Celest Mech Dyn Astr
DOI 10.1007/s10569-013-9522-7

ORIGINAL ARTICLE

Orbit and uncertainty propagation: a comparison of Gauss–Legendre-, Dormand–Prince-, and Chebyshev–Picard-based approaches

Jeffrey M. Aristoff · Joshua T. Horwood · Aubrey B. Poore

Received: 23 April 2013 / Revised: 10 July 2013 / Accepted: 27 September 2013

Abstract We present a new variable-step Gauss–Legendre implicit-Runge–Kutta-based approach for orbit and uncertainty propagation, VGL-IRK, which includes adaptive step-size error control and which collectively, rather than individually, propagates nearby sigma points or states. The performance of VGL-IRK is compared to a professional (variable-step) implementation of Dormand–Prince 8(7) (DP8) and to a fixed-step, optimally-tuned, implementation of modified Chebyshev–Picard iteration (MCPI). Both nearly-circular and highly-elliptic orbits are considered using high-fidelity gravity models and realistic integration tolerances. VGL-IRK is shown to be up to eleven times faster than DP8 and up to 45 times faster than MCPI (for the same accuracy), in a serial computing environment. Parallelization of VGL-IRK and MCPI is also discussed.

Keywords Implicit Runge–Kutta · Uncertainty propagation · Satellites · Variable step

1 Introduction

Precise orbit propagation is needed for many functions in space situational awareness (SSA) such as tracking, catalog maintenance, conjunction analysis, maneuver detection, and object characterization. As the number of objects that must be tracked, cataloged, and characterized grows from tens of thousands to hundreds of thousands due in part to potential collision events in the future and improved sensors, fast and adaptive algorithms for orbit propagation are needed to handle the ever-increasing computational demand without sacrificing accuracy. What is more, efficient methods for uncertainty propagation, which typically involve the propagation of multiple orbital states, are needed to support advanced space surveillance. The need to properly and efficiently characterize uncertainty in a space object's orbital state (i.e., covariance and uncertainty realism) is, in fact, one of the findings made by the National

J. M. Aristoff (✉) · J. T. Horwood · A. B. Poore
Numerica Corporation, Loveland, CO, USA
e-mail: jeff.aristoff@numerica.us

Table 1 Summary of the propagators evaluated in this study

	VGL-IRK	DP8	VGL-s	MCPI
Runge–Kutta type	Implicit	Explicit	Implicit	Implicit
Runge–Kutta scheme	Gauss–Legendre	Dormand–Prince	Gauss–Legendre	Gauss–Chebyshev
Implemented by	Aristoff et al.	Brankin et al. (1991)	Jones (2012)	Aristoff et al.
Variable-step	Yes	Yes	Yes	No
A-stable	Yes	No	Yes	Yes
Parallelizable	Yes	No	Yes	Yes
Symmetric	Yes	No	Yes	Yes
Symplectic	Yes	No	Yes	No
Superconvergent	Yes	No	Yes	No

Gauss–Legendre and Gauss–Chebyshev (Lobatto) are implicit Runge–Kutta schemes. Dormand–Prince is an explicit Runge–Kutta scheme. Variable-step methods estimate and control the numerical truncation error by adaptively adjusting the step size based upon an estimate of the numerical truncation error. A-stable methods do not exhibit instability problems when solving stiff equations. Parallelizable methods do not require sequential evaluation of the force model over a given time step. Symmetric schemes preserve time-reversibility of a dynamical system (if applicable). Symplectic schemes preserve the Hamiltonian of a system (if applicable). Superconvergent schemes achieve the highest possible convergence rate (i.e., an s-stage method achieves order $2s$

Research Council's recent assessment of Air Force Space Command's Astrodynamics Standards (Nielsen et al. 2012).

The purpose of this paper is to (1) present a new variable-step Gauss–Legendre implicit-Runge–Kutta-based approach for orbit and uncertainty propagation (VGL-IRK), (2) demonstrate the viability of VGL-IRK for precise orbit and uncertainty propagation in a realistic simulation environment, and (3) consider alternative Runge–Kutta-based approaches for orbit and uncertainty propagation such as modified Chebyshev–Picard iteration (MCPI) and Dormand–Prince. The general theory of Gauss–Legendre implicit Runge–Kutta and its use for orbit and uncertainty propagation was presented in a companion paper (Aristoff et al. 2014). Here it will be shown that VGL-IRK is up to eleven times faster than Dormand–Prince 8(7) and up to 45 times faster than a fixed-step, optimally-tuned MCPI (for the same accuracy), in a serial computing environment. Parallelization of MCPI and VGL-IRK will also be discussed. Table 1 lists the propagators evaluated in this study, as well as their underlying mathematical properties. We begin by reviewing state-of-the-art numerical methods for orbit propagation and motivating the need for variable-step error estimation and control.

2 Background

Orbit propagation for SSA requires finding precise (numerical) solutions to the equations of motion [i.e., second-order initial value problems (IVPs)]. Numerous numerical integration methods can be used for this purpose, some of which have been specialized to the problem of orbit propagation (Montenbruck 1992; Montenbruck and Gill 2000; Jones and Anderson 2012). Current state-of-the-art numerical integrators used for orbit propagation include Dormand–Prince 8(7) (DP8), Runge–Kutta–Nystrom 12(10) (RKN12), Adams–Bashforth–Moulton (ABM), and Gauss–Jackson (GJ). Both DP8 and RKN12 are high-order explicit Runge–Kutta methods that use variable-step-size error control (Iserles 2004; Butcher 2008; Hairer et al. 2009). However, RKN12 is an integration method that cannot handle velocity-dependent forces, and so we will not consider it further. ABM and GJ are predictor-corrector methods that utilize an explicit method for the prediction and an implicit method for the correction. Fixed-step implementations of GJ are widely used in numerical integration problems for astrodynamics and dynamical astronomy. For example, an eighth-order GJ algorithm has been used for space surveillance since the 1960s (Berry and Healy 2004). It is worth noting that although GJ will reduce its step size (typically by factors of two) to ensure convergence of a given time step, this does not qualify GJ as a variable-step integration method. Variable-step methods quantify the difference between the numerical solution to the IVP and the true solution to the IVP. Estimation of this numerical truncation error occurs after each time step of the method, and it is used to decide whether to accept or reject the step, and how to change the subsequent step size.[1] Of the aforementioned state-of-the-art numerical integrators, we select Dormand–Prince 8(7) for the basis of comparison because it is a high-order variable-step method and because we possess a professional C++ implementation of it, courtesy of Brankin et al. (1991).

