preserving quadratic constants of motion and will generally have higher orders of accuracy than explicit methods with the same number of stages.

Order of Accuracy To classify the accuracy of an integration method, we compare the Taylor series of the true and estimated solution about the initial condition $x(t_0)$. Specifically, since the Taylor series of the true and estimated solution are given by

$$x(t_0+h) = x(t_0) + h\frac{dx}{dt}(t_0) + \frac{h^2}{2!}\frac{d^2x}{dt^2}(t_0) + \frac{h^3}{3!}\frac{d^3x}{dt^3}(t_0) + \cdots \tag{7}$$

$$\hat{x}(t_0+h) = x(t_0) + h\frac{d\hat{x}}{dt}(t_0) + \frac{h^2}{2!}\frac{d^2\hat{x}}{dt^2}(t_0) + \frac{h^3}{3!}\frac{d^3\hat{x}}{dt^3}(t_0) + \cdots \tag{8}$$

then the error $e(t_0 + h)$ in the estimate $\hat{x}(t_0 + h)$ is given by

$$\begin{aligned}e(t_0+h) = h\left[\frac{d\hat{x}}{dt}(t_0) - \frac{dx}{dt}(t_0)\right] + \frac{h^2}{2!}\left[\frac{d^2\hat{x}}{dt^2}(t_0) - \frac{d^2x}{dt^2}(t_0)\right] +\\ + \frac{h^3}{3!}\left[\frac{d^3\hat{x}}{dt^3}(t_0) - \frac{d^3x}{dt^3}(t_0)\right] + \cdots\end{aligned} \tag{9}$$

where the higher-order terms in h will have little effect on the error when h is small. Hence the goal of a numerical integration technique is to ensure that

$$\frac{dx}{dt}(t_0) = \frac{d\hat{x}}{dt}(t_0), \quad \cdots, \quad \frac{d^p x}{dt^p}(t_0) = \frac{d^p \hat{x}}{dt^p}(t_0)$$

for some reasonably high value of p. In fact, this is precisely how a method's *order of accuracy* is defined:

Definition III.2 A numerical integration method is p^{th}*-order accurate* if for all initial conditions $x(t_0)$, all timesteps h, and all differential equations of the form

$$\frac{dx}{dt}(t) = f\big(t, x(t)\big) \tag{10}$$

the first p derivatives of the estimate and true solution are equal, that is,

$$\frac{d^\ell x}{dt^\ell}(t_0) = \frac{d^\ell \hat{x}}{dt^\ell}(t_0), \quad \text{for all} \quad \ell = 0, \ldots, p \tag{11}$$

in which case, the error in the estimate $\hat{x}(t_0 + h)$ is on the order of h^{p+1}, that is,

$$\hat{x}(t_0 + h) - x(t_0 + h) = O\big(h^{p+1}\big) \tag{12}$$

□

Remark III.1 To compute the order of accuracy of a method, we implicitly assume that the Taylor series of the function f in Eq. (10) exists, and that it is locally convergent within some positive radius for every point in the domain of interest. These assumptions are common in the integration literature since one typically requires at least Lipschitz continuity to ensure that there exists a unique solution to the the differential Eq. (10) [7]. □

The Euler integration method is 1^{st}-order accurate, the 4-stage Runge-Kutta method is 4^{th}-order accurate, and the implicit midpoint method is 2^{nd}-order accurate. However, proving that a specific Runge-Kutta method is p^{th}-order accurate requires checking a very nonlinear set of equations that is beyond the scope of this paper [1]. Nonetheless, some of the necessary conditions are illustrative for understanding why implicit methods can have higher orders of accuracy than explicit methods. In this case, it is convenient to think of the coefficients $a_{i,j}, b_j, c_i$ that define a Runge-Kutta method as entries of the matrices $A \in \mathbb{R}^{s\times s}$, $B \in \mathbb{R}^{1\times s}$, and $C \in \mathbb{R}^{s\times 1}$, which we concisely write in a *Butcher tableau* [1]:

$$\left[\begin{array}{c|c} C & A \\ \hline & B \end{array}\right] = \left[\begin{array}{c|ccc} c_1 & a_{1,1} & \cdots & a_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & \cdots & a_{s,s} \\ \hline & b_1 & \cdots & b_s \end{array}\right] \tag{13}$$

For example, the Butcher tableau for Euler integration is given by

$$\left[\begin{array}{c|c} c_1 & a_{1,1} \\ \hline & b_1 \end{array}\right] = \left[\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}\right] \tag{14}$$

the Butcher tableau for the 4^{th}-order Runge-Kutta method is given by

$$\left[\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ c_2 & a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ c_3 & a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ c_4 & a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array}\right] = \left[\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}\right] \tag{15}$$

and the Butcher tableau for the implicit midpoint method is given by

$$\left[\begin{array}{c|c} c_1 & a_{1,1} \\ \hline & b_1 \end{array}\right] = \left[\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}\right] \tag{16}$$

As we can see from Eqs. (14)-(16), the defining feature of an explicit method is that the A-matrix must be strictly lower triangular, which reflects that the values k_i in Eq. (2) only depend on the values $k_1, \ldots, k_{i-1}$. Thus the following fact will allow us to bound the order of accuracy of an explicit Runge-Kutta method in terms of its number of stages:

Fact III.1 If (A, B, C) denote the Runge-Kutta matrices of a p^{th}-order accurate Runge-Kutta integration method, then for all $\ell = 1, \ldots, p$,

$$BA^{\ell-1}\mathbb{1}_s = \frac{1}{\ell!} \tag{17}$$

where $\mathbb{1}_s \in \mathbb{R}^{s\times 1}$ denotes a matrix of 1's.

Proof. Suppose that the differential equation in question is linear, that is,

$$\frac{dx}{dt}(t) = Gx(t) \tag{18}$$

where $x(t) \in \mathbb{R}^n$ an $G \in \mathbb{R}^{n\times n}$. Then the Taylor series of the true solution is given by

$$x(t_0 + h) = x(t_0) + hGx(t_0) + \frac{h^2}{2!}G^2x(t_0) + \frac{h^3}{3!}G^3x(t_0) + \cdots \tag{19}$$

Furthermore, using form Eq. (3) of the Runge-Kutta method, we find that

$$\hat{x}(t_0 + h) = x(t_0) + h\sum_{j=1}^{s} b_j G X_j \tag{20}$$

$$X_i = x(t_0) + h\sum_{j=1}^{s} a_{i,j} G X_j \tag{21}$$

and hence for $q > 0$,

$$\frac{d^q\hat{x}}{dt^q}(t_0 + h) = \sum_{j=1}^{s} b_j G\left[h\frac{d^qX_j}{dt^q} + q\frac{d^{q-1}X_j}{dt^{q-1}}\right] \tag{22}$$

$$\frac{d^qX_i}{dt^q} = \sum_{j=1}^{s} a_{i,j} G\left[h\frac{d^qX_j}{dt^q} + q\frac{d^{q-1}X_j}{dt^{q-1}}\right] \tag{23}$$

Therefore evaluating the derivatives at $h = 0$, we find that

$$\frac{d^q\hat{x}}{dt^q}(t_0) = q\sum_{j=1}^{s} b_j G\frac{d^{q-1}X_j}{dt^{q-1}} \tag{24}$$

$$\frac{d^qX_i}{dt^q} = q\sum_{j=1}^{s} a_{i,j} G\frac{d^{q-1}X_j}{dt^{q-1}} \tag{25}$$

Next, letting

$$X \triangleq \begin{bmatrix} X_1 \\ \vdots \\ X_s \end{bmatrix} \in \mathbb{R}^{ns\times 1} \tag{26}$$

then from Eqs. (24)-(25) it follows that

$$\frac{d^q\hat{x}}{dt^q}(t_0) = q\Big(B \otimes G\Big)\frac{d^{q-1}X}{dt^{q-1}} \tag{27}$$

$$\frac{d^qX}{dt^q} = q\Big(A \otimes G\Big)\frac{d^{q-1}X}{dt^{q-1}} \tag{28}$$

where $\otimes$ denotes the Kronecker product. Therefore

$$\frac{d^1\hat{x}}{dt^1}(t_0) = 1!\Big(B \otimes G\Big)X,$$
$$\frac{d^2\hat{x}}{dt^2}(t_0) = 2!\Big(BA \otimes G^2\Big)X,$$
$$\frac{d^3\hat{x}}{dt^3}(t_0) = 3!\Big(BA^2 \otimes G^3\Big)X$$

Furthermore, evaluating Eq. (20) at $h = 0$, we have that

$$X_i = x(t_0), \qquad \text{for all } i = 1, \ldots, s \tag{29}$$

and hence $X = \Big(\mathbb{1}_s \otimes I_n\Big)x(t_0)$. Thus

$$\frac{d^1\hat{x}}{dt^1}(t_0) = 1!\Big(B\mathbb{1}_s\Big)Gx(t_0), \tag{30}$$

$$\frac{d^2\hat{x}}{dt^2}(t_0) = 2!\Big(BA\mathbb{1}_s\Big)G^2x(t_0), \tag{31}$$

$$\frac{d^3\hat{x}}{dt^3}(t_0) = 3!\Big(BA^2\mathbb{1}_s\Big)G^3x(t_0) \tag{32}$$

Finally, from Eq. (32), we have that the Taylor series of the estimate is given by

$$\hat{x}(t_0 + h) = x(t_0) + h\Big(B\mathbb{1}_s\Big)Gx(t_0) + h^2\Big(BA\mathbb{1}_s\Big)G^2x(t_0) + \tag{33}$$

$$+ h^3\Big(BA^2\mathbb{1}_s\Big)G^3x(t_0) + \cdots \tag{34}$$

Therefore, comparing the Taylor series of the estimate (33) with the Taylor series of the exact solution (19), we find that the coefficients of a p^{th}-order accurate Runge-Kutta method must satisfy Eq. (17). ∎

The significance of Fact III.1 is encapsulated in the following corollary:

Corollary III.1 A p-stage explicit Runge-Kutta method is at most p^{th}-order accurate.

Proof. If a p-stage Runge-Kutta method is explicit, then for all $i \in [1, p]$, the values k_i in Eq. (2) must depend only on the values $k_1, \ldots, k_{i-1}$, meaning that the A-matrix of a p-stage explicit Runge-Kutta must be strictly lower triangular. Furthermore, since the A-matrix must be strictly lower triangular, it follows that $A^p = 0_{s\times s}$, and hence condition of Eq. (17) in Fact III.1 cannot hold for $\ell \geq p + 1$. ∎

The following example demonstrates the effect of the integration timestep and the order of accuracy on the integration error.

Example III.1 The position $r = \begin{bmatrix} x & y & z \end{bmatrix}^T \in \mathbb{R}^3$ of a satellite in orbit about the Earth can be modeled by the simplified differential equation

$$\frac{d^2r}{dt^2}(t) = -\frac{\mu_\oplus r(t)}{\big\|r(t)\big\|_2^3} \tag{35}$$

where $\mu_\oplus = 398,600\ \text{km}^3/\text{s}^2$ denotes Earth's gravitional parameter [10], and r denotes the position of the satellite expressed in an inertial frame located at the center of the Earth. Furthermore, if the orbit is circular, then the magnitude of the velocity vector is given by

$$\left\| \frac{dr}{dt}(t) \right\|_2 = \sqrt{\frac{\mu_\oplus}{\|r(t)\|_2}} \tag{36}$$

Hence if a satellite is in a 300 km circular orbit about the Earth at the initial position and velocity

$$r(t_0) = \begin{bmatrix} 6678\ \text{km} \\ 0 \\ 0 \end{bmatrix}, \qquad \frac{dr}{dt}(t_0) = \begin{bmatrix} 0 \\ 7.726\ \text{km/s} \\ 0 \end{bmatrix} \tag{37}$$

then the position and velocity at a general time t are given by

$$r(t) = 6678\ \text{km} \begin{bmatrix} \cos(\theta(t)) \\ \sin(\theta(t)) \\ 0 \end{bmatrix}, \qquad \frac{dr}{dt}(t) = 7.726\ \text{km/s} \begin{bmatrix} -\sin(\theta(t)) \\ +\cos(\theta(t)) \\ 0 \end{bmatrix} \tag{38}$$

$$\theta(t) = (t - t_0)\sqrt{\frac{\mu_\oplus}{\|r(t)\|_2^3}} = \frac{2\pi(t - t_0)}{5431\ \text{s}} \tag{39}$$

where the satellite's orbital period is 5431 s.

