AAS 04-238

A Variable-Step Double-Integration Multi-Step Integrator

Matthew M. Berry[*] **Liam M. Healy**[†]

Abstract

A variable-step double-integration multi-step integrator is derived using divided differences. The derivation is based upon the derivation of Shampine-Gordon, a single-integration method. Variable-step integrators are useful for propagating elliptical orbits, because larger steps can be taken near apogee. As a double-integration method, the integrator performs only one function evaluation per step, whereas Shampine-Gordon requires two evaluations per step, giving the integrator a significant speed advantage over Shampine-Gordon. Though several implementation issues remain, preliminary results show the integrator to be effective.

INTRODUCTION

Use of numerical integration in space surveillance has grown in recent years as accuracy requirements have increased. Numerical integration requires a great deal of computation time compared to the analytic propagators previously used. An upgrade planned for the Navy's Space Surveillance System (known as the Fence) will greatly increase the number of objects being tracked, and hence significantly increase the amount of computation time required. Numerical integration methods requiring less computation time than those currently employed while maintaining or improving accuracy requirements are needed to reduce this burden.

The Gauss-Jackson method (Ref. 1) is frequently used for orbit propagation in space surveillance applications. It is a predictor-corrector, multi-step integrator. Multi-step integrators have a considerable advantage over single-step integrators (such as Runge-Kutta) because multi-step integrators can take larger step sizes to yield a given accuracy, and also have fewer evaluations per step. The Gauss-Jackson integrator performs double integration, meaning that the algorithm computes position directly from acceleration, without first integrating to find velocity. Because velocity is also needed to compute the state of the satellite and to compute atmospheric drag, Gauss-Jackson is generally implemented alongside an Adams integrator, which performs single integration. The Adams integrator is also a multi-step integrator, and can be implemented with Gauss-Jackson without requiring additional evaluations.

Because Gauss-Jackson is fixed step, it cannot efficiently handle highly elliptical orbits (Ref. 2). In order to achieve a given accuracy at perigee, more steps are taken at apogee than needed. One way to remedy this problem is with s-integration, which involves changing the independent variable using a generalized Sundman transformation (Ref. 3) to another variable, s. This transformation spreads the integration points more evenly about the orbit and can be considered an analytical step size adjustment based on advance knowledge of the orbit and not on error analysis. A disadvantage

[*]Graduate Assistant, Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, Virginia 24061. E-mail: maberry2@vt.edu.

[†]Research Physicist, Naval Research Laboratory, Code 8233, Washington, DC 20375-5355. E-mail: Liam.Healy@nrl.navy.mil.

is that a seventh differential equation must be solved to find time, which makes time subject to integration error. Another disadvantage is that although the size of a step in time space is changing, it is still a fixed step method in s space, so there is no control over the local error.

The Shampine-Gordon integrator (Ref. 4) is a variable-step, variable-order multi-step integrator. Shampine-Gordon integration is derived from divided differences, explained below, which allow the step size to be changed without a restart procedure. The predictor and corrector are of different order, which allows a local error estimate to be made at each step. If the local error is outside user-defined bounds, the step size is adjusted. Unlike Gauss-Jackson, where the coefficients are already known, the Shampine-Gordon coefficients are calculated at each step, depending on the step size and order.

Although the Shampine-Gordon integrator can efficiently handle elliptical orbits because it is variable step, it does have some disadvantages compared to Gauss-Jackson integration. It performs single integration, so acceleration must first be integrated to velocity, and then velocity is integrated to find position. Performing two single integrations causes more round-off error than double integration, and hence makes Shampine-Gordon more subject to instability. It must therefore take two evaluations per step to stay stable, whereas Gauss-Jackson only takes one evaluation per step. This extra evaluation significantly reduces the advantage in computation time gained by the variable-step method.

A double-integration variable-step method, proposed in this paper, needs only one evaluation per step to stay stable, and still has the advantage of a variable-step method.

MOTIVATION

Table 1: Summary of Integration Methods

Method	Single / Multi	Fixed / Variable	Non-Summed / Summed	Single / Double
Runge-Kutta	Single	Fixed	NA	Single
Runge-Kutta-Fehlberg	Single	Variable	NA	Single
Adams	Multi	Fixed	Non-Summed	Single
Summed Adams	Multi	Fixed	Summed	Single
Shampine-Gordon	Multi	Variable	Non-Summed	Single
Störmer-Cowell	Multi	Fixed	Non-Summed	Double
Gauss-Jackson	Multi	Fixed	Summed	Double
Proposed	Multi	Variable	Non-Summed	Double

A study of current numerical integrators motivates the need for a variable-step double-integration multi-step integrator. Numerical integrators may be classified into several categories. Integrators are either single-step or multi-step, which refers to the number of points that are used when integrating to the next point. Multi-step integrators are generally faster than single-step integrators, though there are some disadvantages to multi-step integration, such as the need for a start-up routine. Integrators may either have a fixed or a variable step size. Variable-step integrators reduce the number of steps needed at apogee, where the velocity is at a minimum, and so are generally more efficient for highly elliptical orbits. Multi-step integrators come in two forms, summed and non-summed, which refers to whether the integration is performed from epoch or step-by-step. Finally, integrators can perform either single or double integration. Single-integration integrators find velocity given acceleration, and position given velocity. Double-integration integrators find position directly from acceleration. Table 1 lists the features of several integrators commonly used in astrodynamics. Each integrator has one of two possibilities in the four categories. Comparing the integrators in Table 1 to one another shows the relative advantages and disadvantages of each category. The last line shows the integration method that we propose in this paper.