Our work on variable-step implicit Runge–Kutta (IRK) methods has been motivated by the work of Beylkin and Sandberg (2012), Bradley et al. (2012), and Bai and Junkins (2011), who studied the use of fixed-step IRK methods for orbit propagation.[2] Gauss–Legendre IRK (GL-IRK) and Gauss–Chebyshev IRK methods are of particular interest because, unlike the above explicit methods, they are parallelizable, thus amenable to high-performance computing, and A-stable at all orders, thus allowing for larger (and fewer) time steps to be taken without sacrificing stability (Iserles 2004; Butcher 2008; Hairer and Wanner 2010). The caveat is that a nonlinear system of equations must be solved via iterative methods at each time step. While this is a clear disadvantage for implicit methods in general, for the perturbed two-body problem, one may use approximate analytical solutions to warm-start the iterations, leading to very fast convergence. By tuning the number of time steps, the number of stages (nodes) per time step, and the convergence criteria (iteration tolerance for a given time step), these authors demonstrated that IRK methods can be made both precise and efficient, at least for nearly-circular orbits, based on the number of force-model evaluations required to propagate the orbit.[3] In Junkins' study (Bai and Junkins 2011), zonal harmonics in the Earth's gravity up to degree five were considered; more realistic gravitational force models were considered by Beylkin and Sandberg (2012) and Bradley et al. (2012), specifically, zonal, tesseral, and sectorial harmonics up to degree and order 70.

Variable-step orbital propagators, on the other hand, can be considered self-tuning. The user selects an acceptable level of numerical truncation error (per step) in the state of the object, and the method then estimates and controls the error by adjusting the step size so that the error is close to but does not exceed the specified error tolerance. As a result, the work devoted to each step and the accuracy achieved in the step are balanced for overall efficiency (Butcher 2008; Shampine 1994, 2005). For this reason, modern numerical integration methods use variable-step error estimation and control (Shampine 2005), and reject steps that are too large for a given accuracy requirement. Note that fixed-step propagators do not reject steps because there is no accuracy requirement.

[1] Variable-step methods typically change the step size for each time step.

[2] Chebyshev–Picard iteration (Bai and Junkins 2011), is an alternative, but mathematically equivalent (in the sense that it produces the same discrete approximation) approach to Gauss–Chebyshev–Lobatto IRK using fixed-point (Picard) iteration (Aristoff and Poore 2012; Wright 1970; Hulme 1972).

[3] In the perturbed two-body problem, evaluation of the force models (the Earth gravity model in particular) is the dominant cost of orbit propagation. The number of force-model evaluations can thus be used as a crude measure of propagator performance.

For an object in highly-elliptic orbit, a variable-step propagator will take smaller steps near perigee and larger steps near apogee, thereby solving the IVP in an efficient manner. Conversely, fixed-step propagators such as Gauss–Jackson (Berry and Healy 2004) and MCPI, as reported in the literature (Bai and Junkins 2011), take larger steps near perigee and smaller steps near apogee. As a result, fixed-step propagators are forced to take extremely small time steps throughout the entire propagation in order to achieve the same accuracy as that of a variable-step method of the same order. Alternatively, the equations of motion can be transformed [e.g., using the Sundman or generalized Sundman transform (Berry and Healy 2002)] so that fixed steps can be taken in an orbital anomaly (e.g., eccentric anomaly or true anomaly), rather than in time, in an effort to distribute the numerical truncation error evenly across the steps. Unfortunately, this approach results in the need to numerically integrate an additional ordinary differential equation and it only approximately balances the work along the orbit, in part because the Sundman transform is based on unperturbed Keplerian dynamics (i.e., perturbations to the two-body problem are not accounted for in the transform).[4]

For nearly-circular orbits, one could argue that a fixed-step propagator is sufficient. However, this statement is defensible only if extensive simulation studies have been performed in order to determine the relationship between the step size and the resulting numerical truncation error for a given force model, initial orbital state, and initial time, so that one need not haphazardly select the step size. While it is possible to develop heuristics for this purpose, by taking fixed steps, one can only approximately balance the work per step and the associated accuracy because the forces acting on an object in near Earth orbit are not uniform along the orbit. What is more, heuristics developed for one set of force models would be different from those needed for a different set of force models, thereby limiting the applicability of such fixed-step approaches.

The key to making a variable-step propagator is developing an efficient and accurate method for error estimation. For explicit Runge–Kutta-based propagators, such as DP8, this is straightforward. Error estimates can be cheaply embedded into the method (Hairer et al. 2009; Shampine 1994, 2005). For implicit Runge–Kutta-based propagators, this approach is unsatisfactory (Hairer and Wanner 2010). Instead, we developed an error estimator which makes use of the propagated solution in order to select the optimal time step for a given tolerance, thereby controlling the accumulation of local error (i.e., error per step), and minimizing the cost of error estimation and propagation per time step (Aristoff and Poore 2012; Aristoff et al. 2014). This feature, among others, distinguishes our work from that of Bai and Junkins (2011), Beylkin and Sandberg (2012) and Bradley et al. (2012), who studied the use of fixed-step IRK methods for orbit propagation. An alternative implementation of variable-step GL-IRK, coined VGL-s was developed by Jones (2012) using an approach for error estimation suggested by Houwen and Sommeijer (1990). VGL-s was shown to have comparable performance (in a serial computing environment) to DP8 in a geostationary Earth orbit (GEO) scenario, better performance in a low Earth orbit (LEO) scenario, and worse performance in a highly-elliptic (Molniya) orbit scenario. We recently showed that VGL-IRK (in a serial computing environment) outperforms DP8 in all three of these orbital regimes, measured by the number of force-model evaluations (Aristoff and Poore 2012). We will make a direct comparison between VGL-s, DP8, MCPI, and VGL-IRK in Sect. 4, using runtime measurements whenever possible.

As previously mentioned, precise uncertainty propagation, in addition to precise orbit propagation, is needed to support numerous methods for advanced SSA. A large class of methods for propagating the uncertainty in an object's state (i.e., its probability density

[4] The generalized Sundman transform can account for some perturbations to the two-body problem.

function) requires finding high-order numerical solutions to ensembles of IVPs. Uncertainty propagation via the unscented Kalman filter (UKF) (Julier and Uhlmann 2004), for example, requires $2n+1$ trajectory (orbit) propagations, where n is the dimension of the state.[5] Unlike alternative approaches for uncertainty propagation, VGL-IRK exploits the proximity of the particles or states and enables them to be propagated collectively, rather than individually. Specifically, once the first state in the ensemble is propagated, the remaining states can be propagated at a fraction of the cost by reusing the (near) optimal step sizes taken during propagation of the first state, and by using the solution to the first IVP itself to warm-start the iterations for the Runge–Kutta equations arising when solving the remaining IVPs. For the perturbed two-body problem of orbital mechanics, this approach was found to significantly reduce the computational cost of uncertainty propagation, measured by the number of force-model evaluations for a given accuracy (Aristoff et al. 2012, 2014).

3 Methods

A brief description of the use of Runge–Kutta methods for orbit and uncertainty propagation will now be given, followed by an overview of VGL-IRK. Additional details can be found in Aristoff and Poore (2012) and Aristoff et al. (2014).