However, if we try to obtain the position r by integrating Eq. (35) over one orbital period using timesteps of $h = 100$ s with the 1^{st}-order accurate Euler integration method of Eq. (4), then Fig. 1 shows that the estimated x and y positions of the satellite diverge from the true circular orbit. Furthermore, although it appears that the 4^{th}-order Runge-Kutta in Eq. (5) and implicit midpoint in Eq. (6) methods do a good job of integrating the orbital equation, Fig. 2 shows that even those methods produce nontrivial integration errors which are inversely proportional to both the order of accuracy and the integration timestep.

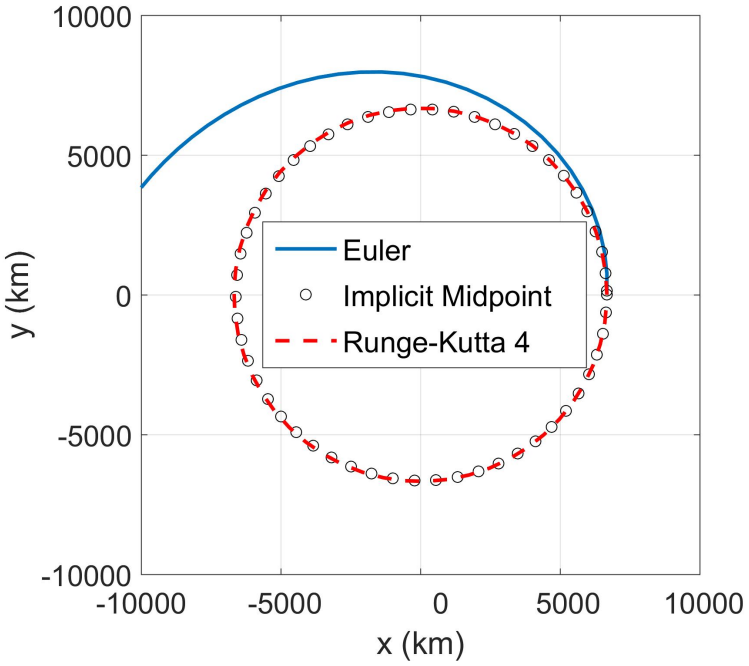


Fig. 1 Position of a satellite in a 300 km circular orbit over one orbital period using 100 second timesteps.

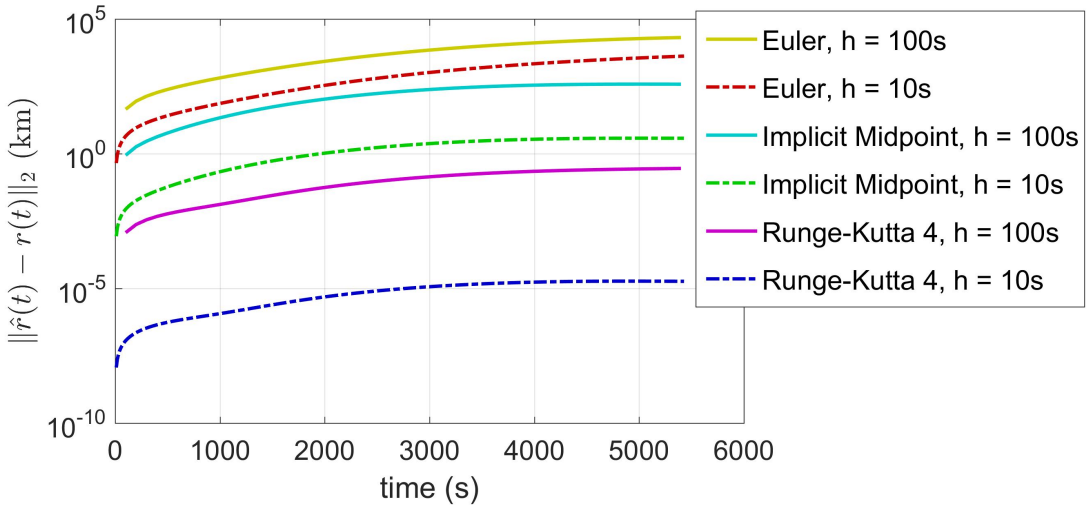


Fig. 2 Example of 2-norm of the error in the the integrated position of a satellite.

Remark III.2 Example III.1 demonstrates that the 4^{th}-order Runge-Kutta method is almost always preferably to both the Euler integration and implicit midpoint methods in terms of integration accuracy when using the same timestep for both methods. However, although the 4^{th}-order Runge-Kutta errors in Fig. 2 might seem reasonable even for the large timestep of $h = 100$ s, it is important to remember that:

1) What constitutes an "acceptable" error is extremely dependent on the intended use of the integrated variables.
2) The timesteps $h = 10$ and 100 s were arbitrarily chosen. If we were simulating many orbits of a satellite, then it might seem reasonable to try an integration timestep of 1 orbital period (5431 s), in which case, both methods would produce unacceptably large errors.

The danger in using fixed timestep Runge-Kutta integrators is that they are typically only needed in cases where we cannot compute the exact integration error, which means that we will often have no way of knowing whether the integration solution is accurate in such cases. For example, if we try to integrate the dynamics of a solar sail in a 300 km orbit about the Earth, then it will be difficult to say *a priori* whether the sail should deorbit due to aerodynamic drag, or actually gain altitude by harnessing solar radiation pressure. Hence unlike Example III.1, where we used the knowledge that the orbit was circular to conclude that Euler integration produced a bad estimate in Fig. 1, we would have no rationale for thinking that a solar sail simulation was correct or incorrect. This is precisely the problem that variable timestep integrators address.

Variable Timestep Integrators Variable timestep numerical integrators improve upon fixed timestep integrators, such as the Euler of Eq. (4) and 4^{th}-order Runge-Kutta of Eq. (5) methods, by producing an estimate of the integration error, and decreasing the integration timestep if the estimated error is above a predefined threshold. For example, embedded Runge-Kutta pairs produce an estimate of the integration error by comparing the estimated solutions of two different Runge-Kutta integrators with the same A and C-matrices, but different B-matrices and different orders of accuracy, p_{low} and p_{high}. For instance, one of the most famous embedded Runge-Kutta pairs is the Dormand-Prince

method, which has the Butcher tableau

$$\left[\begin{array}{c|c} C & A \\ \hline & \begin{matrix} B_{\text{high}} \\ B_{\text{low}} \end{matrix} \end{array}\right] =$$

$$= \left[\begin{array}{c|cccccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{10} & \frac{3}{40} & \frac{9}{40} & 0 & 0 & 0 & 0 & 0 \\ \frac{4}{5} & \frac{44}{45} & -\frac{56}{15} & \frac{32}{9} & 0 & 0 & 0 & 0 \\ \frac{8}{9} & \frac{19372}{6561} & -\frac{25360}{2187} & \frac{64448}{6561} & -\frac{212}{729} & 0 & 0 & 0 \\ 1 & \frac{9017}{3168} & -\frac{355}{33} & \frac{46732}{5247} & \frac{49}{176} & -\frac{5103}{18656} & 0 & 0 \\ 1 & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\ \hline & \frac{35}{384} & 0 & \frac{500}{1113} & \frac{125}{192} & -\frac{2187}{6784} & \frac{11}{84} & 0 \\ & \frac{5179}{57600} & 0 & \frac{7571}{16695} & \frac{393}{640} & -\frac{92097}{339200} & \frac{187}{2100} & \frac{1}{40} \end{array}\right] \tag{40}$$

where the integrators (A, B_{low}, C) and (A, B_{high}, C) are $p_{\text{low}} = 4^{th}$ and $p_{\text{high}} = 5^{th}$ order accurate, respectively [2].

The usefulness of having estimates $\hat{x}_{\text{low}}$ and $\hat{x}_{\text{high}}$ with different orders of accuracy is due to the fact that their difference is approximately equal to the integration error in the lower-order estimate, that is,

$$\begin{aligned} \hat{x}_{p_{\text{low}}}(t_0 + h) - \hat{x}_{p_{\text{high}}}(t_0 + h) &= \left[\hat{x}_{p_{\text{low}}}(t_0 + h) - x(t_0 + h)\right] + \\ &\quad - \left[\hat{x}_{p_{\text{high}}}(t_0 + h) - x(t_0 + h)\right] \\ &= O\left(h^{p_{\text{low}}+1}\right) + O\left(h^{p_{\text{high}}+1}\right) = O\left(h^{p_{\text{low}}+1}\right) \end{aligned} \tag{41}$$

Hence using the difference

$$\varepsilon \triangleq \hat{x}_{p_{\text{low}}}(t_0 + h) - \hat{x}_{p_{\text{high}}}(t_0 + h) \tag{42}$$

as an estimate of the integration error in the lower-order accurate method, we find that ε is approximately of the form $\varepsilon \approx \alpha h^{p_{\text{low}}+1}$. Therefore given the estimated integration error ε after one integration timestep h, it follows that $\alpha \approx \dfrac{\varepsilon}{h^{p_{\text{low}}+1}}$, where the timestep h_δ that would have been required to produce an error of $\|\varepsilon\| = \delta$ is approximately given by

$$h_\delta \approx h \left(\frac{\delta}{\|\varepsilon\|}\right)^{1/(p_{\text{low}}+1)} \tag{43}$$

where the norm $\|\varepsilon\|$ is typically taken to be the infinity norm. Thus if the norm of the estimated integration error ε is larger than some predefined tolerance δ, then we should reject the estimated solution $\hat{x}_{\text{low}}(t_0 + h)$ and recompute the Runge-Kutta estimate using a timestep less than or equal to the one computed by using Eq. (43).

Remark III.3 There are numerous additional issues to be addressed to make the implementation of a variable timestep integrator with the update in Eq. (43) efficient. However, to avoid a complete and detailed explanation of the programming strategies involved, we use Matlab's `ode45` function in the following examples to show the *typical* performance of such an implementation [11]. Note, however, that Matlab's implementation is far from perfect, and in fact, although the `ode45` function allows one to set both the absolute and relative integration error tolerances, one cannot completely decouple these tolerances in their implementation. Hence we always set the absolute and relative tolerances to be equal. ❐

Example III.2 Consider again a satellite in a 300 km circular Earth orbit, as in Example III.1. Figure 3 shows the integration error in the position r of the satellite during one orbital period when using Matlab's implementation of the Dormand-Prince method (`ode45`) along with the integration tolerances `AbsTol = RelTol` $= \delta = 10^{-15}, 10^{-10}, 10^{-5}$, and 1. From Fig. 3, we see that the integrated errors are indeed proportional to the chosen tolerances, with no need to

choose the sample timestep *a priori*. However, since the tolerances are also proportional to the required timestep (see Eq. (43)), the timesteps required to achieve these tolerances vary widely. For instance, the median timesteps used by `ode45` to achieve the tolerances `AbsTol = RelTol` $= \delta = 10^{-15}, 10^{-10}, 10^{-5}$, and 1 are h =1.1 s, 6.0 s, 60 s, and 140 s, respectively.

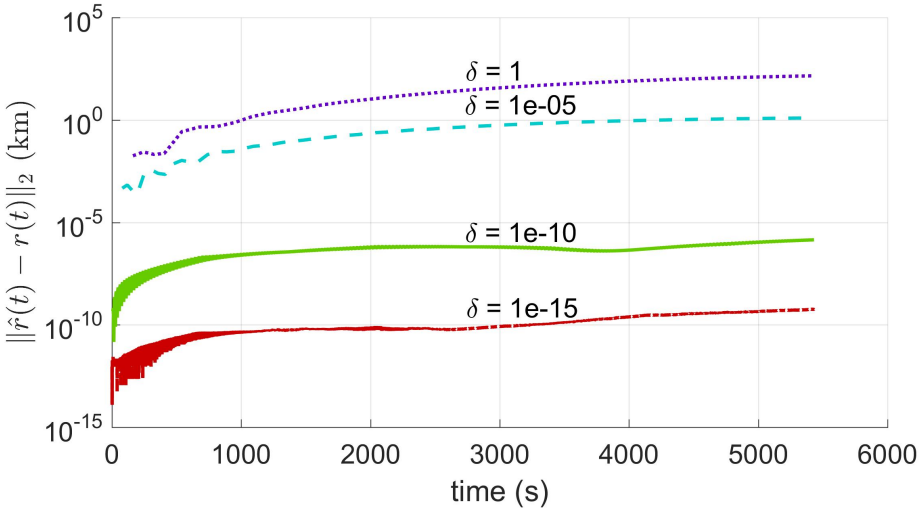


Fig. 3 Example of 2-norm of the error in the the integrated position of a satellite when using Matlab's `ode45`.