Variable Step

In Ref. 2, we presented a study of the speed and accuracy effects of variable-step integrators. In the study the fixed-step Gauss-Jackson integrator was compared to Shampine-Gordon, as well as Gauss-Jackson with s-integration, which provides an analytic step size control, though it does not provide error control. The study compared run-time of the integrators when they were set to give equivalent accuracy. Comparisons were made of orbits having various perigee heights and eccentricities. Figure 1 shows the results from the study at 400 km, which is typical of other perigee heights. The plot shows the speed ratio of the variable-step methods vs. eccentricity, where speed ratio is the run-time of the fixed-step Gauss-Jackson divided by the run-time of the variable-step methods. The variable-step methods have an advantage when the speed ratio is above one.

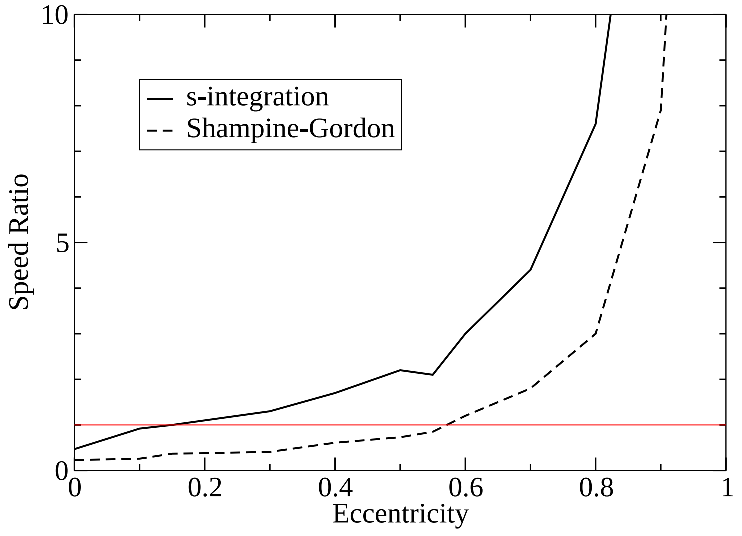

Figure 1: Speed ratios to fixed step integration at 400 km perigee.

Figure 1 shows that s-integration is more efficient than fixed-step integration at an eccentricity of approximately 0.15, and that Shampine-Gordon is more efficient than Gauss-Jackson at an eccentricity of approximately 0.60. Test results at other perigee heights show that the eccentricity where the variable-step methods are more efficient is independent of the perigee height. The plot also shows that s-integration is always more efficient than Shampine-Gordon. Shampine-Gordon requires two full evaluations per step, while s-integration only requires one full evaluation (Ref. 2). Though s-integration does provide a significant advantage for elliptical orbits, it still does not provide local error control, so there is no guarantee that it is meeting a specific accuracy requirement. Also, s-integration requires that a seventh differential equation be integrated to find time, which results in in-track error (Ref. 3). A variable-step integrator that only requires one evaluation per step would combine some advantages of both Shampine-Gordon and s-integration.

Double vs. Single Integration

Because two integrations are necessary, single integration has more round-off error than double integration. To examine the advantage of double integration over single integration, the double-integration Störmer-Cowell method is compared to the single-integration Adams method. These integrators are tested on test case orbits with varying eccentricity and perigee heights. The test cases all have an inclination of 40° and a ballistic coefficient of 0.01 $^{\mathrm{m}^2}/_{\mathrm{kg}}$. The epoch of the test cases is 1999-10-01 00:00:00 UTC. The tests are performed using the Special-K software suite (Ref. 5), developed by the Naval Research Laboratory, which is used operationally by Naval Network and Space Operations Command (NNSOC, formerly Naval Space Command).

In the tests a metric for integration accuracy is defined using an error ratio defined in terms of the RMS error of the integration (Ref. 6). First define position errors as

$$\Delta r = |r_{\text{computed}} - r_{\text{reference}}|. \tag{1}$$

The RMS position error can be calculated,

$$\Delta r_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\Delta r_i)^2}. \tag{2}$$

The RMS position error is normalized by the apogee distance and the number of orbits to find the position error ratio,

$$\rho_r = \frac{\Delta r_{\text{RMS}}}{r_A N_{\text{orbits}}}. \tag{3}$$

Table 2 gives error ratios over three days for the Störmer-Cowell and Adams integrators without perturbations. The table shows that the error ratio is always smaller with Störmer-Cowell. The reference value used to compute the error ratio in (1) is the analytic two-body solution. The integrators both use a 30 second step size in the tests, so have nearly identical run-times. The integrators are both set to use two evaluations per step. The first evaluation is a full evaluation, and the second evaluation is a "pseudo-evaluation," in which only the two-body force is re-evaluated, and the perturbation force from the first evaluation is added to it (Ref. 2).

Table 2: Error Ratios for Störmer-Cowell and Adams, Two Body

Height (km)	Eccentricity	Störmer-Cowell	Adams
300	0.00	2.47×10^{-13}	2.66×10^{-12}
300	0.25	3.05×10^{-12}	7.90×10^{-12}
300	0.50	1.28×10^{-11}	9.35×10^{-11}
300	0.75	4.01×10^{-11}	2.66×10^{-10}
500	0.00	3.49×10^{-13}	7.90×10^{-13}
500	0.25	2.87×10^{-12}	9.21×10^{-12}
500	0.50	7.94×10^{-12}	6.46×10^{-11}
500	0.75	2.21×10^{-11}	1.69×10^{-10}
1000	0.00	9.63×10^{-14}	4.78×10^{-12}
1000	0.25	3.53×10^{-13}	9.58×10^{-12}
1000	0.50	1.73×10^{-12}	2.40×10^{-11}
1000	0.75	9.70×10^{-12}	7.03×10^{-11}

Table 3 gives error ratios over three days for the two integrators with perturbations. The table shows that the error ratio is always smaller with Störmer-Cowell, except in the case of the 300 km circular orbit. The perturbations used are lunar and solar forces, the Jacchia 70 drag model, and the 36 × 36 WGS-84 geopotential. Again a step size of 30 seconds is used for both integrators, and two evaluations per step are performed, with the second one being a pseudo-evaluation. In these tests the reference used in (1) is found by integrating with the same integrator, but with half the step size. This step-size halving test has been shown to give a reasonable estimate of integration error (Ref. 7).