The equations of motion describing an object in space take the following form:

$$\mathbf{Y}''(t) = \mathbf{g}\left(t, \mathbf{Y}(t), \mathbf{Y}'(t)\right), \quad \mathbf{Y}(t_0) = \mathbf{Y}_0, \quad \mathbf{Y}'(t_0) = \mathbf{Y}'_0. \tag{1}$$

The independent variable t (time) is permitted to take on any real value, the dependent variable $\mathbf{Y}$ (the state of the object) is a vector-valued function, and the prime symbol denotes the time derivative. If the IVP (1) contains parameters whose values are unknown, these parameters can be estimated via state augmentation, resulting in an additional evolution equation for each parameter.

Runge–Kutta methods may be used to solve the IVP given by (1), that is, to find the state of the object at a later (or earlier) time (Iserles 2004; Butcher 2008; Hairer et al. 2009; Hairer and Wanner 2010). An s-stage Runge–Kutta method is defined by its weights $\mathbf{b} = (b_1, b_2, \ldots, b_s)$, nodes $\mathbf{c} = (c_1, c_2, \ldots, c_s)$ and s by s integration matrix $\mathbf{A}$ whose elements are a_{ij}. It can be used to solve the IVP over a given time step t_0 to $t_0 + h$ by writing

$$\begin{aligned} \boldsymbol{\xi}_i &= \mathbf{Y}'_0 + h \sum_{j=1}^{s} a_{ij} \boldsymbol{\xi}'_j \\ \boldsymbol{\xi}'_i &= \mathbf{g}\left(t_0 + c_i h, \mathbf{Y}_0 + h \sum_{j=1}^{s} a_{ij} \boldsymbol{\xi}_j, \mathbf{Y}'_0 + h \sum_{j=1}^{s} a_{ij} \boldsymbol{\xi}'_j\right) \end{aligned} \quad i = 1, 2, \ldots, s, \tag{2}$$

where $(\boldsymbol{\xi}_i, \boldsymbol{\xi}'_i)$ are the internal stages, that encode the state of the object at the intermediate times $t_0 + c_i h$ for $i = 1, 2, \ldots, s$. Once obtained, the internal stages can be used to advance the solution from t_0 to $t_0 + h$ according to

$$\mathbf{Y}_1 = \mathbf{Y}_0 + h \sum_{i=1}^{s} b_i \boldsymbol{\xi}_i, \qquad \mathbf{Y}'_1 = \mathbf{Y}'_0 + h \sum_{i=1}^{s} b_i \boldsymbol{\xi}'_i. \tag{3}$$

If $a_{ij} = 0$ for $i < j$, then (2) is an explicit expression for the internal stages. Otherwise, as with implicit Runge–Kutta methods, iterative techniques must be used to solve (2), as dis-

[5] The $2n+1$ states arising within the UKF are often referred to as sigma points.

cussed in Sect. 2, and unperturbed Keplerian dynamics or higher-fidelity analytical approximations can be used as a guess to warm-start the iterations. We note that a more efficient implementation of (2)–(3) can be made by substituting the first formula of (2) into the second to yield

$$\boldsymbol{\xi}'_i = \mathbf{g}\left(t_0 + c_i h, \mathbf{Y}_0 + \bar{c}_i h \mathbf{Y}'_0 + h^2 \sum_{j=1}^{s} \bar{a}_{ij} \boldsymbol{\xi}'_j, \mathbf{Y}'_0 + h \sum_{j=1}^{s} a_{ij} \boldsymbol{\xi}'_j\right), \tag{4}$$

and by substituting the first formula of (2) into (3) to yield

$$\mathbf{Y}_1 = \mathbf{Y}_0 + h\mathbf{Y}'_0 + h^2 \sum_{i=1}^{s} \bar{b}_i \boldsymbol{\xi}'_i, \quad \text{and} \quad \mathbf{Y}'_1 = \mathbf{Y}'_0 + h \sum_{i=1}^{s} b_i \boldsymbol{\xi}'_i. \tag{5}$$

where

$$\bar{a}_{ij} = \sum_{k=1}^{s} a_{ik} a_{kj}, \qquad \bar{b}_i = \sum_{j=1}^{s} b_j a_{ji}, \quad \text{and} \quad \bar{c}_i = \sum_{j=1}^{s} a_{ij}. \tag{6}$$

This second-order formulation of Runge–Kutta is analogous to the second-order formulation of MCPI presented by Bai and Junkins (2011). It is also known as double integration because the position is given directly from the acceleration.

An s-stage Gauss–Legendre IRK method can be constructed by (1) letting $c_1, c_2, \ldots, c_s$ be the s zeros of the s-order Legendre polynomial and (2) computing

$$a_{ij} = \int_{-1}^{c_i} \mathscr{L}_j(\tau) d\tau, \quad i, j = 1, 2, \ldots, s, \tag{7}$$

and

$$b_i = \int_{-1}^{1} \mathscr{L}_i(\tau) d\tau, \quad i = 1, 2, \ldots, s, \tag{8}$$

where

$$\mathscr{L}_i(\tau) = \prod_{\ell=1,\ \ell \neq i}^{s} \frac{\tau - c_\ell}{c_i - c_\ell} \tag{9}$$

is the Lagrange interpolating polynomial. An s-stage Gauss–Legendre implicit Runge–Kutta method has order $2s$, which is the highest order any s-stage Runge–Kutta method can achieve (Butcher 2008; Hairer et al. 2009). Hence, the difference between the approximate numerical solution $(\tilde{\mathbf{Y}}, \tilde{\mathbf{Y}}')$ and the exact solution $(\mathbf{Y}, \mathbf{Y}')$ is given by

$$||\tilde{\mathbf{Y}} - \mathbf{Y}|| = O(h^{2s+2}) \qquad \text{and} \qquad ||\tilde{\mathbf{Y}}' - \mathbf{Y}'|| = O(h^{2s+1}) \tag{10}$$

as $h \to 0$. This asymptotic result can be used to estimate the numerical truncation error and to adjust the step size after an accepted (or rejected) time step, so that the next time step taken is near optimal.

Implementation of a variable-step Gauss–Legendre implicit Runge–Kutta (VGL-IRK) method requires a number of careful considerations, details of which are presented in Aristoff and Poore (2012) and in a companion paper Aristoff et al. (2014). A high-level overview of VGL-IRK is given in Figs. 1 and 2, which describe propagation of the first state and of the remaining states, respectively.