Runge-Kutta Methods which Preserve Quadratic Constants of Motion A quadratic constant of motion (or quadratic *invariant*) of the differential equation

$$\frac{dx}{dt}(t) = f\big(t, x(t)\big) \tag{44}$$

is a quadratic equation that a solution $x(t) \in \mathbb{R}^n$ of Eq. (44) satisfies for all times t, that is,

$$x^T(t)Qx(t) = \gamma, \qquad \text{for all } t \tag{45}$$

where $Q \in \mathbb{R}^{n\times n}$ is symmetric and $\gamma \in \mathbb{R}$. The following fact, along with a complete proof of [12], gives a sufficient condition for Runge-Kutta methods that preserve quadratic invariants:

Fact III.2 Let (A, B, C) denote the matrices of an s-stage Runge-Kutta method. If

$$B^T B = \mathtt{diag}(B)A + A^T \mathtt{diag}(B) \tag{46}$$

then for all time-invariant differential equations of the form

$$\frac{dx}{dt}(t) = f\big(x(t)\big) \tag{47}$$

the Runge-Kutta method will preserve quadratic invariants of the form of Eq. (45), that is, for all timesteps h, we will find that

$$\hat{x}^T(t_0 + h)Q\hat{x}(t_0 + h) = x^T(t_0)Qx(t_0) \tag{48}$$

Proof. From the equivalent form in Eq. (3) of a Runge-Kutta method, we have that

$$\hat{x}(t_0 + h) = x(t_0) + \Big(B \otimes I_n\Big)F \tag{49}$$

$$X = \Big(\mathbb{1}_s \otimes I_n\Big)x(t_0) + \Big(A \otimes I_n\Big)F \tag{50}$$

where

$$F \triangleq h \begin{bmatrix} f(X_1) \\ \vdots \\ f(X_s) \end{bmatrix}, \qquad X \triangleq \begin{bmatrix} X_1 \\ \vdots \\ X_s \end{bmatrix} \tag{51}$$

Furthermore, since Q is symmetric, then from Eq. (49), we find that

$$\hat{x}^T(t_0+h)Q\hat{x}(t_0+h) = x^T(t_0)Qx(t_0) + 2F^T\Big(B^T \otimes Q\Big)x(t_0) + F^T\Big(B^T B \otimes Q\Big)F \tag{52}$$

Next, pre-multiplying Eq. (50) by $F^T\Big(\mathtt{diag}(B) \otimes Q\Big)$, we find that

$$F^T\Big(\mathtt{diag}(B) \otimes Q\Big)X = F^T\Big(B^T \otimes Q\Big)x(t_0) + F^T\Big(\mathtt{diag}(B)A \otimes Q\Big)F \tag{53}$$

where differentiating the quadratic invariant in Eq. (45), it follows that

$$\left[\frac{dx}{dt}(t)\right]^T Qx(t) + x^T(t)Q\frac{dx}{dt}(t) = 2\left[\frac{dx}{dt}(t)\right]^T Qx(t) = 2f^T(x(t))Qx(t) = 0 \tag{54}$$

for all t, and hence

$$F^T\Big(\mathtt{diag}(B) \otimes Q\Big)X = \sum_{i=1}^{s} b_i f^T(X_1)\Big[Q + Q^T\Big]X_i = 0 \tag{55}$$

Furthermore, combining Eqs. (52), (53), and (55), we find that

$$\hat{x}^T(t_0+h)Q\hat{x}(t_0+h) = x^T(t_0)Qx(t_0) + F^T\Big(\Big[B^T B - 2\mathtt{diag}(B)A\Big] \otimes Q\Big)F \tag{56}$$

Finally, since

$$F^T\Big(\mathtt{diag}(B)A \otimes Q\Big)F = F^T\Big(A^T\mathtt{diag}(B) \otimes Q\Big)F \tag{57}$$

it follows that if there exists a $\theta \in \mathbb{R}$ for which

$$B^T B = (2-\theta)\mathtt{diag}(B)A + \theta A^T\mathtt{diag}(B) \tag{58}$$

then Eq. (48) holds. However, since $B^T B$ is symmetric, we must have that $(2-\theta)\mathtt{diag}(B)A + \theta A^T\mathtt{diag}(B)$ is symmetric, and therefore either $\mathtt{diag}(B)A$ is symmetric, or $\theta = 1$. Therefore, if Eq. (58) holds for some θ, then Eq. (46) must also hold. ❐

Remark III.4 Aside from degenerate Runge-Kutta methods where $B = 0$ (and hence $\hat{x}(t_0+h) = x(t_0)$), the condition in Eq. (46) in Fact III.2 can only be satisfied by implicit Runge-Kutta methods since the A-matrices of explicit Runge-methods will have zeros on the diagonal, whereas the product $B^T B$ will have non-zero diagonal entries. To show that Fact III.2 is also necessary condition, one would also have to show that if

$$F^T\Big(\Big[B^T B - 2\mathtt{diag}(B)A\Big] \otimes Q\Big)F = 0 \tag{59}$$

then Eq. (46) holds, although there are clearly pathological cases where this is not true, such as when $f(x) = 0$. Hence although we have not ruled out the possibility of developing an explicit Runge-Kutta method which preserves quadratic invariants for specific differential equations, it will not be possible to develop explicit Runge-Kutta methods which preserve quadratic invariants in the general case. ❐

Example III.3 The implicit midpoint rule will preserve quadratic invariants since it satisfies Fact III.2, that is,

$$1 = B^T B = \mathtt{diag}(B)A + A^T\mathtt{diag}(B) = \frac{1}{2} + \frac{1}{2} = 1 \tag{60}$$

However, neither the Euler in Eq. (4) nor 4^{th}-order Runge-Kutta method satisfy Fact III.2. ❐

IV. Collocation Methods

Definition IV.1 An *s-stage collocation method* produces an estimate

$$\hat{x}(t) = x(t_0) + (t - t_0)\hat{x}_1 + \cdots + (t - t_0)^s \hat{x}_s \tag{61}$$

of the solution $x(t)$ of the differential equation

$$\frac{dx}{dt}(t) = f\big(t, x(t)\big) \tag{62}$$

by determining the constant vectors $\hat{x}_1, \ldots, \hat{x}_s \in \mathbb{R}^n$ such that

$$\frac{d\hat{x}}{dt}(t_0 + c_i h) = f\Big(t_0 + c_i h, \hat{x}(t_0 + c_i h)\Big) \tag{63}$$

for all of the distinct points $c_1, \ldots, c_s \in \mathbb{R}$, where $c_1, \ldots, c_s \in \mathbb{R}$ are called the *collocation points*. □

Fact IV.1 An s-stage collocation method with the collocation points $c_1, \ldots, c_s$ produces the same estimate $\hat{x}(t_0 + h)$ as an s-stage Runge-Kutta method with the Butcher tableau

$$\left[\begin{array}{c|c} C & A \\ \hline & B \end{array}\right] = \left[\begin{array}{c|c} \begin{matrix} c_1 \\ \vdots \\ c_s \end{matrix} & V_s \tilde{V}_s^{-1} \\ \hline & \mathbb{1}_s^T \tilde{V}_s^{-1} \end{array}\right] \tag{64}$$

where

$$V_s \triangleq \begin{bmatrix} c_1 & c_1^2 & \cdots & c_1^s \\ \vdots & \vdots & & \vdots \\ c_s & c_s^2 & \cdots & c_s^s \end{bmatrix}, \qquad \tilde{V}_s \triangleq \begin{bmatrix} 1 & 2c_1 & \cdots & sc_1^{s-1} \\ \vdots & \vdots & & \\ 1 & 2c_s & \cdots & sc_s^{s-1} \end{bmatrix} \tag{65}$$

Proof. First, note that at the collocation points, the collocation solution in Eq.(61) is of the form

$$\hat{x}(t_0 + c_i h) = x(t_0) + c_i h \hat{x}_1 + \cdots (c_i h)^s \hat{x}_s \tag{66}$$

$$\frac{d\hat{x}}{dt}(t_0 + c_i h) = \hat{x}_1 + 2c_i h \hat{x}_2 + \cdots + s(c_i h)^{s-1} \hat{x}_s \tag{67}$$

Hence letting $k_i \triangleq \dfrac{d\hat{x}}{dt}(t_0 + c_i h)$, we have that

$$\hat{x}(t_0 + h) = x(t_0) + \begin{bmatrix} \hat{x}_1 & \cdots & \hat{x}_s \end{bmatrix} \mathtt{diag}\Big(h, h^2, \ldots, h^s\Big)\mathbb{1}_s \tag{68}$$

$$\begin{bmatrix} \hat{x}(t_0 + c_1 h) & \cdots & \hat{x}(t_0 + c_s h) \end{bmatrix} = \Big(\mathbb{1}_{1\times s} \otimes x(t_0)\Big) + \\ + \begin{bmatrix} \hat{x}_1 & \cdots & \hat{x}_s \end{bmatrix} \mathtt{diag}\Big(h, h^2, \ldots, h^s\Big) V_s^T \tag{69}$$

$$\begin{bmatrix} k_1 & \cdots & k_s \end{bmatrix} = \begin{bmatrix} \hat{x}_1 & \cdots & \hat{x}_s \end{bmatrix} \mathtt{diag}\Big(1, h, \ldots, h^{s-1}\Big) \tilde{V}_s^T \tag{70}$$

and therefore

$$\hat{x}(t_0 + h) = x(t_0) + h \begin{bmatrix} k_1 & \cdots & k_s \end{bmatrix} \Big(\mathbb{1}_s^T \tilde{V}_s^{-1}\Big)^T \tag{71}$$

$$\begin{bmatrix} \hat{x}(t_0 + c_1 h) & \cdots & \hat{x}(t_0 + c_s h) \end{bmatrix} = \Big(\mathbb{1}_{1\times s} \otimes x(t_0)\Big) + \\ + h \begin{bmatrix} k_1 & \cdots & k_s \end{bmatrix} \Big(V_s \tilde{V}_s^{-1}\Big)^T \tag{72}$$

where the matrix $\tilde{V}_s$ is invertible since it is the product of the full rank matrix $\texttt{diag}(1, \ldots, s)$ and a Vandermonde matrix constructed from $c_1, \ldots, c_s$, which has full rank since the collocation points are distinct [13].

Finally, since the collocation solution of Eq. (61) satisfies Eq. (63) at each of the collocation points, then from Eq. (72) it follows that

$$k_i = f\left(t_0 + c_i h, x(t_0) + h \sum_{j=1}^{s} \left(V_s \tilde{V}_s^{-1}\right)_{i,j} k_j\right), \qquad \text{for all } i = 1, \ldots, s \tag{73}$$

where $(\cdot)_{i,j}$ denotes the (i, j) entry of $(\cdot)$. Hence, combined with Eq. (71), that is,

$$\hat{x}(t_0 + h) = x(t_0) + h \sum_{j=1}^{s} \left(\mathbb{1}_s^T \tilde{V}_s^{-1}\right)_j k_j \tag{74}$$

we see that the collocation method of Eqs. (73)-(74) is equivalent to a Runge-Kutta method of Eq. (2) with the Butcher tableau in Eq. (64). □

In this paper, we are primarily concerned with Gauss collocation methods:

Definition IV.2 An *s-stage Gauss collocation method* is a collocation method where the collocation points are the zeros of the polynomial

$$\frac{d^s}{dt^s}\left[t^s(t-1)^s\right] \tag{75}$$

□

For example, the Butcher tableaus of 1 and 2-stage Gauss collocation methods are shown in Table 1, where comparing Table 1 with the Butcher tableau given in Eq. (16), we see that the 1-stage Gauss collocation method is equivalent to the implicit midpoint method. Furthermore, Gauss collocation methods are interesting in the context of this paper due to

Stages	1	2
$\left[\begin{array}{c\|c} C & A \\ \hline & B \end{array}\right]$	$\left[\begin{array}{c\|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}\right]$	$\left[\begin{array}{c\|cc} \frac{3-\sqrt{3}}{6} & \frac{1}{4} & \frac{3-2\sqrt{3}}{12} \\ \frac{3+\sqrt{3}}{6} & \frac{3+2\sqrt{3}}{12} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}\right]$

Table 1 Butcher tableaus of 1 and 2-stage Gauss collocation methods.

the following fact:

Fact IV.2 Gauss collocation methods preserve quadratic invariants.