The results in Tables 2 and 3 show that Störmer-Cowell is generally more accurate than Adams for equivalent run-times. These results show the advantage of double integration over single integration.

Table 3: Error Ratios for Störmer-Cowell and Adams, Perturbations

Height (km)	Eccentricity	Störmer-Cowell	Adams
300	0.00	2.60×10^{-09}	1.40×10^{-09}
300	0.25	2.96×10^{-09}	6.50×10^{-09}
300	0.50	4.14×10^{-09}	1.35×10^{-08}
300	0.75	1.32×10^{-08}	4.18×10^{-08}
500	0.00	7.30×10^{-11}	2.47×10^{-10}
500	0.25	4.76×10^{-10}	1.39×10^{-09}
500	0.50	1.30×10^{-09}	3.82×10^{-09}
500	0.75	3.94×10^{-09}	1.12×10^{-08}
1000	0.00	3.58×10^{-12}	1.90×10^{-11}
1000	0.25	1.85×10^{-11}	5.75×10^{-11}
1000	0.50	5.58×10^{-11}	1.71×10^{-10}
1000	0.75	2.01×10^{-10}	6.29×10^{-10}

In the tests used to create these results, the integrators performed two evaluations per step. A further advantage of double integration is shown by removing the second evaluation. In a predict, evaluate, correct (PEC) implementation, the Adams method goes unstable in all of the test cases, with and without perturbations. The Störmer-Cowell method does not go unstable. Without the second evaluation Störmer-Cowell gives results that are only slightly less accurate than with two evaluations. This finding implies that double integration gives a stability advantage over single integration. This stability advantage allows a variable-step double-integration method to require only one evaluation per step, giving it a significant advantage over Shampine-Gordon.

SHAMPINE-GORDON INTEGRATION

Shampine-Gordon follows a predict, evaluate, correct, evaluate (PECE) implementation, so there are two evaluations at every step. The corrector is one order higher than the predictor, which allows for a local error estimate at each step. The step size and order are adjusted at every step based on the local error estimate. The derivation of Shampine-Gordon was originally presented in Ref. 4. That derivation is repeated here because it is the starting point of the derivation of the double-integration variable-step integrator. The derivation is based on the concept of divided differences.

Divided Differences

The Shampine-Gordon integrator is derived by integrating a polynomial which interpolates through the function values at the backpoints. This polynomial is written in divided difference form so that the backpoints do not have to be equally spaced.

The $(k-1)^{\text{th}}$ degree polynomial $P_{k,n}(x)$ interpolates through the k function values $f_{n-k+1} \dots f_n$ if (Ref. 4, p. 76)

$$P_{k,n}(x_{n+1-i}) = f_{n+1-i}, \quad i = 1 \dots k. \tag{4}$$

Here $f_i = f(x_i, y_i)$, where f is the right-hand side function in the differential equation

$$y' = f(x, y). \tag{5}$$

In divided difference form, the polynomial is written (Ref. 4, p. 77, (3))

$$\begin{aligned} P_{k,n}(x) = {} & f[x_n] + (x - x_n) f[x_n, x_{n-1}] + (x - x_n)(x - x_{n-1}) f[x_n, x_{n-1}, x_{n-2}] + \cdots \\ & + (x - x_n)(x - x_{n-1}) \cdots (x - x_{n-k+2}) f[x_n, x_{n-1}, \dots, x_{n-k+1}], \end{aligned} \tag{6}$$

where $f[x_n, \ldots, x_{n-i+1}]$ is the i^{th} divided difference of f_n. The divided differences are calculated through a recursive relation,

$$\begin{aligned} f[x_n] &= f_n, \\ f[x_n, x_{n-1}] &= \frac{f_n - f_{n-1}}{x_n - x_{n-1}}, \\ f[x_n, \ldots, x_{n-i}] &= \frac{f[x_n, \ldots, x_{n-i+1}] - f[x_{n-1}, \ldots, x_{n-i}]}{x_n - x_{n-i}}. \end{aligned} \tag{7}$$

As an example, consider the divided difference table in Table 4. Each divided difference is calculated by subtracting the two values to the left and dividing by the total span in x that the difference covers.

Table 4: Example Divided Difference Table

$$\begin{tikzcd}
n & x_n & f[x_n] & f[x_n, x_{n-1}] & f[x_n, x_{n-1}, x_{n-2}] & f[x_n, \ldots, x_{n-3}] \\
1 & 1 & 1 \arrow[dr] & & & \\
2 & 3 & 5 \arrow[r] \arrow[dr] & 2 \arrow[dr] & & \\
3 & 4 & 9 \arrow[r] \arrow[dr] & 4 \arrow[r] \arrow[dr] & 2/3 \arrow[dr] & \\
4 & 7 & 7 \arrow[r] & -2/3 \arrow[r] & -14/12 \arrow[r] & -11/36
\end{tikzcd}$$

For the example in Table 4, the polynomial $P_{4,4}$ that passes through the values is found from (6),

$$P_{4,4} = 7 + (x-7)(-2/3) + (x-7)(x-4)(-14/12) + (x-7)(x-4)(x-3)(-11/36). \tag{8}$$

The example can also be used to illustrate how a new point can be added to an existing polynomial. When $n = 3$, the polynomial $P_{3,3}$ is known,

$$P_{3,3} = 9 + (x-4)(4) + (x-4)(x-3)(2/3). \tag{9}$$

The polynomial $P_{4,4}$ can be found from $P_{3,3}$ by adding one more term which includes the new difference,

$$P_{4,4} = 9 + (x-4)(4) + (x-4)(x-3)(2/3) + (x-4)(x-3)(x-1)(-11/36). \tag{10}$$

The two forms of $P_{4,4}$, (8) and (10), are equivalent. This procedure is used in the derivation of the corrector, which uses a polynomial that passes through the predicted value as well as all the same values as the polynomial used by the predictor.