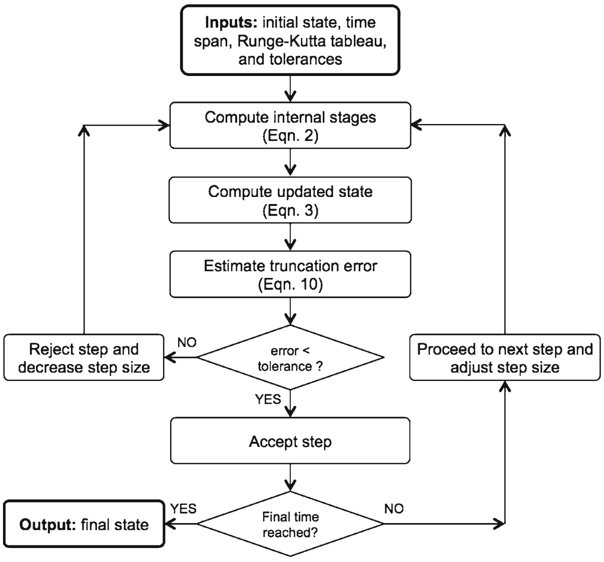
Inputs: initial state, time span, Runge-Kutta tableau, and tolerances
Compute internal stages (Eqn. 2)
Compute updated state (Eqn. 3)
Estimate truncation error (Eqn. 10)
error < tolerance ?
NO
Reject step and decrease step size
YES
Accept step
Final time reached?
YES
Output: final state
NO
Proceed to next step and adjust step size

Fig. 1 High-level overview of VGL-IRK: propagation of the first (or only) state

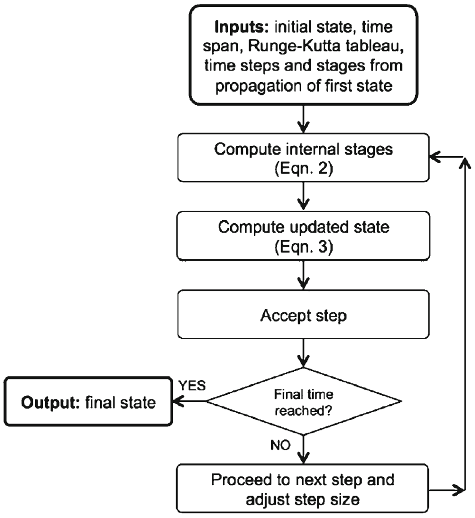
Inputs: initial state, time span, Runge-Kutta tableau, time steps and stages from propagation of first state
Compute internal stages (Eqn. 2)
Compute updated state (Eqn. 3)
Accept step
Final time reached?
YES
Output: final state
NO
Proceed to next step and adjust step size

Fig. 2 High-level overview of VGL-IRK: propagation of the remaining states

4 Results

In what follows, a rigorous comparison is made between VGL-IRK and DP8 for precise orbit and uncertainty propagation. Specifically, we compare the efficiency (runtime versus accuracy) of our C++ implementation of VGL-IRK to that of a professional C++ implementation of DP8 (Brankin et al. 1991). Multiple orbital regimes are considered, as are high-fidelity gravitational models. A direct comparison of efficiency is now possible because (1) each method uses the same set of force models, (2) each method is written in a compiled language, (3) each method is timed on the same computer, and (4) each method uses the same compiler and compilation flags. We note that DP8 is more efficient than

Table 2 Initial (Keplerian) orbital elements for the orbit propagation scenarios A, B, and C [taken from Jones (2012)]

Scenario	Orbit type	a (km)	e	$i\,(^\circ)$	$\Omega\,(^\circ)$	$\omega\,(^\circ)$	$M\,(^\circ)$	Orbits
A	LEO	6745.592	0.01	7.81	100.21	152.83	0.0	3
B	GEO	42164.0	0.001	1.0	145.92	266.13	0.0	3
C	Molniya	26553.0	0.74	63.4	330.21	270.0	0.0	3

lower-order Dormand–Prince methods [e.g., Dormand–Prince 5(4)] when used for precise orbit and uncertainty propagation.

We will also consider the performance of VGL-s and MCPI for orbit and uncertainty propagation. However, methods without variable-step error control (e.g., MCPI) cannot be compared fairly to methods with variable-step error control (e.g., VGL-IRK, VGL-s, DP8), since the former require a-priori unknown knowledge of the step size needed to achieve the desired tolerance. This is a significant disadvantage from an operational standpoint that would not appear in performance tests. Nevertheless, we have implemented a fixed-step version of MCPI, and measured the number of function evaluations needed to obtain a prescribed accuracy (after first tuning the algorithm to achieve optimal performance for said accuracy) when used in conjunction with high-fidelity force models and realistic integration tolerances.

4.1 Comparison to results published in the literature

We begin by evaluating the performance of the propagators given in Table 1 using the scenarios listed in Table 2. Specifically, we measure the number of force-model evaluations necessary to achieve centimeter accuracy after three orbital periods for an object in LEO, an object in GEO, and an object in highly-elliptic (Molniya) orbit. A degree and order 70 Earth gravity model (EGM2008), together with analytically-derived solar and lunar gravitational perturbations (Montenbruck and Gill 2000), is used to model the forces. These particular scenarios were chosen so that direct comparison can be made to the variable-step implementation of Gauss–Legendre implicit Runge–Kutta developed by Jones (2012), coined VGL-s. A direct comparison to the fixed-step implementation of MCPI developed by Bai and Junkins (2011) is not possible because the authors did not present the number of force-model evaluations nor did they use high-fidelity gravity models. It was therefore necessary to develop an MCPI-based propagator in-house following Bai and Junkins (2011). Upon completion, the algorithm was validated (using the aforementioned force model) against an independent implementation of MCPI by Koblick (2012).[6] For each of the scenarios in Table 2, we varied the step size and number of nodes within MCPI in order to determine the optimal combination of these tuning parameters. This was necessary because MCPI as reported in the literature is a fixed-step method and unlike DP8, VGL-IRK, and VGL-s it does not adaptively adjust the step size to meet a local accuracy requirement. We note that the solution to the unperturbed two-body problem is used to warm-start the iterations arising in VGL-s, MCPI, and VGL-IRK. This enhancement was not included in the original implementation of MCPI, but we have included it here because doing so reduces the number of iterations required for convergence. Truth is generated using a high-accuracy (fourteen digits of accuracy per time step), high-order VGL-IRK method.

[6] We observe a reduction in the number of force-model evaluations in our second-order implementation of MCPI (analogous to the second-order implementation of IRK discussed in Sect. 3) compared to B. Koblick's first-order implementation of MCPI.