Proof. A formal proof of Fact IV.2 is showed in [4]. □

V. Hybrid Jacobian Computation

To compute an s-stage collocation estimate, one must solve the implicit set of Eq. (63) for the constant vectors $\hat{x}_1, \ldots, \hat{x}_s \in \mathbb{R}^n$ at all of the collocation points $c_1, \ldots, c_s$. Specifically, since the collocation estimate is of the form

$$\hat{x}(t) = x(t_0) + (t - t_0)\hat{x}_1 + \cdots + (t - t_0)^s \hat{x}_s \tag{76}$$

$$\frac{d\hat{x}}{dt}(t) = \hat{x}_1 + 2(t - t_0)\hat{x}_1 + \cdots + s(t - t_0)^{s-1}\hat{x}_s \tag{77}$$

then like in Eqs. (69)-(70), we find that at the collocation points, the estimate and its derivative are of the form

$$\begin{bmatrix} \hat{x}(t_0 + c_1 h) & \cdots & \hat{x}(t_0 + c_s h) \end{bmatrix} = \Big(\mathbb{1}_{1\times s} \otimes x(t_0)\Big) + \begin{bmatrix} \hat{x}_1 & \cdots & \hat{x}_s \end{bmatrix} W_s \tag{78}$$

$$\begin{bmatrix} \frac{d\hat{x}}{dt}(t_0 + c_1 h) & \cdots & \frac{d\hat{x}}{dt}(t_0 + c_1 h) \end{bmatrix} = \begin{bmatrix} \hat{x}_1 & \cdots & \hat{x}_s \end{bmatrix} \tilde{W}_s \tag{79}$$

$$W_s \triangleq V_s \texttt{diag}\Big(h, h^2, \ldots, h^s\Big) \tag{80}$$

$$\tilde{W}_s \triangleq \tilde{V}_s \texttt{diag}\Big(1, h, \ldots, h^{s-1}\Big) \tag{81}$$

where V_s and $\tilde{V}_s$ are given by Eq. (65). Hence to compute the collocation estimate, one must solve the implicit system of constraints

$$F\big(\hat{x}_1, \ldots, \hat{x}_s\big) \triangleq \begin{bmatrix} \frac{d\hat{x}}{dt}(t_0 + c_1 h) - f\Big(t_0 + c_1 h, \hat{x}(t_0 + c_1 h)\Big) \\ \vdots \\ \frac{d\hat{x}}{dt}(t_0 + c_s h) - f\Big(t_0 + c_s h, \hat{x}(t_0 + c_s h)\Big) \end{bmatrix} = 0_{sn\times 1} \tag{82}$$

or equivalently,

$$F\big(\hat{X}\big) \triangleq = \big(\tilde{W}_s \otimes I_n\big)\, \hat{X} - \begin{bmatrix} f\Big(t_0 + c_1 h, x(t_0) + \big(e_1^T W_s^T \otimes I_n\big)\, \hat{X}\Big) \\ \vdots \\ f\Big(t_0 + c_s h, x(t_0) + \big(e_s^T W_s^T \otimes I_n\big)\, \hat{X}\Big) \end{bmatrix} = 0_{sn\times 1} \tag{83}$$

where $\hat{X} \triangleq \begin{bmatrix} \hat{x}_1^T & \cdots & \hat{x}_s^T \end{bmatrix}^T \in \mathbb{R}^{ns\times 1}$, and e_i is an $s \times 1$ vector whose only nonzero element is the i^{th} entry, which is 1. Furthermore, to increase optimization algorithm's precision and speed, we will use the Jacobian of the constraints in Eq. (82), which we propose splitting into the sum of an analytical and numerical component, that is,

$$J(\hat{X}) \triangleq \frac{\partial F}{\partial \hat{X}}(\hat{X}) = J_A + J_N(\hat{X}) \tag{84}$$

where the analytical part J_A can be computed exactly *a priori*, and the numerical part J_N is determined by the differential equation being integrated.

Analytical Jacobian The analytic component J_A of the Jacobian matrix defined by Eq. (84) is the contribution due to the derivatives $\frac{d\hat{x}}{dt}(t_0 + c_1 h), \ldots, \frac{d\hat{x}}{dt}(t_0 + c_s h)$ given in Eq. (82), which is exactly given by

$$J_A = \tilde{W}_s \otimes I_n \tag{85}$$

Since this contribution to the Jacobian depends on the timestep and collocation points, but not the differential equation, it only needs to be computed once at the beginning of the optimization, thereby reducing the time required to compute the Jacobian and solve the implicit system of collocation constraints of Eq. (82).

Numerical Jacobian The numerical component $J_N(\hat{X})$ of the Jacobian matrix in Eq. (84) is due to the derivatives of the differential equation terms $f\big(t_0 + c_1 h, \hat{x}(t_0 + c_1 h)\big), \ldots, f\big(t_0 + c_s h, \hat{x}(t_0 + c_s h)\big)$ in Eq. (82) with respect to $\hat{X}$. Since these terms are problem-dependent and could be highly nonlinear, it will not always be possible to calculate their symbolic derivatives in a reasonable amount of time. Instead, we use complex-step differentiation [9, 14, 15] to numerically compute their derivatives with respect to $\hat{X}$, since unlike traditional finite-differences schemes, complex-step differentiation does not suffer from roundoff errors.

Example V.1 The complex-step estimate of the derivative of a scalar function g is given by

$$\dot{g}_{cs}(y) = \frac{\texttt{imag}\Big(g(y + ih)\Big)}{h} \tag{86}$$

which is 2^{nd}-order accurate when g is analytic [16], that is,

$$\frac{dg}{dy}(y) = \dot{g}_{cs}(y) + O\left(h^2\right) \tag{87}$$

Similarly, the 3, 5, and 7-point central difference approximations given by

$$\dot{g}_3 = \frac{g\,(y+h) - g\,(y-h)}{2h} \tag{88}$$

$$\dot{g}_5 = \frac{-g\,(y+2h) + 8g\,(y+h) - 8g\,(y-h) + g\,(y-2h)}{12h} \tag{89}$$

$$\dot{g}_7 = \frac{g\,(y+3h) - 9g\,(y+2h) + 45g\,(y+h) - 45g\,(y-h) + 9g\,(y-2h) - g\,(y-3h)}{60h} \tag{90}$$

are 2^{nd}, 4^{th}, and 6^{th}-order accurate, respectively. The absolute errors in the complex-step and 3, 5, and 7-point central difference approximations of the Jacobian of the function

$$g(y) = \sin(y)e^{-y^2} \tag{91}$$

about the point $y = 1$ are shown in Fig. 4 for various stepsizes h. Figure 4 shows that although the higher-order central difference methods are more accurate than the complex-step derivative approximation for small stepsizes, the complex-step approximation is not affected by roundoff, and hence yields derivative approximations that are on the order of machine precision for small stepsizes [9]. Conversely, for small stepsizes, Fig. 4 shows that the traditional central difference approximations are dominated by roundoff, and hence drastically underperform the complex-step approximation. □

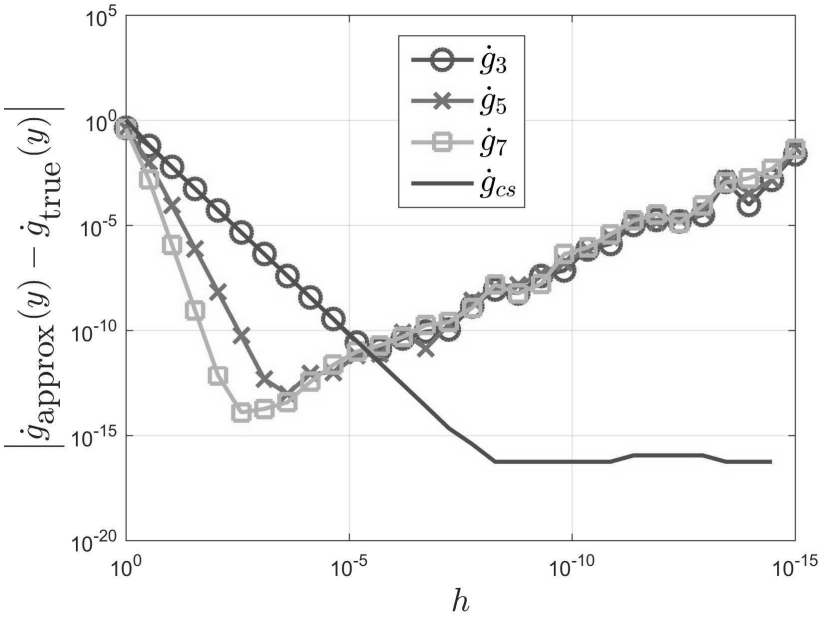


Fig. 4 Absolute errors in the computation of first derivative of $g(y)$ for various stepsizes h.

The k^{th} column $J_{N,k}(\hat{X})$ of the numerical Jacobian $J_N(\hat{X})$ is therefore given by

$$J_{N,k}\left(\hat{X}\right) = = -\frac{1}{h}\texttt{imag}\begin{bmatrix} f\left(t_0 + c_1 h, x(t_0) + \left(e_1^T W_s^T \otimes I_n\right)\left(\hat{X} + ihe_k\right)\right) \\ \vdots \\ f\left(t_0 + c_s h, x(t_0) + \left(e_s^T W_s^T \otimes I_n\right)\left(\hat{X} + ihe_k\right)\right) \end{bmatrix} \tag{92}$$

where we use $h = 10^{-12}$ for the remainder of this paper.

Remark V.1 Although the complex-step approximation could be applied to the entire Jacobian in Eq. (84), we compute the terms separately since the analytical part only needs be computed once *a priori*, thereby improving the computational efficiency of the calculation. ❐

Remark V.2 The Jacobian can also be approximated using dual-numbers [17, 18], although this yields a heavier computational load than the complex-step approach, with negligible changes in the results. Alternatively, tools for symbolic differentiation may be employed, but most of the time they require third-party tools, while complex-step can easily be coded with any programming language. ❐

VI. Self-scaling: GEVP method

Obtaining fast, precise, and reliable results from numerical optimization is often dependent on having a well-scaled, and therefore well-conditioned, Jacobian matrix [15, 19]. One traditional approach to scaling has been to scale the variables of the problem manually to either the range $[0,\ 1]$ or $[-1,\ 1]$ by using knowledge about the magnitudes of the variables that one expects. For instance, in atmospheric reentry problems, the radial position of the reentry vehicle is typically divided by the radius of Earth, resulting in a radial position that is very close to 1 [20].

However, although manual scaling can be effective, it is time-consuming, problem-dependent, and in some cases, must be changed dramatically for the same problem with different parameters or initial conditions. For example, the scaling needed for an orbital-dynamics problem around the Moon is very different from the scaling needed for the same problem around the Sun, and the scaling needed at apogee might be very different from at perigee. Furthermore, since one only knows *a posteriori* whether the scaling was useful, choosing a good range for variable scaling is a matter of experience and luck.

Over the years, several automatic scaling techniques were proposed to address these problems [15, 19, 21], although they all still require at least a rough estimate of the upper and lower bounds of the variables involved in the problem. Furthermore, although one could simply guess conservative bounds when no *a priori* estimate can be made, the scaling process will suffer as a result. Hence we implement a generalized eigenvalue problem (GEVP) based scaling technique to automatically determine the function and variable scalings [8]. This method, which has already been successfully used for Lattice Quantum Chromodynamics problems [22, 23], does not require an *a priori* estimate of the variable bounds, and can be solved efficiently with many off-the-shelf packages such as YALMIP [24] and SeDuMi [25].