Predictor

The Shampine-Gordon predictor finds the predicted value, p_{n+1}, of the solution at point $(n+1)$ from the value at n, y_n, by integrating the interpolating polynomial $P_{k,n}$ defined in (6) (Ref. 4, p.

76, (2a)),

$$p_{n+1} = y_n + \int_{x_n}^{x_{n+1}} P_{k,n}(x)\,dx. \tag{11}$$

In order to develop an effective algorithm to integrate $P_{k,n}(x)$, the terms are rewritten. First, the independent variable x is replaced by s which measures the fraction of the current interval covered (Ref. 4, p. 77, (4)),

$$s = \frac{x - x_n}{h_{n+1}}, \tag{12}$$

with $h_n = x_n - x_{n-1}$ the separation between adjacent values of x. It is helpful to replace the divided differences with modified divided differences, ϕ, for which (Ref. 4, p. 77, (4))

$$\begin{aligned} \phi_1(n) &= f[x_n] = f_n, \\ \phi_i(n) &= \psi_1(n)\psi_2(n)\cdots\psi_{i-1}(n) f[x_n, x_{n-1}, \ldots, x_{n-i+1}] \qquad i > 1, \end{aligned} \tag{13}$$

where $\psi_i(n)$ is the size of the interval from the point i steps prior, that is, the sum of the i steps leading up to point n (Ref. 4, p. 77, (4)),

$$\psi_i(n) = h_n + h_{n-1} + \cdots + h_{n+1-i}. \tag{14}$$

The first term of the polynomial $P_{k,n}$, (6), is simply $f[x_n]$. For $i > 1$, the i^{th} term of $P_{k,n}(x)$ (Ref. 4, p. 78, (5)),

$$(x - x_n)(x - x_{n-1})\cdots(x - x_{n-i+2}) f[x_n, x_{n-1}, \ldots, x_{n-i+1}] \tag{15}$$

if $i > 1$, can be written in terms of s and $\phi_i(n)$ (Ref. 4, p. 78),

$$(sh_{n+1})(sh_{n+1} + h_n)\cdots(sh_{n+1} + h_n + \cdots + h_{n-i+3}) \frac{\phi_i(n)}{\psi_1(n)\psi_2(n)\ldots\psi_{i-1}(n)}. \tag{16}$$

Next, the term is multiplied and divided by $\psi_1(n+1)$ through $\psi_{i-1}(n+1)$ which allows the differences to be easily computed from one step to the next (Ref. 4, p. 78),

$$\begin{aligned} &\left(\frac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\frac{sh_{n+1} + h_n}{\psi_2(n+1)}\right)\cdots\left(\frac{sh_{n+1} + h_n + \cdots + h_{n-i+3}}{\psi_{i-1}(n+1)}\right) \\ &\qquad\qquad \times \frac{\psi_1(n+1)\psi_2(n+1)\cdots\psi_{i-1}(n+1)}{\psi_i(n)\psi_2(n)\cdots\psi_{i-1}(n)}\phi_i(n). \end{aligned} \tag{17}$$

This expression is simplified by introducing β (Ref. 4, p. 77, (4)),

$$\begin{aligned} \beta_1(n+1) &= 1, \\ \beta_i(n+1) &= \frac{\psi_1(n+1)\psi_2(n+1)\cdots\psi_{i-1}(n+1)}{\psi_1(n)\psi_2(n)\cdots\psi_{i-1}(n)} \qquad i > 1, \end{aligned} \tag{18}$$

and by condensing the sums of h in the numerators into ψ (Ref. 4, p. 78),

$$\left(\frac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\frac{sh_{n+1} + \psi_1(n)}{\psi_2(n+1)}\right)\cdots\left(\frac{sh_{n+1} + \psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right)\beta_i(n+1)\phi_i(n). \tag{19}$$

Noting that the $i = 1$ term of $P_{k,n}(x)$ is $f[x_n] = \phi_1(x_n)$, the i^{th} term (15) can be written (Ref. 4, p. 79, (6)),

$$c_{i,n}(s)\beta_i(n+1)\phi_i(n), \tag{20}$$

where $c_{i,n}(s)$ is defined (Ref. 4, p. 76, (6))

$$c_{i,n}(s) = \begin{cases} 1 & i = 1, \\ \left(\dfrac{sh_{n+1}}{\psi_1(n+1)}\right) = s & i = 2, \\ \left(\dfrac{sh_{n+1}}{\psi_1(n+1)}\right)\left(\dfrac{sh_{n+1}+\psi_1(n)}{\psi_2(n+1)}\right)\cdots\left(\dfrac{sh_{n+1}+\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) & i \geq 3. \end{cases} \tag{21}$$

In turn (20) can be written (Ref. 4, p. 79),

$$c_{i,n}(s)\phi_i^*(n) \tag{22}$$

by defining ϕ^* (Ref. 4, p. 79),

$$\phi_i^*(n) = \beta_i(n+1)\phi_i(n). \tag{23}$$

The interpolating polynomial $P_{k,n}(x)$ can now be written as a summation (Ref. 4, p. 79, (7)),

$$P_{k,n}(x) = \sum_{i=1}^{k} c_{i,n}(s)\phi_i^*(n). \tag{24}$$

Returning to the original problem of finding the predicted value, the polynomial in (11) can be replaced with (24),

$$p_{n+1} = y_n + \int_{x_n}^{x_{n+1}} \sum_{i=1}^{k} c_{i,n}(s)\phi_i^*(n)\,dx. \tag{25}$$