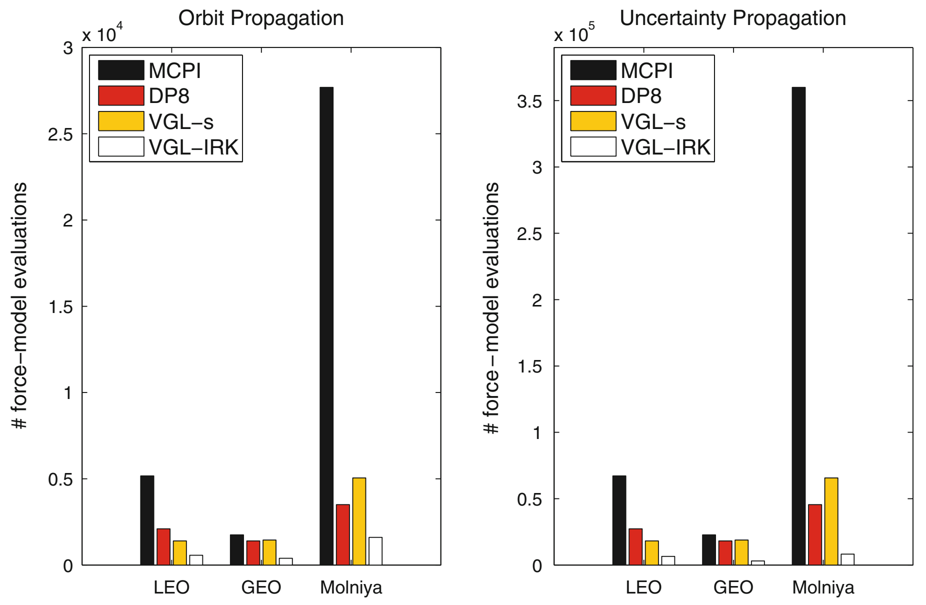

Fig. 3 Computational cost of orbit and uncertainty propagation for the scenarios given in Table 2 using the propagators listed in Table 1. (*Left*) Work required to propagate a single orbital state to within one centimeter from truth for an object in LEO, GEO, and highly-elliptic (Molniya) orbit. (*Right*) Work required to propagate thirteen orbital states (arising within the prediction step of the UKF) to within an average of one centimeter from truth. A serial computing environment is assumed. In a parallel computing environment, the number of force-model evaluations per processor/core could be substantially reduced when using MCPI, VGL-s, and VGL-IRK, but not when using DP8

The work required to propagate a single position-velocity orbital state is shown in Fig. 3(left). A serial computing environment is assumed. A number of comments can be made. First, as expected, the work required to propagate an object in highly-elliptic (Molniya) orbit exceeds the work required to propagate an object in LEO, which in turn exceeds the work required to propagate on object in GEO. Second, despite using an optimal combination of tuning parameters, MCPI is less efficient than the other methods. This is particularly evident for the Moniya object in highly-elliptic orbit, in which the variable-step methods are 5 to 17 times faster than MCPI for the same accuracy. Although the performance of MCPI (as well as VGL-IRK and VGL-s) could be further enhanced by distributing the force-model evaluations among parallel processors, there are inherent limitations to this approach (see Sect. 5). Third, we observe that in all three scenarios, VGL-IRK requires fewer force-model evaluations than each of the other methods, even for the highly-elliptic orbits. An alternative implementation of Gauss–Legendre IRK, VGL-s, requires roughly three times the number of force-model evaluations compared to that of VGL-IRK for the propagation of a single orbit.

Since the orbital state under consideration is six-dimensional, (orbit and) uncertainty propagation via the UKF requires the propagation of 13 orbits (or sigma points). The work required to propagate these sigma points to within an average of one centimeter accuracy is shown in Fig. 3(right). For each scenario, the state covariance is representative of a high-accuracy orbit belonging to an object in the space catalog (Aristoff et al. 2014). Because VGL-IRK is able to exploit the proximity of nearby orbits (Aristoff et al. 2012, 2014), an additional reduction in computational cost is achieved relative to the other methods. In other words, uncertainty propagation does not cost 13 times that of a single orbit propagation (as

Table 3 Initial (Keplerian) orbital elements for the uncertainty propagation scenarios, some of which are taken from Vinti (1998), others from CelesTrak (Kelso 2013)

Scenario	Orbit type	a (km)	e	$i\,(^\circ)$	$\Omega\,(^\circ)$	$\omega\,(^\circ)$	$M\,(^\circ)$	Orbits
1S	LEO	6640	0.0095	72.9	116	57.7	105	3
1L	LEO	6640	0.0095	72.9	116	57.7	105	30
2S	LEO	7878	0	30.0	137	0	36.0	3
2L	LEO	7878	0	30.0	137	0	36.0	30
3S	MEO	25508	0.0023	65.9	358	343	107	3
3L	MEO	25508	0.0023	65.9	358	343	107	30
4S	GEO	42164	0	0	0	0	250	3
4L	GEO	42164	0	0	0	0	250	30
5S	GEO	42164	0.0005	14	18	333	26.5	3
5L	GEO	42164	0.0005	14	18	333	26.5	30
6S	Molniya	26628	0.7416	63.4	120	261	144	3
6L	Molniya	26628	0.7416	63.4	120	261	144	30

Both short and long propagations are considered, as are nearly-circular and highly-elliptic orbits

it does for MCPI, DP8, and VGL-s). This is particularly evident for the Molniya object in highly-elliptic orbit, in which uncertainty propagation via VGL-IRK can be achieved at a fraction of the cost. Specifically, VGL-IRK is roughly 45 times faster than MCPI, 8 times faster than VGL-s, and 6 times faster than DP8 when propagating the uncertainty of a Molniya object in highly-elliptic orbit.

4.2 Performance of DP8 and VGL-IRK: runtime analysis

Next, we present a series of runtime comparisons between Dormand–Prince 8(7) and VGL-IRK methods used for orbit and uncertainty propagation. Specifically, we construct work-accuracy plots for VGL-IRK and DP8. We also consider the performance of VGL-IRK wherein the orbital states are propagated individually, rather than collectively, in order to show the benefit of collective propagation. The tests are run on a 2.6 GHz single threaded process on Ubuntu 12.04 (64-bit). The algorithms (as well as the force models) are written in C++ and compiled using GCC version 4.6.3 with the –O2 optimization flag. Neither MCPI nor VGL-s are included in this suite of tests because we do not have a C++ implementation of the version of these methods reported in the literature.

Twelve realistic uncertainty propagation scenarios are used to quantify the performance of VGL-IRK and DP8 (see Table 3). These include scenarios with objects in low-altitude LEO, high-altitude LEO, medium Earth orbit (MEO), non-inclined geostationary orbit (GEO), inclined GEO, and highly-elliptic orbit. Both short (3 orbital periods) and long (30 orbital periods) propagations are considered. All propagations use the J2000 epoch. We present the cost of uncertainty propagation via the UKF in which 13 sigma points (or states) are used to represent uncertainty (of a six-dimensional Gaussian state). The mean sigma point is given in Table 3. The other twelve sigma points are perturbed from the mean, and characterize the uncertainty in the orbital state (the state covariance is representative of a high-accuracy orbit belonging to an object in the space catalog). The following truncation levels for EGM2008 are used: 90 for LEO, 20 for MEO, and 12 for GEO (Petit and Luzum 2010). A degree and order 90 gravity model is used for the Molniya object. Analytically-derived solar and lunar perturbations are also included in the force model (Montenbruck and Gill 2000). Truth is generated using a high-accuracy (fourteen digits of accuracy per time step), high-order VGL-IRK method. Note that both DP8 and VGL-IRK are variable-step propagators that estimate and control the integration error based on a user-provided error tolerance.