The GEVP-based automatic scaling method that we propose is based on premultiplying the problem constraints F and optimization parameters $\hat{X}$ in Eq. (83) with nonsingular diagonal matrices K_p and K_q, respectively, before solving the modified problem

$$K_p F(Y) = 0, \qquad Y = K_q \hat{X} \tag{93}$$

for Y. Then, once a solution Y is obtained via numerical optimization, the parameters $\hat{X}$ are found to be given by $\hat{X} = K_q^{-1} Y$. Specifically, we propose using the scaling matrices that minimize the condition number of the Jacobian in Eq. (84) evaluated at the initial condition, that is, the scaling matrices which solve the scaling problem:

Problem VI.1 Given a Jacobian matrix $J \in \mathbb{R}^{n_p \times n_q}$, find two diagonal and nonsingular matrices $K_p \in \mathbb{R}^{n_p \times n_p}$ and $K_q \in \mathbb{R}^{n_q \times n_q}$ which minimize the condition number of $K_p J K_q$, that is,

$$\underset{K_p, K_q}{\text{minimize}} \quad \frac{\overline{\sigma}\left(K_p J K_q\right)}{\underline{\sigma}\left(K_p J K_q\right)} \quad \text{such that } K_p \text{ and } K_q \text{ are diagonal and nonsingular} \tag{94}$$

where $\overline{\sigma}(\cdot)$ and $\underline{\sigma}(\cdot)$ denote the largest and smallest singular values of $(\cdot)$, respectively. ❐

If the Jacobian J at the initial condition has full rank, then one can show that Problem VI.1 is equivalent to the generalized eigenvalue problem [8]:

Problem VI.2 Given $J \in \mathbb{R}^{n_p \times n_q}$,

$$\underset{P, Q, \gamma}{\text{minimize}} \quad \gamma^2 \qquad \begin{array}{c} \text{such that } Q \le J^T P J \le \gamma^2 Q, \quad P > 0, \quad Q > 0 \\ \text{and } P \text{ and } Q \text{ are diagonal} \end{array} \tag{95}$$

❐

Specifically, if the Jacobian J has full rank, then the solution to the scaling problem (Problem VI.1) is given in terms of the solution of the GEVP (Problem VI.2) by $K_p = P^{1/2}$ and $K_q = Q^{-1/2}$, where γ is an upper bound for the condition number of the scaled matrix $K_p J K_q$. However, since the GEVP is bilinear in γ and Q, Problem VI.2 cannot be solved directly by most LMI parsers such as YALMIP. Instead, we solve the GEVP by using the method of bisection, that is, via the following algorithm:

Algorithm VI.1 A bisection-based GEVP solver which uses an LMI parser such as YALMIP and SeDuMi as solver to deal with the GEVP when γ is fixed:

function $\begin{bmatrix} P, Q, \gamma \end{bmatrix}$ = **GEVP**$\left(\quad J, \quad \varepsilon \quad \right)$
  $\gamma_{\text{low}} = 1, \quad P = I_{n_p}, \quad Q = I_{n_q}$
  $\gamma_{\text{high}} = \overline{\sigma}(J) \,/ \underline{\sigma}(J)$
  while $\left(\quad \gamma_{\text{high}} - \gamma_{\text{low}} > \varepsilon \quad \right)\{$
    $\gamma = \left(\gamma_{\text{high}} - \gamma_{\text{low}}\right)/2$
    \\Use an LMI parser and Sedumi as solver such as YALMIP to determine if γ is feasible:
    if $\left(\quad \exists P, Q \quad \text{such that} \quad Q \leq J^T P J \leq \gamma^2 Q \quad \text{and} \quad P > 0 \quad \text{and} \quad Q > 0 \quad \right)\{$
      $\gamma_{\text{high}} = \gamma$
      store the solutions P, Q
    } **else** {
      $\gamma_{\text{low}} = \gamma$
  }}
  return P, Q, γ ∎

Remark VI.1 If the system is already perfectly scaled, that is, if the condition number of J is 1, then the GEVP algorithm (Algorithm VI.1) will return $P = I_{n_p}$ and $Q = I_{n_q}$. ∎

Remark VI.2 SeDuMi is only capable of solving semi-definite LMIs. Therefore, to enforce the positive definiteness of P and Q in Problem VI.2, we impose the constraints

$$P > \varepsilon_p I_{n_p}, \qquad Q > \varepsilon_q I_{n_q} \tag{96}$$

where $\varepsilon_p = \varepsilon_q = 10^{-7}$ throughout the remainder of this paper. ∎

As we will show in Sec. VII, the use of the scaling matrices obtained in this manner improves both the accuracy and speed of computing the collocation estimates.

VII. Simulations

In the following sections, use both a 4-stage Gauss collocation integrator and Matlab's `ode45` to integrate:

1) The attitude dynamics of a satellite.
2) The orbital dynamics of a satellite.
3) The full 6DOF dynamics of satellite, that is, both the attitude and orbital dynamics of a satellite simultaneously.

where we compare how well the integrators preserve quadratic invariants such as the norm of the quaternion, angular momentum, position vector, and specific mechanical energy.

Attitude propagation The attitude dynamics of a rigid satellite in torque-free motion are given by

$$\begin{aligned} \frac{dq}{dt}(t) &= \frac{1}{2} q(t)\omega(t) \\ \frac{d\omega}{dt}(t) &= I^{-1}\Big(\omega(t) \times \big[I\omega(t)\big]\Big) \end{aligned} \tag{97}$$

where $q(t) \in \mathbb{R}^4$ denotes the quaternion representation of the satellite's attitude, $\omega(t) \in \mathbb{R}^3$ denotes the angular velocity of the satellite, $q(t)\omega(t)$ denotes the quaternion product, $I \in \mathbb{R}^{3\times 3}$ denotes the positive-definite moment of inertia of the satellite about its center of mass, and I, ω, and q are all expressed in the body-fixed frame. The attitude dynamics defined by Eq. (97) preserve the quadratic invariants:

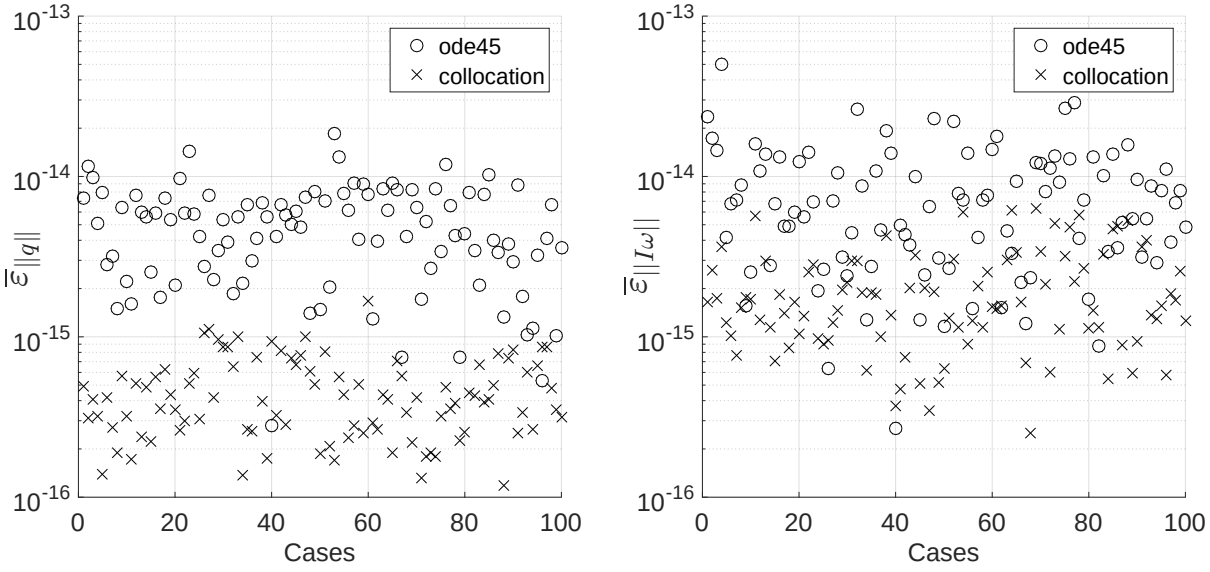
1) The quaternion norm: $\|q(t)\| = \|q(t_0)\| = 1$

2) The angular momentum: $\|I\omega(t)\| = \|I\omega(t_0)\|$

To compare how well the integrators preserve the these quadratic invariants, we integrate Eq. (97) using a 4-stage Gauss collocation integrator and Matlab's `ode45` with 100 uniformly-generated random initial conditions and inertia matrices, where the initial quaternion $q(t_0)$ is scaled to have a norm of 1, the angular velocity vector $\omega(t_0)$ is uniformly generated in the range $[-0.05, 0.05]$ rad/s, and the inertia matrix is randomly generated so that it is symmetric and positive definite with values approximately in the range $[1, 10]$. The dynamics are then integrated over 100 seconds using a timestep of $h = 0.5$ s, where the `AbsTol` and `RelTol` options of `ode45` are set to 10^{-20}. Under these conditions, Figs. 5a and 5b show the mean error evaluated over the N_s (in this case equal to 200) time steps in the quaternion and angular momentum norms for each of the 100 simulations, that is,

$$\overline{\varepsilon}_{\|q\|} \triangleq \frac{1}{N_s} \sum_{k=1}^{N_s} \left| \left\| q(t_0 + hk) \right\| - 1 \right| \tag{98}$$

$$\overline{\varepsilon}_{\|I\omega\|} \triangleq \frac{1}{N_s} \sum_{k=1}^{N_s} \left| \left\| I\omega(t_0 + hk) \right\| - \left\| I\omega(t_0) \right\| \right| \tag{99}$$



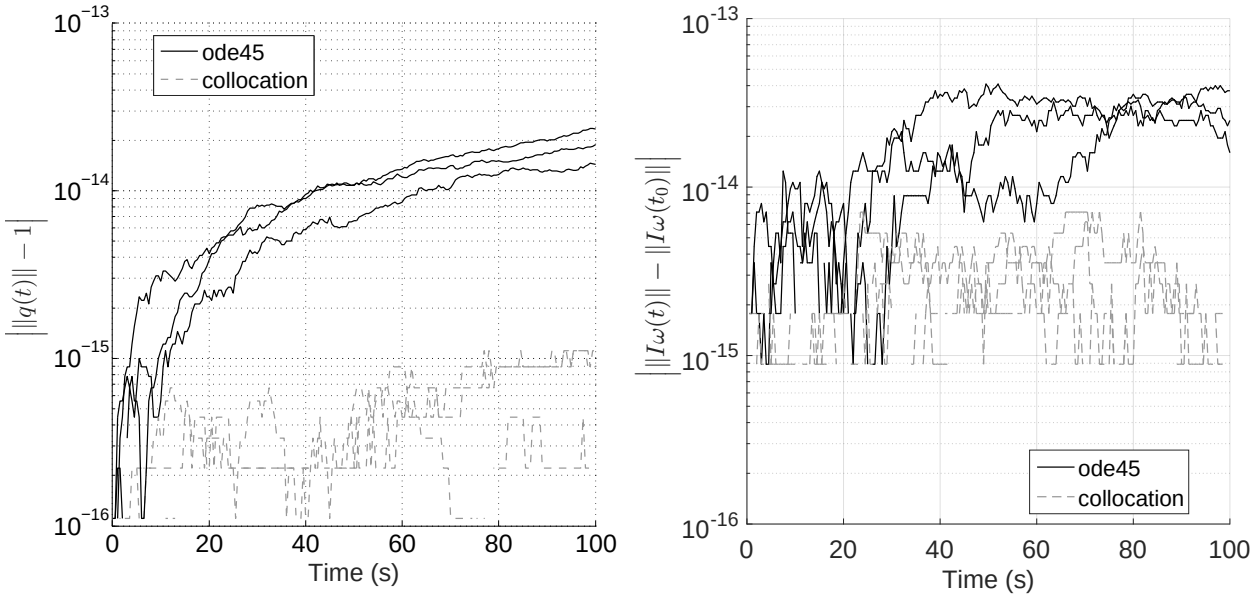
(a) Mean error in the quaternion norm. **(b) Mean error in the angular momentum norm in Eq.** (99)**.**

Fig. 5 Mean errors in the quaternion and angular momentum norms.

From Figs. 5a and 5b, we see that the collocation integrator typically produces a mean error that is 10 to 100 times smaller than `ode45`, although the increased accuracy comes at the price of increased computational time. Specifically, the mean and standard deviation of the collocation integrator's runtimes are 7.8 and 3.0 s, respectively, while the mean and standard deviation of `ode45`'s runtimes are 5.3 and 4.6 s, respectively. The errors in the quaternion and angular momentum norms for three randomly selected simulations are shown in Figs. 6a-6b for demonstrative purposes.

Remark VII.1 Since the condition number of the Jacobian was in the range $[200, 220]$ for all of the attitude dynamics cases that we considered, the collocation problem was already well-conditioned, and hence the GEVP-based scaling did not greatly improve the results. Hence the results shown in Figs. 5-6 do not use the GEVP-based scaling. In fact, since the GEVP scaling is the most time-consuming part of the algorithm, for these cases it is recommended to use the hybrid-jacobian collocation scheme only. ◻

Table 2 shows the mean and the standard deviation of the errors for both the schemes. Moreover, the corresponding statistics for the CPU time is provided. One can observe that the use of the collocation schemes improves the accuracy of the results both in terms of quaternions and angular momentum. Specifically, the improvement in the mean is more



(a) Error in the quaternion norm. **(b) Error in the angular momentum norm.**

Fig. 6 Errors in the quaternion and angular momentum norms during the first three simulations of the test campaign.

Method/Performance	ode45	Collocation	Self-scaling Collocation
$\overline{\varepsilon}_{\|q\|}$ (μ)	5.257e-15	4.710e-16	4.280e-16
$\overline{\varepsilon}_{\|q\|}$ (σ)	3.277e-15	2.725e-16	2.697e-16
$\overline{\varepsilon}_{\|I\omega\|}$, kg m^2/s ($\mu$)	8.587e-15	2.053e-15	2.001e-15
$\overline{\varepsilon}_{\|I\omega\|}$, kg m^2/s ($\sigma$)	7.538e-15	1.457e-15	1.561e-15
CPU Time, s (μ)	5.276e-0	7.859e-0	1.511e+1
CPU Time, s (σ)	4.622e-0	2.957e-0	4.021e+0

Table 2 Attitude propagation - statistics (100 simulations).

than one order of magnitude for the quaternion, and about 4 times in terms of angular momentum. Similar trends are observed for the standard deviations. This improvement is paid in terms of a larger CPU time (with a mean value of about 15 s for the self-scaling collocation versus a value of 5.3 s for the standard ode45. It is worth noting that there is no significant difference between the results with and without GEVP scaling, as expected, in terms of accuracy, which slightly improves when the GEVP is employed. However the larger CPU time required for solving the GEVP problem suggests to use the simple version of the algorithm in cases like this one.