Since the $\phi_i^*(n)$ are constants, they can be pulled out of the integration,

$$p_{n+1} = y_n + \sum_{i=1}^{k} \phi_i^*(n) \int_{x_n}^{x_{n+1}} c_{i,n}(s)\,dx. \tag{26}$$

The integration variable can be changed to s by (12), noting that $s = 0$ when $x = x_n$, $s = 1$ when $x = x_{n+1}$, and $dx = h_{n+1}\,ds$ (Ref. 4, p. 79, (8)),

$$p_{n+1} = y_n + h_{n+1} \sum_{i=1}^{k} \phi_i^*(n) \int_0^1 c_{i,n}(s)\,ds. \tag{27}$$

To solve the problem, the integral of $c_{i,n}(s)$ is needed. First, the expression for $c_{i,n}(s)$ is simplified through a recursion formula,

$$c_{i,n}(s) = \left(\frac{sh_{n+1}+\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) c_{i-1,n}(s), \tag{28}$$

when $i \geq 3$. This expression is further simplified by defining $\alpha_i(n+1)$ (Ref. 4, p. 77, (4)),

$$\alpha_i(n+1) = \frac{h_{n+1}}{\psi_i(n+1)}, \tag{29}$$

so that $c_{i,n}(s)$, (21), can be written (Ref. 4, p. 80),

$$c_{i,n}(s) = \begin{cases} 1 & i = 1, \\ s & i = 2, \\ \left(\alpha_{i-1}(n+1)s + \dfrac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) c_{i-1,n}(s) & i \geq 3. \end{cases} \tag{30}$$

Focusing on $i \geq 3$, the integral of $c_{i,n}$ to an arbitrary point s can be written using (30) (Ref. 4, p. 80),

$$\int_0^s c_{i,n}(s_0)\,ds_0 = \int_0^s \left(\alpha_{i-1}(n+1)s_0 + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) c_{i-1,n}(s_0)\,ds_0. \tag{31}$$

This integral can be solved using integration by parts,

$$\int u\,dv = uv - \int v\,du, \tag{32}$$

where in this case

$$u = \left(\alpha_{i-1}(n+1)s_0 + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right), \tag{33}$$

and

$$dv = c_{i-1,n}(s_0)\,ds_0, \tag{34}$$

which gives (Ref. 4, p. 80, (9)),

$$\begin{aligned}\int_0^s c_{i,n}(s_0)\,ds_0 = &\left(\alpha_{i-1}(n+1)s + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) \int_0^s c_{i-1,n}(s_0)\,ds_0 \\ &- \alpha_{i-1}(n+1) \int_0^{s_0} \int_0^{s_1} c_{i-1,n}(s_0)\,ds_0\,ds_1.\end{aligned} \tag{35}$$

Though a double integral has been introduced, it is on a $c_{i-1,n}$ term. Since the integrals of the $c_{1,n}$ and $c_{2,n}$ terms can be found easily, a recursive formula for the integral of $c_{i,n}$ can be found in terms of multiple integrals of lower i values of c.

The recursive formula is found by first introducing notation for multiple integrals of $c_{i,n}(s)$ (Ref. 4, p. 81),

$$c_{i,n}^{(-q)}(s) = \int_0^s \int_0^{s_{q-1}} \int_0^{s_{q-2}} \cdots \int_0^{s_0} c_{i,n}(s_0)\,ds_0\,ds_1 \ldots ds_{q-1}. \tag{36}$$

Under this notation (35) may be written (Ref. 4, p. 81)

$$c_{i,n}^{(-1)}(s) = \left(\alpha_{i-1}(n+1)s + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) c_{i-1,n}^{(-1)}(s) - \alpha_{i-1}(n+1) c_{i-1,n}^{(-2)}(s). \tag{37}$$

The general case is (Ref. 4, p. 81)

$$c_{i,n}^{(-q)}(s) = \left(\alpha_{i-1}(n+1)s + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) c_{i-1,n}^{(-q)}(s) - q\alpha_{i-1}(n+1) c_{i-1,n}^{(-q-1)}(s). \tag{38}$$

To find the predicted value, (27) indicates that $c_{i,n}^{(-1)}(1)$ is needed. A variable $g_{i,q}$ is introduced to simplify the process of finding these values (Ref. 4, p. 81),

$$g_{i,q} = (q-1)!\,c_{i,n}^{(-q)}(1). \tag{39}$$

Substituting in the formula for $c_{i,n}^{(-q)}$ given in (37),

$$\begin{aligned} g_{i,q} &= \left(\alpha_{i-1}(n+1) + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right)(q-1)!\,c_{i-1,n}^{(-q)}(1) - \alpha_{i-1}(n+1)q!\,c_{i-1,n}^{(-q-1)}(1) \\ &= \left(\frac{h_{n+1}}{\psi_{i-1}(n+1)} + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) g_{i-1,q} - \alpha_{i-1}(n+1) g_{i-1,q+1}, \end{aligned} \tag{40}$$

where $\alpha_{i-1}(n+1)$ has been replaced with (29) on the second line. This formula is further simplified by noting that $h_{n+1} + \psi_{i-2}(n) = \psi_{i-1}(n+1)$ (Ref. 4, p. 82),

$$g_{i,q} = g_{i-1,q} - \alpha_{i-1}(n+1) g_{i-1,q+1}. \tag{41}$$

This equation holds when $i \geq 3$.