We present in Figs. 4 and 5 the relationship between the global root-mean-square integration error in position, averaged over the 13 states, and the computational cost (runtime) in a serial computing environment, for each of the twelve scenarios. The runtime is an average over 10 trials. The local integration tolerance is varied so that the global integration error is between 1 mm and 10 m for the short propagations, and between 1 cm and 100 m for the long propagations. An additional speedup would be observed for VGL-IRK in a parallel computing environment (see Sect. 5). The results are summarized in Table 4. Overall, VGL-IRK takes 70–91 % less time than DP8 when propagating orbits collectively (70–89 % for nearly-circular orbits, 74–91 % for highly-elliptic orbits). Collective propagation of the sigma points is particularly advantageous when propagating highly-elliptic orbits, as evidenced by Fig. 5e, f.

5 Discussion

The above results were obtained in a serial computing environment. In a parallel computing environment, evaluation of the force model (which tends to be expensive) could be done independently and in parallel with VGL-IRK, VGL-s, or MCPI. Over a given time step, the number of force-model evaluations is proportional to the number of nodes (or stages) in the method. It is therefore tempting to dedicate available processors to the task of evaluating the force model and to use high-order methods with a large number of nodes. However, there are two issues with this approach. First, as the number of nodes is increased, the order of the method increases, and larger time steps can and must to be taken in order to maintain efficiency (since more work is done per step). However, time steps which are too large (e.g., on the order of an orbital period or more) will lead to inefficient or imprecise integration [and possibly numerical instability (Aristoff et al. 2014)], because the work devoted to each step and the accuracy achieved in the step becomes unbalanced. Large steps are also discouraged because when a step is rejected, the amount of work that is wasted is proportional to the step size. Larger steps also mean fewer chances for the step size to be adjusted. Hence, if the initial time step is too small and the duration of the propagation is short, then the propagated state may be much more accurate than what the user requested, and therefore too much work was devoted to the propagation. The second issue is that rarely does one need to propagate only a single orbit (trajectory). When multiple orbits need to be propagated (e.g., for uncertainty propagation), each orbit propagation could be done in parallel before the force-model evaluations themselves are parallelized. Moreover, when multiple uncertainties need to be propagated (e.g., for scoring different association hypotheses within a tracking system), each uncertainty propagation could be done in parallel before the orbit propagations themselves are parallelized. Parallelization of higher-level functions should precede parallelization of the lower-level force-model evaluations, in part because the level of communication between parallel force-model evaluations would be non-negligible and lead to slower propagations.

As previously mentioned, Chebyshev–Picard iteration is mathematically equivalent to Gauss–Chebyshev (Lobatto) IRK wherein fixed-point (Picard) iteration is used to solve the nonlinear system of equations that arise on each time step. One could therefore incorporate variable-step error estimation and control within an implementation of MCPI in order to significantly improve its performance and make the method self-tuning. However, Gauss–Legendre IRK remains the preferred IRK approach because the scheme is both superconvergent and symplectic (Gauss–Chebyshev–Lobatto schemes are neither superconvergent

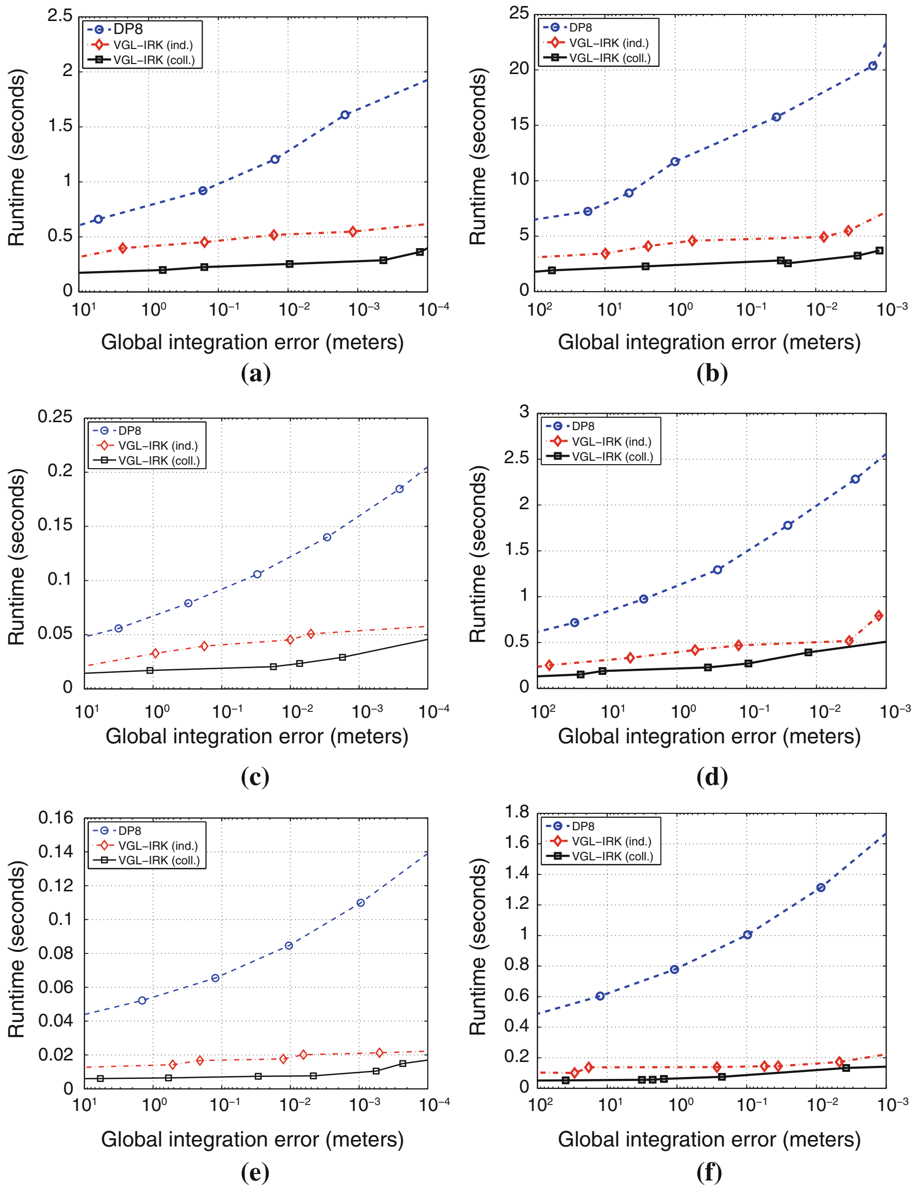

Fig. 4 Work-accuracy plots for Scenarios 1–3 (see Table 3) using a variable-step Gauss–Legendre-implicit-Runge–Kutta-based uncertainty propagator (VGL-IRK) and a variable-step Dormand–Prince 8(7)-based uncertainty propagator (DP8). The result for both collective and individual propagation of the orbits within VGL-IRK is shown. Note that DP8 is an explicit method and must therefore propagate orbits individually. A serial computing environment is assumed. An additional speed-up would be observed for VGL-IRK in a parallel computing environment. **a** Scenario 1S: LEO (low-altitude). **b** Scenario 1L: LEO (low-altitude). **c** Scenario 2S: LEO (high-altitude). **d** Scenario 2L: LEO (high-altitude). **e** Scenario 3S: MEO. **f** Scenario 3L: MEO

nor symplectic). Superconvergent methods achieve the highest possible convergence rate (i.e., an s-stage method achieves order $2s$). Symplectic methods are better suited for long-time propagations (because the Hamiltonian will be conserved) as well as propagations involving