Orbit propagation The orbital dynamics of a satellite are given by Eq. (35) in Example III.1, where the orbital dynamics preserve the quadratic invariants:

1) The norm of the position vector: $\|r(t)\| = \|r(t_0)\|$
2) The specific mechanical energy:

$$E(t) = \frac{1}{2}\|v(t)\|^2 - \frac{\mu_\oplus}{\|r(t)\|} = \frac{1}{2}\|v(t_0)\|^2 - \frac{\mu_\oplus}{\|r(t_0)\|} = E(t_0) \tag{100}$$

Furthermore, as in Section VII, we compare how well a 4-stage Gauss collocation integrator and Matlab's `ode45` preserve these quadratic invariants by integrating the orbital dynamics of Eq. (35) with 100 uniformly-generated random initial conditions. Specifically, we generate 100 random initial positions $r(t_0) \in \mathbb{R}^3$ with altitudes between 300 and 900 km, and choose an initial velocity vector $v(t_0)$ perpendicular to the initial position with the magnitude necessary to

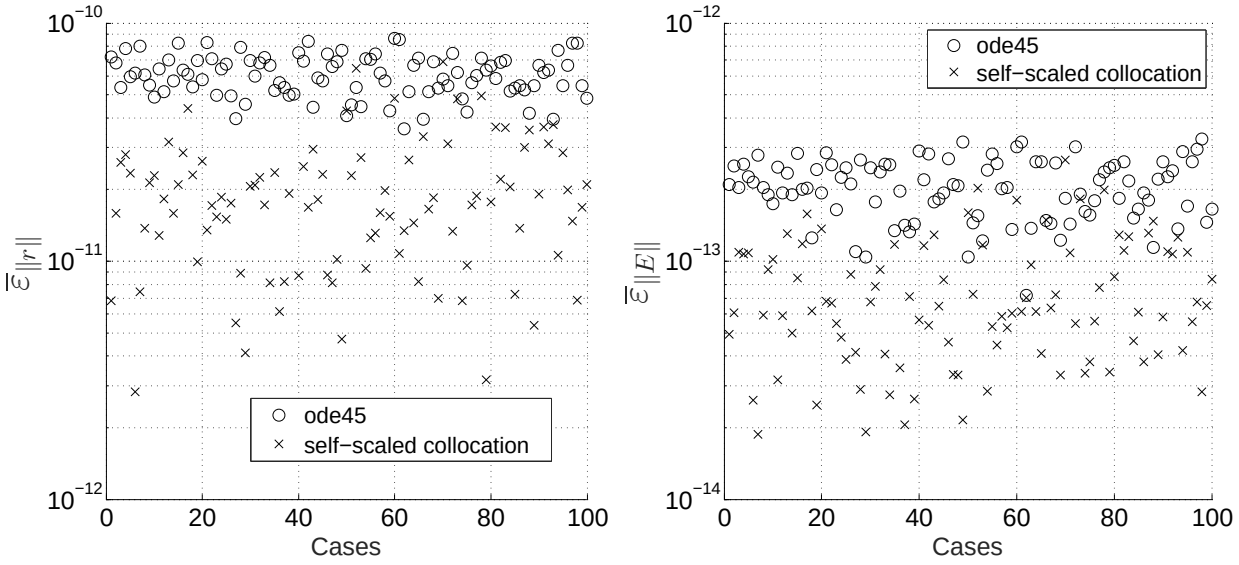
achieve a circular orbit. The dynamics are then integrated over one orbital period using a timestep of $h = 10$ s, where the `AbsTol` and `RelTol` options of `ode45` are set to 10^{-20}.

However, whereas the attitude dynamics in Eq. (97) were well-conditioned for the parameters we considered, the orbital dynamics in Eq. (35) are not, with condition numbers on the order of 22500. The use of the GEVP-based automatic scaling method reduces the condition numbers to the range $[50, 75]$. Furthermore, using the self-scaling collocation integrator, Figs. 7a and 7b show the mean error in the norm of the position vector and the mean error in the specific mechanical energy for each of the 100 simulations, that is,

$$\overline{\varepsilon}_{\|r\|} \triangleq \frac{1}{N_s} \sum_{k=1}^{N_s} \left| \left\| r(t_0 + hk) \right\| - \left\| r(t_0) \right\| \right| \tag{101}$$

$$\overline{\varepsilon}_{\|E\|} \triangleq \frac{1}{N_s} \sum_{k=1}^{N_s} \left| \left\| E(t_0 + hk) \right\| - \left\| E(t_0) \right\| \right| \tag{102}$$

with N_s that in this case is not constant, as the final time for each of the orbit changes according to the random value of the radius.



(a) Mean error in the norm of the position vector.

(b) Mean error in the specific mechanical energy.

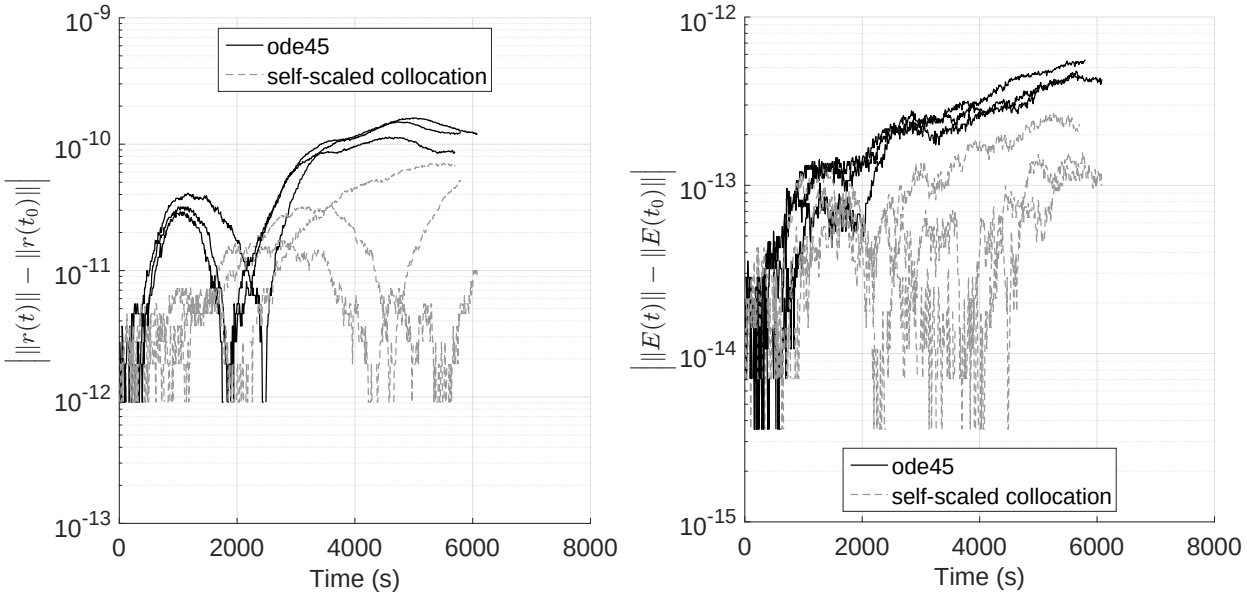
Fig. 7 Mean errors in the norm of the position vector and the specific mechanical energy for 100 randomly generated initial conditions.

From Figs. 7a-7b, we see that the self-scaled collocation integrator typically produces a mean error that is 3 times smaller than ode45, although again the at the price of increased computational time. Specifically, the mean and standard deviation of the self-scaled collocation integrator's runtimes are 13.8 and 0.47 s, respectively, while the mean and standard deviation of `ode45`'s runtimes are 0.31 and 0.17 s, respectively. The errors in the norm of the position and specific mechanical energy for three randomly selected simulations are shown in Figs. 8a-8b for demonstrative purposes.

Remark VII.2 Since the norm of the position vector is a quadratic invariant, then we see that the specific mechanical energy is of the form

$$E = \frac{1}{2} \|v(t)\|^2 - c = \frac{1}{2} \|v(t_0)\|^2 - c \tag{103}$$

and hence it can also be classified as a quadratic invariant. Table 3 summarizes the results obtained for this case. Also in this case units are omitted, and are km for the radius, and km^2/s^2 for the specific energy. ◻



(a) Error in the norm of the position vector. **(b) Error in the specific mechanical energy.**

Fig. 8 Errors in the norm of the position vector and specific mechanical energy during the first three simulations of the test campaign.

Method/Performance	ode45	Collocation	Self-scaling Collocation
$\overline{\varepsilon}_{\|r\|}$, km ($\mu$)	6.119e-11	2.084e-11	1.993e-11
$\overline{\varepsilon}_{\|r\|}$, km ($\sigma$)	1.194e-11	1.285e-11	1.258e-11
$\overline{\varepsilon}_{\|E\|}$, km^2/s^2 ($\mu$)	2.082e-13	7.953e-14	7.759e-14
$\overline{\varepsilon}_{\|E\|}$, km^2/s^2 ($\sigma$)	5.666e-14	4.573e-14	4.663e-14
CPU Time, s (μ)	3.068e-1	1.473e+1	1.385e+1
CPU Time, s (σ)	1.733e-1	5.265e-0	4.763e-0

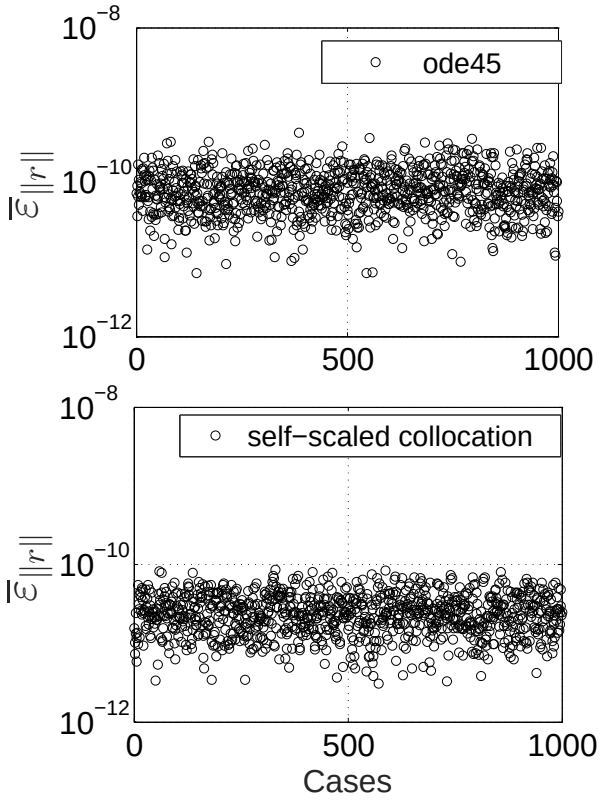
Table 3 Orbit propagation - statistics (100 simulations).

Also in this case we can observe an improvement of the accuracies for both the invariants of the system, that is, the radius and the specific energy of the spacecraft. The mean error is reduced by about 70% for what regards the radius and 63% in terms of energy when the self-scaling approach is used. No significant variations of the error' standard deviations are observed. Finally the CPU time is larger when the self-scaling collocation approach is implemented, as in the previous case. Also in this case no significant differences are observed when the GEVP is scaled with respect to the case where the collocation scheme is employed without it. For instance the error in terms of radius magnitude reduced from 2.084e-11 (when there is no GEVP scaling) to 1.993e-11 (with GEVP scaling). However, an interesting result with respect to the previous case is that despite the extra CPU time required to compute the GEVP scaling matrices, the overall CPU time to obtain a solution is 1 s less than the case when there is no GEVP scaling. This means that the better-conditioned problem is not only more accurate, but the optimization underlying the implicit scheme is significant quicker in finding a solution if compared to the non-scaled case.