To write a recursive formula for g, the special cases of $i = 1$ and $i = 2$ must be considered. When $i = 1$, $c_{1,n} = 1$, so the integral is

$$g_{1,q} = (q-1)! c_{1,n}^{(-q)}(1) = (q-1)! \int_0^1 \int_0^{s_{q-1}} \cdots \int_0^{s_1} ds_0\, ds_1 \ldots ds_{q-1} = \frac{(q-1)!}{q!} = \frac{1}{q}. \tag{42}$$

When $i = 2$, $c_{2,n} = s$, so the integral is

$$g_{2,q} = (q-1)! \int_0^1 \int_0^{s_{q-1}} \cdots \int_0^{s_1} s\, ds_0\, ds_1 \ldots ds_{q-1} = \frac{(q-1)!}{(q+1)!} = \frac{1}{q(q+1)}. \tag{43}$$

Now a recursive formula for the coefficients $g_{i,q}$ is available (Ref. 4, p. 82, (10)),

$$g_{i,q} = \begin{cases} \dfrac{1}{q} & i = 1, \\ \dfrac{1}{q(q+1)} & i = 2, \\ g_{i-1,q} - \alpha_{i-1}(n+1) g_{i-1,q+1} & i \geq 3, \end{cases} \tag{44}$$

and the predictor formula (11), (27), can be written (Ref. 4, p. 82, (11)),

$$p_{n+1} = y_n + h_{n+1} \sum_{i=1}^{k} g_{i,1} \phi_i^*(n). \tag{45}$$

This formula is the Shampine-Gordon predictor. The predictor is followed by an evaluation, and then the corrector.

Corrector

After the predicted value p_{n+1} is found, the function $f(x, y)$ is evaluated at the new point, giving a predicted function value, $f_{n+1}^p = f(x_{n+1}, p_{n+1})$. The superscript p is used to indicate the value is predicted. The corrector then uses an interpolating polynomial that is one degree higher than $P_{k,n}(x)$, interpolating through all the same points as $P_{k,n}(x)$ plus the new point f_{n+1}^p. This new polynomial $P_{k+1,n}^*(x)$ can be written in terms of $P_{k,n}(x)$ (see Divided Differences Section) (Ref. 4, p. 84),

$$P_{k+1,n}^*(x) = P_{k,n}(x) + c_{k+1,n}(s) \phi_{k+1}^p(n+1). \tag{46}$$

The new modified divided difference, $\phi_{k+1}^p(n+1)$, is calculated from the previous modified divided differences, $\phi_i(n)$, and the new function value f_{n+1}^p. From the definition of divided differences, (7), and the definition of modified divided differences, (13), the relation from $\phi_i^p(n+1)$ to $\phi_i(n)$ can be found,

$$\phi_i^p(n+1) = \phi_{i-1}(n+1) - \frac{\psi_1(n+1) \cdots \psi_{i-1}(n+1)}{\psi_1(n) \cdots \psi_{i-1}(n)} \phi_{i-1}(n), \tag{47}$$

where $\phi_1^p(n+1) = f_{n+1}^p$. This relation is simplified by the definition of β_i, (18), and the definition of $\phi_i^*(n)$, (23) (Ref. 4, p. 85, (14)),

$$\begin{aligned} \phi_1^p(n+1) &= f_{n+1}^p, \\ \phi_i^p(n+1) &= \phi_{i-1}^p(n+1) - \phi_{i-1}^*(n+1). \end{aligned} \tag{48}$$

This relation motivates why β is introduced into the derivation of the predictor.

The corrected value at point $(n+1)$, y_{n+1}, may be found by integrating $P_{k+1,n}^*(x)$,

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} \left[P_{k,n}(x) + c_{k+1,n}(s) \phi_{k+1}^p(n+1) \right] dx. \tag{49}$$

Combining the terms already known to comprise p_{n+1}, and changing the integration variable to s, the expression is

$$y_{n+1} = p_{n+1} + h_{n+1} \int_0^1 c_{k+1,n}(s)\phi^p_{k+1}(n+1)\, ds. \tag{50}$$

The integral term may be replaced with $g_{k+1,1}$ (Ref. 4, p. 85),

$$y_{n+1} = p_{n+1} + h_{n+1} g_{k+1,1} \phi^p_{k+1}(n+1), \tag{51}$$

which is the Shampine-Gordon corrector formula.

After the corrected value is found, another function evaluation is performed, to find $f_{n+1} = f(x_{n+1}, y_{n+1})$. A new set of differences, $\phi_i(n+1)$ are found from a relation analogous to (48) (Ref. 4, p. 85, (15)),

$$\begin{aligned} \phi_1(n+1) &= f_{n+1}, \\ \phi_i(n+1) &= \phi_{i-1}(n+1) - \phi^*_{i-1}(n+1). \end{aligned} \tag{52}$$

These differences are used by the predictor in the next step.

Step-Size Control

The step size is controlled by keeping the local error at each step below a user-defined tolerance, ϵ. The local error is estimated by subtracting from the corrected value y_{n+1} a value given by a lower order corrector $y_{n+1}(k)$ (Ref. 4, p. 101),

$$le_{n+1}(k) \approx y_{n+1} - y_{n+1}(k). \tag{53}$$

The value of $y_{n+1}(k)$ comes from integrating a $(k-1)^{\text{th}}$ degree interpolating polynomial $P^*_{k,n}$ which passes through the predicted point f^p_{n+1}. This polynomial can be written by adding a term to the polynomial $P_{k,n}$,

$$P^*_{k,n}(x) = P_{k,n}(x) + c_{k,n}(s)\phi^p_{k+1}(n+1). \tag{54}$$

Using this polynomial in place of $P^*_{k+1,n}$ in (49) leads to an expression for $y_{n+1}(k)$ (Ref. 4, p. 85),

$$y_{n+1}(k) = p_{n+1} + h_{n+1} g_{k,1} \phi^p_{k+1}(n+1). \tag{55}$$

The local error, (53), is estimated by subtracting (55) from (51) (Ref. 4, p. 101),

$$le_{n+1}(k) \approx h_{n+1}(g_{k+1,1} - g_{k,1})\phi^p_{k+1}(n+1). \tag{56}$$

At each step, this local error estimate is compared to the tolerance. If the local error is above the tolerance, the step fails, and is tried again with half the step size, $h_{n+1} = 0.5 h_{n+1\ (\text{failed})}$ (Ref. 4, p. 117). Note that this error estimate is made with $\phi^p_{k+1}(n+1)$, before the second evaluation is performed. Therefore, if a step fails, the second evaluation does not need to be performed.