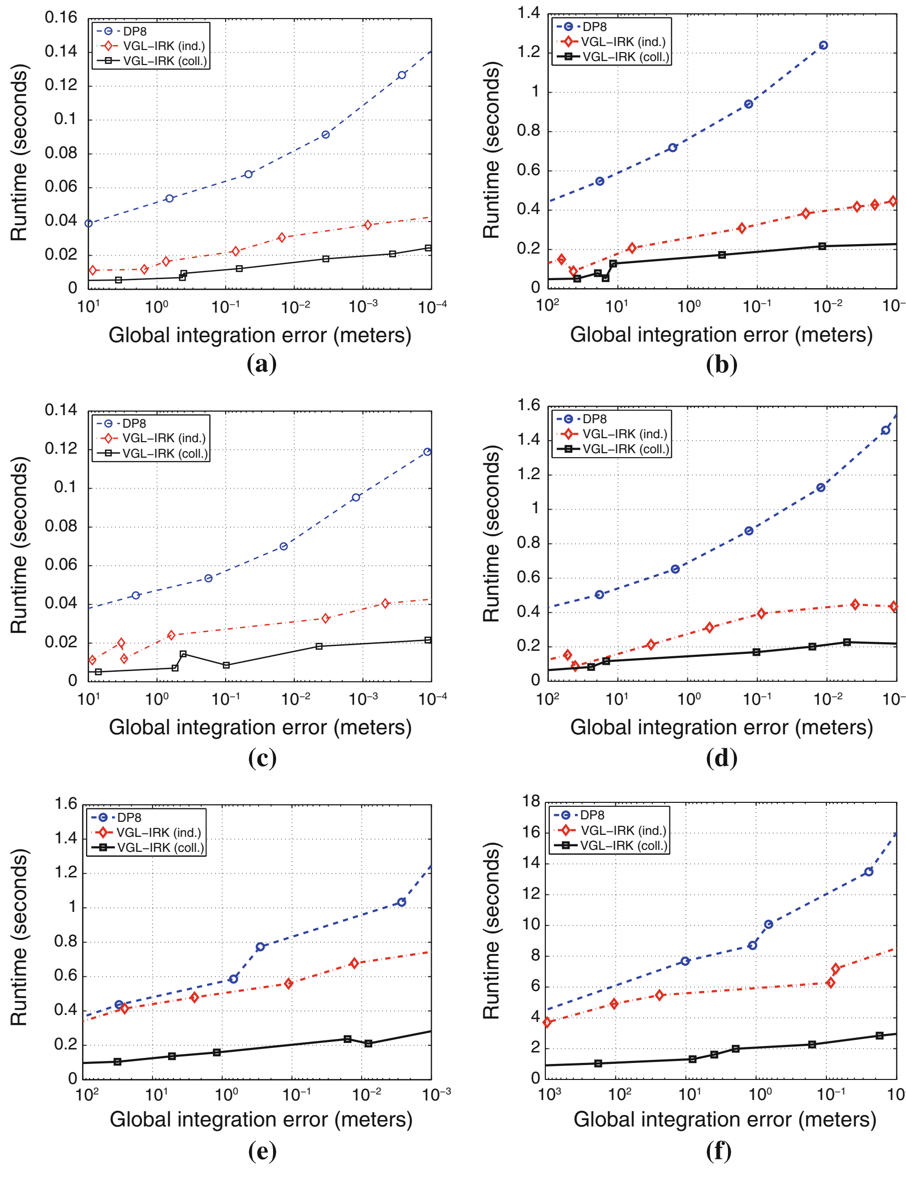

Fig. 5 Work-accuracy plots for Scenarios 4–6 in (see Table 3) using a variable-step Gauss–Legendre-implicit-Runge–Kutta-based uncertainty propagator (VGL-IRK) and a variable-step Dormand–Prince 8(7)-based uncertainty propagator (DP8). The result for both collective and individual propagation of the orbits within VGL-IRK is shown. Note that DP8 is an explicit method and must therefore propagate orbits individually. A serial computing environment is assumed. An additional speed-up would be observed for VGL-IRK in a parallel computing environment. **a** Scenario 4S: GEO (non-inclined). **b** Scenario 4L: GEO (non-inclined). **c** Scenario 5S: GEO (inclined). **d** Scenario 5L: GEO (inclined). **e** Scenario 6S: Molniya. **f** Scenario 6L: Molniya

the state of an object and its attitude (because the quaternion normalization constraint will be automatically satisfied). Although the underlying IRK scheme is symplectic, the present implementation of VGL-IRK is designed for short- to medium-time propagations. Some

Table 4 Summary of computational savings [relative to Dormand–Prince 8(7)] for the propagation of uncertainty in Scenarios 1–6 (see Table 3, Figs. 4 and 5) using VGL-IRK in a serial computing environment

	Scenario 1S (LEO)	Scenario 1L (LEO)	Scenario 2S (LEO)	Scenario 2L (LEO)	Scenario 3S (MEO)	Scenario 3L (MEO)
Savings (%)	72–80	72–80	70–77	79–81	86–87	90–91
	Scenario 4S (GEO)	Scenario 4L (GEO)	Scenario 5S (GEO)	Scenario 5L (GEO)	Scenario 6S (Molniya)	Scenario 6L (Molniya)
Savings (%)	85–86	82–89	82–87	84–86	74–81	81–82

The computational savings tend to increase as the accuracy of the propagation increases. Further computational savings would be achieved in a parallel computing environment

changes to the implementation would be necessary to ensure high accuracy for very-long-time propagations, wherein round-off errors may dominate truncation errors (Hairer et al. 2007, 2010; Farrés et al. 2013). This is the subject of ongoing work.

6 Conclusions

A variable-step Gauss–Legendre IRK (VGL-IRK) method has been developed specifically for orbit and uncertainty propagation. The VGL-IRK method has a number of desirable numerical properties. It is A-stable, parallelizable, symmetric, symplectic, and superconvergent (see Table 1). The method is adaptive in that the user only needs to specify the requested accuracy to be achieved. This feature removes the guess-work needed to select a step size and leads to a more efficient propagation, especially when dealing with objects in highly-elliptic orbits. Finally, the VGL-IRK method propagates nearby trajectories collectively, rather than individually, which significantly reduces the computational cost of uncertainty propagation without sacrificing accuracy.