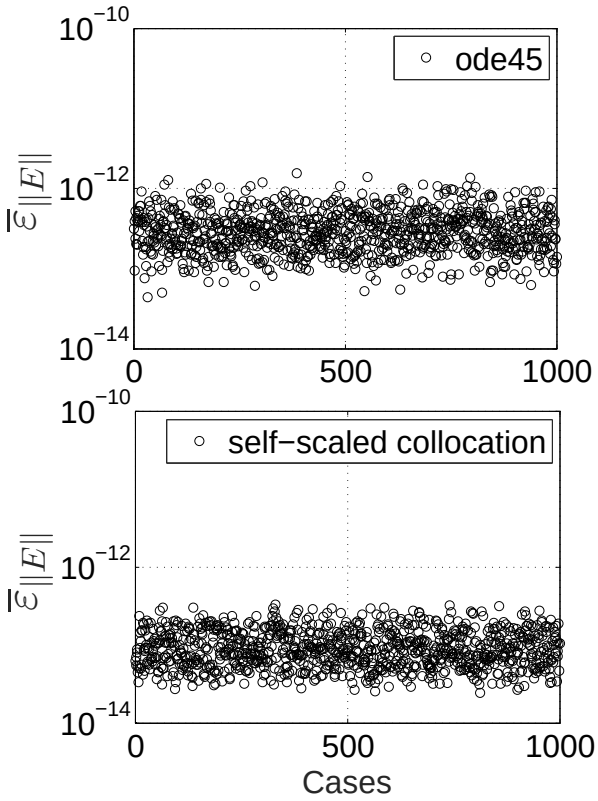
6-DOF propagation The full six degree of freedom (6DOF) dynamics of a satellite are obtained by simultaneously simulating both the attitude (defined by Eq. (97)) and orbital dynamics (defined by Eq. (35)), in which case we should find that the norm of the position vector, specific mechanical energy, quaternion, and angular momentum are preserved. To compare how well a 4-stage Gauss collocation integrator and Matlab's `ode45` preserve these quadratic invariants, we integrate the 6DOF dynamics over one orbital period with a timestep of $h = 5$ s and 1000 uniformly-generated random initial conditions, that is, random initial positions, velocities, quaternions, angular velocities, and inertia matrices, as

described in Sections VII-VII.

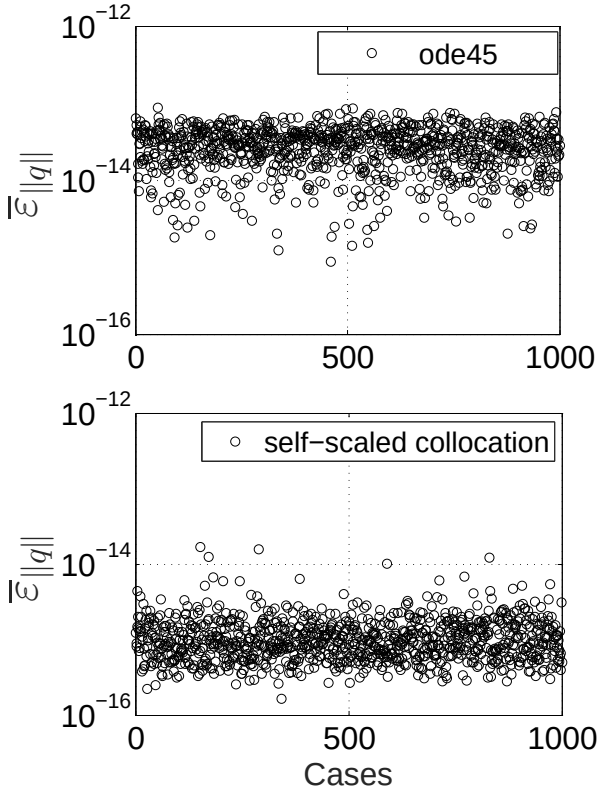
As in the case of pure orbital dynamics, we find that the condition number of the Jacobian is approximately 300-400 times smaller after applying the GEVP-based scaling, and hence the automatic scaling is used throughout this section. Specifically, using the automatic GEVP-based scaling, Fig. 9 shows that mean errors in the norm of the position vector, specific mechanical energy, quaternion, and angular momentum are approximately 10 times smaller than `ode45`, which is also apparent in the three randomly selected simulations shown in Fig. 10. However, as in the previous cases, the self-scaled collocation integrator is slower than Matlab's `ode45`, with a mean and standard deviation runtime of 68.3 and 9.3 s, respectively, compared to `ode45`'s mean and standard deviation of 25.4 and 18.3 s. However, although `ode45` typically is faster than the self-scaled collocation integrator, the worst-case `ode45` runtime was 199.5 s, compared to 137.5 for the self-scaled integrator.



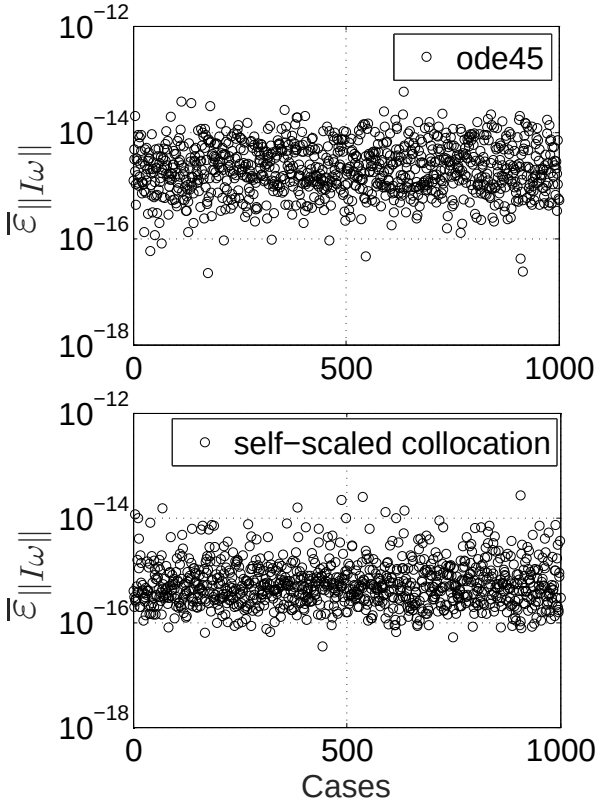
(a) Mean error in the norm of the position vector.



(b) Mean error in the specific mechanical energy.

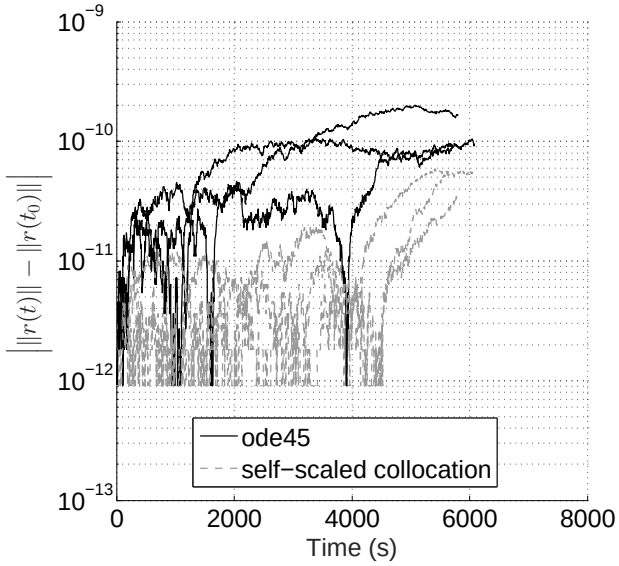


(c) Mean error in the quaternion norm.

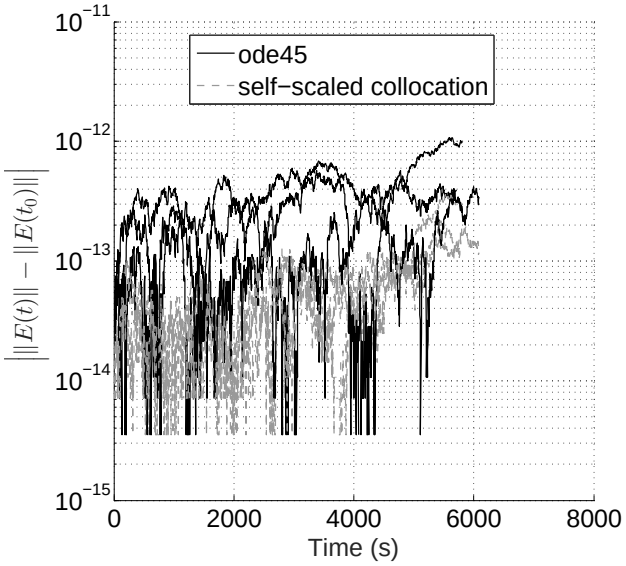


(d) Mean error in the angular momentum norm.

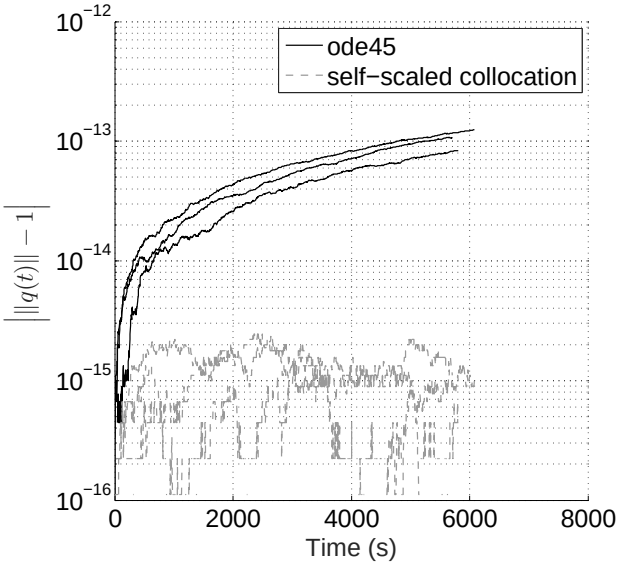
Fig. 9 Mean errors in the norm of the position vector, specific mechanical energy, quaternion, and angular momentum over one orbital period.



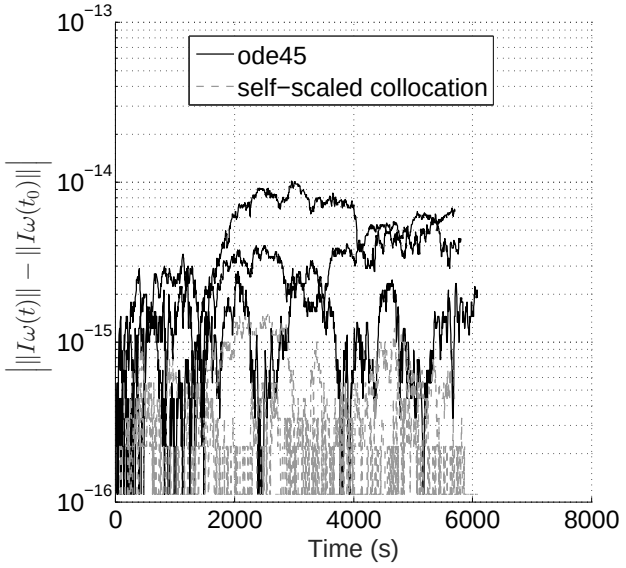
(a) Error in the norm of the position vector.



(b) Error in the specific mechanical energy.



(c) Error in the quaternion norm.



(d) Error in the angular momentum norm.

Fig. 10 Errors in the norm of the position vector, specific mechanical energy, quaternion, and angular momentum during the first three simulations of the test campaign.

Method/Performance	ode45	Collocation	Self-scaling Collocation
$\overline{\varepsilon}_{\|q\|}$ (μ)	2.929e-14	1.327e-15	1.131e-15
$\overline{\varepsilon}_{\|q\|}$ (σ)	1.653e-14	7.359e-16	7.476e-16
$\overline{\varepsilon}_{\|I\,\omega\|}$, kg m^2/s ($\mu$)	2.775e-15	5.381e-15	1.048e-15
$\overline{\varepsilon}_{\|I\,\omega\|}$, kg m^2/s ($\sigma$)	3.288e-15	4.341e-15	2.256e-15
$\overline{\varepsilon}_{\|r\|}$, km ($\mu$)	9.057e-11	2.860e-11	2.672e-11
$\overline{\varepsilon}_{\|r\|}$, km ($\sigma$)	5.833e-11	1.533e-11	1.508e-11
$\overline{\varepsilon}_{\|E\|}$, km^2/s^2 ($\mu$)	3.470e-13	1.076e-13	1.028e-13
$\overline{\varepsilon}_{\|E\|}$, km^2/s^2 ($\sigma$)	2.139e-13	5.635e-14	5.649e-14
CPU Time, s (μ)	2.543e+1	1.842e+2	6.835e+1
CPU Time, s (σ)	1.823+1	2.036e+1	9.258e+0

Table 4 6-DOF propagation - statistics (100 simulations).