If the step succeeds, the next step size, h_{n+2}, is chosen as a multiple of the current step size, $h_{n+2} = r h_{n+1}$, to keep the local error at the next step (Ref. 4, p. 111),

$$le_{n+2}(k) \approx h_{n+2}(g_{k+1,1} - g_{k,1})\phi^p_{k+1}(n+2), \tag{57}$$

as close as possible to the tolerance. The modified divided difference at the next step (Ref. 4, p. 111),

$$\phi^*_{k+1}(n+2) = \psi_1(n+2) \cdots \psi_k(n+2) f^p[x_{n+2}, \ldots, x_{n+2-k}], \tag{58}$$

is unknown. However, it may be approximated from $\phi_{k+1}(n+1)$ if the divided differences are assumed to be slowly varying (Ref. 4, p. 111).

Because the step size h_{n+2} appears in the values of $\psi_i(n+2)$, and is needed to calculate the coefficients $g_{k+1,1}$ and $g_{k,1}$, there is no easy way to solve (56) for a value of h_{n+2} that meets the tolerance. Instead, a value of h_{n+2} is found that meets the tolerance if all the preceeding steps were also taken with h_{n+2}. Using this assumption, and the assumption that the divided differences are slowly varying, the modified divided difference at $(n+2)$ is approximated (Ref. 4, p. 111),

$$\begin{aligned}\phi^p_{k+1}(n+2) &\approx (rh_{n+1})(2rh_{n+1})\cdots(krh_{n+1})f^p[x_{n+2},\ldots,x_{n+2-k}]\\ &\approx r^k\sigma_{k+1}(n+1)\phi^p_{k+1}(n+1),\end{aligned} \tag{59}$$

where $\sigma_i(n+1)$ is defined (Ref. 4, p. 111),

$$\begin{aligned}\sigma_1(n+1) &= 1,\\ \sigma_i(n+1) &= \frac{h_{n+1}\cdot 2h_{n+1}\cdots(i-1)h_{n+1}}{\psi_1(n+1)\cdot\psi_2(n+1)\cdots\psi_{i-1}(n+1)} \qquad i>1.\end{aligned} \tag{60}$$

When the step size is constant, the coefficients $g_{k+1,1}$ and $g_{k,1}$ become the fixed step Adams-Bashforth predictor coefficients, γ_k and γ_{k-1} (Ref. 4, p. 83). The local error estimate, (57), using $h_{n+2} = rh_{n+1}$ can now be approximated (Ref. 4, p. 112),

$$le_{n+2}(k) = r^{k+1}h_{n+1}\gamma^*_k\sigma_{k+1}(n+1)\phi^p_{k+1}(n+1), \tag{61}$$

where γ^*_k is the difference between the constant step size coefficients, $\gamma^*_k = \gamma_k - \gamma_{k-1}$. The value of $\sigma_{k+1}(n+1)$ can be found through a recursive formula (Ref. 4, p. 112),

$$\begin{aligned}\sigma_1(n+1) &= 1,\\ \sigma_i(n+1) &= (i-1)\alpha_{i-1}(n+1)\sigma_{i-1}(n+1) \qquad i>1.\end{aligned} \tag{62}$$

To solve for the value of r to get the next step size, $h_{n+2} = rh_{n+1}$, the Shampine-Gordon integrator calculates the approximated error (ERK) that would be made if the previous steps had been taken with h_{n+1} (Ref. 4, p. 112),

$$\text{ERK} = |h_{n+1}\gamma^*_k\sigma_{k+1}(n+1)\phi^p_{k+1}(n+1)|. \tag{63}$$

From (61), the approximated error at the next step with a step size of rh_{n+1} is r^{k+1}ERK, so the optimal value of r to satisfy the tolerance ϵ can be found (Ref. 4, p. 115),

$$r = \left(\frac{\epsilon}{\text{ERK}}\right)^{\frac{1}{k+1}}. \tag{64}$$

However, because of the assumptions made in this derivation, the integrator uses a so-called "chicken factor" (Ref. 8) of 0.5, giving a more conservative value of r (Ref. 4, p. 116),

$$r = \left(\frac{0.5\epsilon}{\text{ERK}}\right)^{\frac{1}{k+1}}. \tag{65}$$

Shampine-Gordon places further restrictions on changes to the step size (Ref. 4, pp. 115-118). If the calculated value of r is greater than 2, the step size is only doubled. The step size is not changed at all if the calculated value of r is between 0.9 and 2. And if r is less than 0.5, the step size is cut in half. These restrictions serve two purposes. First, bounding changes in the step size between 0.5 and 2 keeps the method stable. Second, not changing the step size when r is between 0.9 and 2 keeps the step size constant for a significant number of steps. Shampine and Gordon sought to keep a constant step size as much as possible. With a constant step size, the coefficients g do not have to be calculated, because they are simply the known Adams-Bashforth coefficients. Not needing to calculate the coefficients provides a reduction in overhead when the step size is constant. Also, keeping a constant step size allows the order of the method to be increased, which is described in the following section.

Variable Order

Shampine-Gordon also controls the local error by adjusting the order of the method (Ref. 4, pp. 112-115). Local error estimates, similar to (63), are made for $(k-1)$ and $(k-2)$, and the order is reduced from k to $(k-1)$ if these estimates indicate that reducing the order would give less error. These estimates rely on $\phi_k^p(n+1)$ and $\phi_{k-1}^p(n+1)$, so they are made before the second evaluation is performed. Therefore, the order can be reduced if the step fails.