The performance of Gauss–Legendre-, Dormand–Prince-, and Chebyshev–Picard-based propagators was analyzed within a realistic simulation environment using high-fidelity gravity models and a range of realistic integration (error) tolerances. Both nearly-circular and highly-elliptic orbits were considered, as was both orbit and uncertainty propagation. We demonstrated that the variable-step implementation of Gauss–Legendre IRK (VGL-IRK) described above and outlined in Sect. 3 is up to eleven times faster than a professional implementation of Dormand–Prince 8(7) (Brankin et al. 1991) for the same accuracy, even before VGL-IRK is potentially parallelized, and three to eight times faster than an alternative implementation of Gauss–Legendre IRK, VGL-s (Jones 2012). Moreover, VGL-IRK was shown to be 8 to 45 times more efficient than our implementation of MCPI (Bai and Junkins 2011), even after MCPI had been optimally tuned for a prescribed accuracy. Because VGL-IRK is fully-adaptive and substantially reduces the cost of uncertainty propagation in multiple regimes of space (or equivalently, increases the accuracy of uncertainty propagation for the same cost), the method promises to have a significant impact on SSA functions that rely on uncertainty propagation such as tracking, uncorrelated track resolution, and conjunction analysis.

Acknowledgments The authors thank Alex Mont and David Beach for their assistance with the C++ software implementation. This work was funded, in part, by a Phase II STTR from the Air Force Office of Scientific Research (FA9550-12-C-0034) and a grant from the Air Force Office of Scientific Research (FA9550-11-1-0248).

References

Aristoff, J.M., Poore, A.B.: Implicit Runge–Kutta methods for orbit propagation. In: Proceedings of the 2012 AIAA/AAS Astrodynamics Specialist Conference, AIAA 2012-4880, pp. 1–19. Minneapolis, MN (2012)

Aristoff, J.M., Horwood, J.T., Poore, A.B.: Implicit Runge–Kutta methods for uncertainty propagation. In: Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference, Wailea, HI (2012)

Aristoff, J.M., Horwood, J.T., Poore, A.B.: Implicit Runge–Kutta-based methods for fast, precise, and scalable uncertainty propagation. Under Review (2014)

Bai, X., Junkins, J.L.: Modified Chebyshev–Picard iteration methods for orbit propagation. J. Astronaut. Sci. **3**, 1–27 (2011)

Berry, M.M., Healy, L.M.: The generalized Sundman transformation for propagation of high-eccentricity elliptical orbits. In: Proceedings of the 12th AAS/AIAA Space Flight Mechanics Meeting, San Antonio, TX (2002). Paper AAS-02-109

Berry, M.M., Healy, L.M.: Implementation of Gauss–Jackson integration for orbit propagation. J. Astronaut. Sci. **52**(3), 331–357 (2004)

Beylkin, G., Sandberg, K.: ODE Solvers using bandlimited approximations. arXiv:1208.3285v1 [math.NA] (2012)

Bradley, B.K., Jones, B.A., Beylkin, G., Axelrad, P.: A new numerical integration technique in astrodynamics. In: Proceedings of the 22nd Annual AAS/AIAA Spaceflight Mechanics Meeting, pp. 1–20, Charleston, SC (2012). AAS 12–216

Brankin, R.W., Gladwell, I., Shampine, L.F.: RKsuite Release 1.0. http://www.netlib.org/ode/rksuite/rksuitec++.zip (1991)

Butcher, J.C.: Numerical Methods for Ordinary Differential Equations. Wiley, West Sussex (2008)

Farrés, A., Laskar, J., Blanes, S., Casas, F., Makazaga, J., Murua, A.: High precision symplectic integrators for the solar system, Celest. Mech. Dyn. Astr. **116**, 141–174 (2013)

Hairer, E., Wanner, G.: Solving ordinary differential equations II: stiff and differential-algebraic problems. In: Springer Series in Computational Mathematics, 2nd edn. Springer (2010)

Hairer, E., McLachlan, R.I., Razakarivony, A.: Achieving Brouwer's law with implicit Runge–Kutta methods. BIT Numer. Math. **48**, 231–243 (2007)

Hairer, E., Norsett, S.P., Wanner, G.: Solving ordinary differential equations I: nonstiff problems. In: Springer Series in Computational Mathematics, 2nd edn. Springer (2009)

Hairer, E., Lubich, C., Wanner, G.: Geometric numerical integration: structure-preserving algorithms for ordinary differential equations. In: Springer Series in Computational Mathematics. Springer (2010)

Hulme, B.L.: One-step piecewise polynomial Galerkin methods for initial value problems. Math. Comput. **26**(118), 415–426 (1972)

Iserles, A.: A First Course in the Numerical Analysis of Differential Equations. Cambridge University Press, Cambridge (2004)

Jones, B.A.: Orbit propagation using Gauss-Legendre collocation. In: Proceedings of the 2012 AIAA/AAS Astrodynamics Specialist Conference, AIAA 2012-4967, pp. 1–16. Minneapolis, MN (2012)

Jones, B.A., Anderson, R.L.: A survey of symplectic and collocation integration methods for orbit propagation. In: Proceedings of the 22nd Annual AAS/AIAA Spaceflight Mechanics Meeting, pp. 1–20, Charleston, SC (2012). AAS 12-214.

Julier, S.J., Uhlmann, J.K.: Unscented filtering and nonlinear estimation. Proc. IEEE **92**, 401–422 (2004)

Kelso, T.S.: Celestrak. http://www.celestrak.com (2013)

Koblick, D.: Vectorized Picard–Chebyshev Method. http://www.mathworks.com/matlabcentral/fileexchange/36940 (2012)

Montenbruck, O.: Numerical integration methods for orbital motion. Celest. Mech. Dyn. Astr. **53**, 59–69 (1992)

Montenbruck, O., Gill, E.: Satellite Orbits: Models, Methods, and Applications. Springer, Berlin (2000)

Nielsen, P.D., Alfriend, K.T., Bloomfield, M.J., Emmert, J.T., Miller, J.G., Guo, Y., et al.: Continuing Kepler's Quest: Assessing Air Force Space Command's Astrodynamics Standards. National Academies Press, Washington, DC (2012)

Petit, G., Luzum, B.: IERS Conventions (2010). Technical Report, International Earth Rotation and Reference Systems Service (2010)

Shampine, L.F.: Numerical Solution of Ordinary Differential Equations. Chapman and Hall, London (1994)

Shampine, L.F.: Error estimation and control for ODEs. J. Sci. Comput. **25**, 3–16 (2005)

van der Houwen, P.J., Sommeijer, D.P.: Parallel iteration of high-order Runge–Kutta methods with stepsize control. J. Comput. Appl. Math. **29**(1), 111–127 (1990)

Vinti, J.P.: Orbital and celestial mechanics. In: Der, G.J., Bonavito, N.L. (eds.) Progress in Astronautics and Aeronautics, vol. 177. American Institute of Aeronautics and Astronautics, Cambridge, MA (1998)

Wright, K.: Some relationships between implicit Runge–Kutta, collocation and Lanczos methods and their stability properties. BIT **10**, 217–227 (1970)