The results associated with the 6-DOF case are summarized in Table 4. Also in this case we can observe a slight improvement when the GEVP scaling with respect to the unscaled case. Both are in general more accurate than the traditional Runge-Kutta schemes. For instance, the accuracy in terms of quaternion magnitude invariant is about 20 times better than the results obtained by using `ode45`, and three times better in terms of radius magnitude. In terms of performance of the hybrid-Jacobian collocation method with and without the GEVP scaling, we can observe that the accuracy is equal or better when the GEVP is employed for this case. However, the CPU time required to obtain a solution is much smaller when the GEVP technique is used (the mean CPU time is about 70% less than the time obtained without GEVP scaling), which means that, as expected, a better conditioned problem converges faster, despite the time spent in the computation of the scaling matrices, and can lead to a significant difference of the running time for large simulation campaigns.

Long-time run Finally, to show the validity of the results over longer timespans, an example of simulation carried over a total time of 30 orbits has been included. The results are depicted in Figs. 11a through 11d.

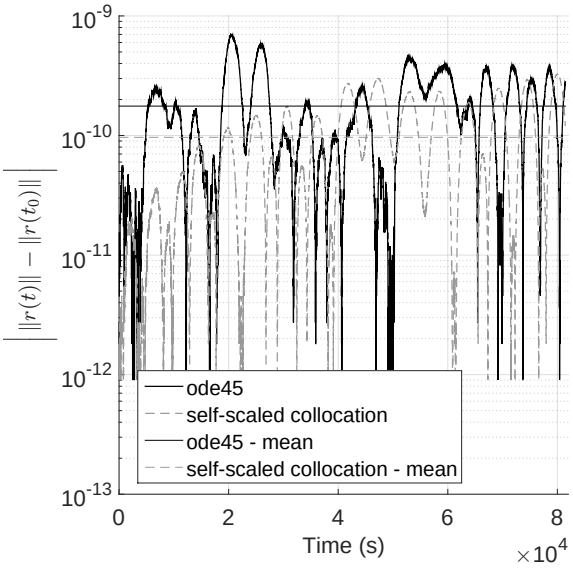
Figures 11a-11d show the results obtained for one case in terms of errors with respect to the true invariants. Moreover, the mean of the errors are overlapped. One can observe that even for longer runs the proposed algorithm behaves better than the traditional schemes. This is true for all the invariants involved, and is particularly stark for the quaternion norm and the angular momentum. For this specific case the collocation algorithm becomes also more efficient in terms of CPU time. Indeed, the ode45 scheme generates the solution in 812 s, against a CPU time of 495 s when the collocation scheme is adopted, which means that for a long run the proposed method is both more accurate and faster than the traditional solutions.

VIII. Conclusions

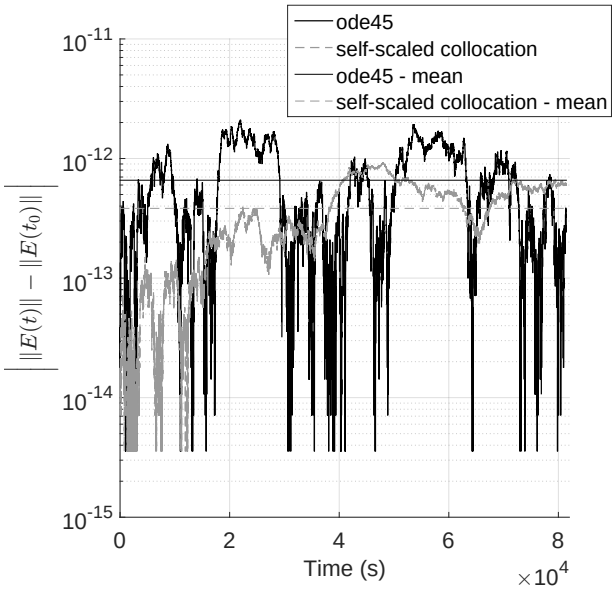
We presented a class of self-scaling collocation integrators which, unlike explicit Runge-Kutta integrators, automatically preserve quadratic constants of motion such as energy, linear momentum, and angular momentum. Specifically, whereas variable timestep Runge-Kutta integrators, such as the Dormand-Prince method (`ode45`), can theoretically reduce the integration errors in invariants by reducing the integration timestep, the Gauss collocation integrators that we presented automatically preserve such invariants, regardless of the integration timestep.

However, since collocation integrators in general require numerical optimization to solve an implicit set of equations which are difficult and computationally expensive to solve when the problem's Jacobian becomes ill-conditioned, we introduced a GEVP-based approach to automatically scale the collocation conditions, thereby making their solution faster and more reliable. For example, for the case of the orbit propagation the GEVP-based scaling approach yielded a Jacobian with a condition number 300-400 smaller than in the unscaled case.

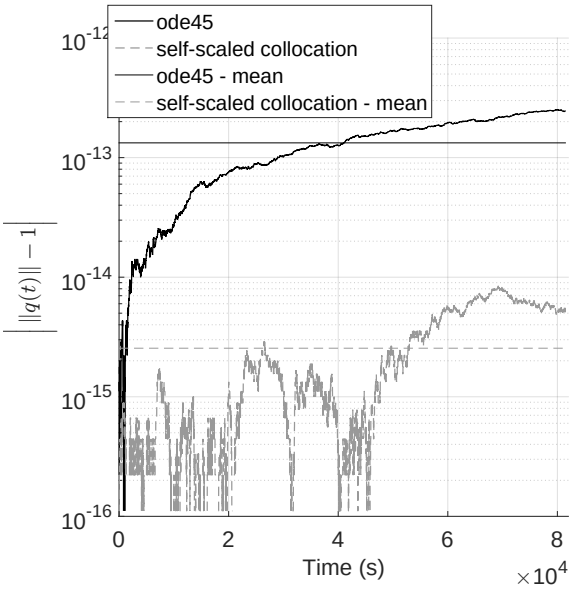
In addition, we introduced a novel hybrid Jacobian computation technique for collocation methods which splits the computation into a component that can be computed *a priori*, and a contribution due to the differential equation that can be computed efficiently using a complex-step approximation, thereby making the Jacobian computation faster and more



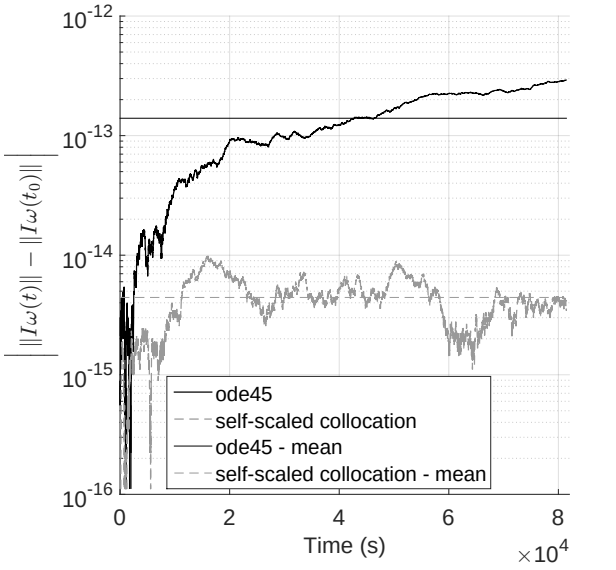
(a) Error in the norm of the position vector.



(b) Error in the specific mechanical energy.



(c) Error in the quaternion norm.



(d) Error in the angular momentum norm.

Fig. 11 Long Timespan - Errors in the norm of the position vector, specific mechanical energy, quaternion, and angular momentum for a total time equal to 10 orbits.

accurate than via traditional finite difference approaches. Together, the GEVP-based self-scaling collocation integrator and hybrid Jacobian computation that we presented are better at preserving quadratic invariants such as the norm of the quaternion, angular momentum, position vector, and specific mechanical energy than variable timestep explicit Runge-Kutta methods, such as Matlab's `ode45`, over a wide range of problems and initial conditions. Specifically, as we demonstrated in Sec. VII, our self-scaling integrator typically achieved errors in the quadratic invariants that were one to two orders of magnitude smaller than `ode45`.

Acknowledgments

This work was supported in part by the Institutional Strategy of the University of Bremen, funded by the German Excellence Initiative.

References

[1] Butcher, J. C., *Numerical Methods for Ordinary Differential Equations*, Wiley, 2003.

[2] Hairer, E., Norsett, S., and Wanner, G., *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd ed., Springer-Verlag, 2000.

[3] Hairer, E., and Wanner, G., *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd ed., Springer-Verlag, 2010.

[4] Hairer, E., Lubich, C., and Wanner, G., *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equation*, 2nd ed., Vol. 31, Springer-Verlag, 2006.

[5] Kreyszig, E., *Advanced Engineering Mathematics*, eigth ed., Wiley, 1999.

[6] Feagin, T., "A tenth-order Runge-Kutta method with error estimate," *Proceedings of the IAENG Conference on Scientific Computing*, 2007.

[7] Iserles, A., *Numerical Analysis of Differential Equations*, 2nd ed., Cambridge University Press, 2009.

[8] Boyd, S., Ghaoui, L. E., E., E. F., and Balakrishnan, V., *Linear Matrix Inequalities in System and Control Theory*, Studies in Applied Mathematics, Vol. 15, 1994.

[9] Martins, J. R. R., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software, Vol. 29, No. 3, September 2003, Pages 245262*, 2003. doi:10.1145/838250.838251.

[10] Curtis, H. D., *Orbital Mechanices for Engineering Students*, Butterworth-Heinemann, 2005.

[11] L., S., and W., R. M., "The MATLAB ODE Suite," *SIAM Journal on Scientific Computing*, Vol. 18, 1997, pp. 1–22.

[12] Cooper, G. J., "Stability of Runge-Kutta methods for trajectory problems," *IMA journal of numerical analysis*, Vol. 7, No. 1, 1987, pp. 1–13.

[13] Bernstein, D. S., *Matrix Mathematics*, 2nd ed., Princeton University Press, Princeton, NJ, 2009.

[14] Sagliano, M., and Theil, S., "Hybrid Jacobian Computation for Fast Optimal Trajectories Generation," *AIAA Guidance, Navigation, and Control Conference, Boston, USA,*, 2013. doi:10.2514/6.2013-4554.

[15] Rao, A. V., "A Survey of Numerical Methods for Optimal Control," *AAS/AIAA Astrodynamics Specialist Conference, AAS Paper 09-334, Pittsburgh, PA, August 10 - 13*, 2009.

[16] Lai, K. L., and Crassidis, J. L., "Extensions of the first and second complex-step derivative approximations," *Journal of Computational and Applied Mathematics*, Vol. 219, 2008, pp. 276–293. doi:10.1016/j.cam.2007.07.026.

[17] Fike, J., and J.Alonso, "The Development of Hyper-Dual Numbes for Exact Second-Derivative Calculations," *49th AIAA Aerospace Sciences meeting including the New Horizons Forum and Aerospace Exposition , Orlando, USA, 2011*, 2011.

[18] D'Onofrio, V., Sagliano, M., and Arslantas, Y., "Exact Hybrid Jacobian Computation for Optimal Trajectory Computation via Dual Number Theory," *AIAA Science and Technology Forum and Exposition*, 2016.

[19] Sagliano, M., "Performance analysis of linear and nonlinear techniques for automatic scaling of discretized control problems," *Operations Research Letters, Vol.42 Issue 3, May 2014, pp. 213-216*, 2014. doi:10.1016/j.orl.2014.03.003.

[20] Lu, P., "Entry Guidance: A Unified Method," *Journal of Guidance, Control, and Dynamics, Vol.37 No.3, May-June*, 2014. doi:10.2514/1.62605.

[21] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed., Society for Industrial and Applied Mathematics, 2010. doi:10.1137/1.9780898718577, URL `http://epubs.siam.org/doi/abs/10.1137/1.9780898718577`.

[22] Lüscher, M., and Wolff, U., "How To Calculate the Elastic Scattering Matrix in Two-Dimensional Quantum Field Theories by Numerical Simulation," *Nuclear Physics B*, Vol. 339, 1990, pp. 222–252.

[23] Mendes, T., "Handling Excited States on the Lattice: The GEVP Method," *Acta Physica Polonica, B Proceedings Supplement*, Vol. 3, No. 4, 2010, pp. 905–910.

[24] Lofberg, J., "YALMIP: a Toolbox for Modeling and Optimization in MATLAB," *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, 2004. doi:10.1109/CACSD.2004.1393890.

[25] Storm, J., "Using SeDuMi 1.02, A Matlab Toolbox for Optimization over Symmetric Cones," *Optimization Methods and Software*, 1999. doi:10.1080/10556789908805766.