The step size is increased only when the step size has been kept constant. A local error estimate is made for $(k+1)$, and the order is increased if the estimate indicates the error would be reduced. The local error estimate for $(k+1)$ uses the value $\phi_{k+2}(n+1)$ (Ref. 4, p. 113), so the second evaluation must be performed before increasing the order can be considered.

The variable-order algorithm allows Shampine-Gordon to be a self-starting method. The method is started as first order ($k=1$), so only the initial conditions are required. The order is then increased at every step until either a step fails, the order selection algorithms indicate to lower the order, or the maximum order of 12 is reached (Ref. 4, pp 118-120).

DOUBLE INTEGRATION

Second-order differential equations, of the form

$$y'' = f(x, y, y') \tag{66}$$

can be solved by double integration. This method is especially useful when the contribution of y' is small. A variable-step double-integration method can be derived using the concepts in the derivation of Shampine-Gordon. However, the proposed implementation differs from Shampine-Gordon in several ways. Only one evaluation is performed per step, for a PEC implementation, which significantly reduces the run-time of the method. Because the second evaluation is not performed, the method is not variable order, because the second evaluation would be necessary to estimate when the order should be increased. Finally, because our main goal is to reduce run-time, and because we are not considering order increases which require a constant step, less restrictions are placed on the factor r which changes the step size. The argument that keeping the step size constant reduces the overhead does not apply, because the force model used in orbit propagation is so expensive that the overhead associated with calculating the coefficients is insignificant.

Predictor

To solve for y, take the double integral of each side in (66),

$$\int_{x_n}^{x_{n+1}} \int_{x_n}^{\tilde{x}} y''(x)\,dx\,d\tilde{x} = \int_{x_n}^{x_{n+1}} \int_{x_n}^{\tilde{x}} f(x, y, y')\,dx\,d\tilde{x}. \tag{67}$$

Performing the integration on the left side of (67) gives

$$y_{n+1} = y_n + h_{n+1} y_n' + \int_{x_n}^{x_{n+1}} \int_{x_n}^{\tilde{x}} f(x, y, y')\,dx\,d\tilde{x}. \tag{68}$$

This equation contains a first derivative term, y_n', which must be removed for a truly double integration formula. To remove the first derivative term, take a step backward (Ref. 9, p. 290),

$$\int_{x_n}^{x_{n-1}} \int_{x_n}^{\tilde{x}} y''(x)\,dx\,d\tilde{x} = \int_{x_n}^{x_{n-1}} \int_{x_n}^{\tilde{x}} f(x, y, y')\,dx\,d\tilde{x}, \tag{69}$$

which leads to the relation

$$0 = y_n - y_{n-1} - h_n y_n' + \int_{x_n}^{x_{n-1}} \int_{x_n}^{\tilde{x}} f(x, y, y')\,dx\,d\tilde{x}. \tag{70}$$

The first derivative term is removed by adding (h_{n+1}/h_n) (70) to (68),

$$y_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + \int_{x_n}^{x_{n+1}} \int_{x_n}^{\tilde{x}} f(x, y, y')\, dx\, d\tilde{x} + \frac{h_{n+1}}{h_n} \int_{x_n}^{x_{n-1}} \int_{x_n}^{\tilde{x}} f(x, y, y')\, dx\, d\tilde{x}. \tag{71}$$

As with single integration, the interpolating polynomial $P_{k,n}$ is used in place of $f(x, y, y')$ to give an expression for the predicted value, p_{n+1},

$$p_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + \int_{x_n}^{x_{n+1}} \int_{x_n}^{\tilde{x}} P_{k,n}(x)\, dx\, d\tilde{x} + \frac{h_{n+1}}{h_n} \int_{x_n}^{x_{n-1}} \int_{x_n}^{\tilde{x}} P_{k,n}(x)\, dx\, d\tilde{x}. \tag{72}$$

Substituting (24) for $P_{k,n}$ gives

$$p_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + \int_{x_n}^{x_{n+1}} \int_{x_n}^{\tilde{x}} \sum_{i=1}^{k} c_{i,n}(s) \phi_i^*(n)\, dx\, d\tilde{x} + \frac{h_{n+1}}{h_n} \int_{x_n}^{x_{n-1}} \int_{x_n}^{\tilde{x}} \sum_{i=1}^{k} c_{i,n}(s) \phi_i^*(n)\, dx\, d\tilde{x}. \tag{73}$$

The integration variable can be changed to s by noting that $s = -h_n/h_{n+1}$ when $x = x_{n-1}$,

$$p_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + h_{n+1}^2 \sum_{i=1}^{k} \phi_i^*(n) \int_0^1 \int_0^{\tilde{s}} c_{i,n}(s)\, ds\, d\tilde{s} + \frac{h_{n+1}^3}{h_n} \sum_{i=1}^{k} \phi_i^*(n) \int_0^{\frac{-h_n}{h_{n+1}}} \int_0^{\tilde{s}} c_{i,n}(s)\, ds\, d\tilde{s}. \tag{74}$$

This expression can be written with the simplified notation $c_{i,n}^{(-q)}$ using (36),

$$p_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + h_{n+1}^2 \sum_{i=1}^{k} \phi_i^*(n) c_{i,n}^{(-2)}(1) + \frac{h_{n+1}^3}{h_n} \sum_{i=1}^{k} \phi_i^*(n) c_{i,n}^{(-2)} \left(\frac{-h_n}{h_{n+1}}\right) \tag{75}$$

The coefficients $g_{i,2}$, already found for single integration, can be used to calculate the first integration term. A new set of coefficients, $g'_{i,q}$, is needed to find the second integration term. Define $g'_{i,q}$ as

$$g'_{i,q} = (q-1)! c_{i,n}^{(-q)} \left(\frac{-h_n}{h_{n+1}}\right). \tag{76}$$

Using (38), for $i \geq 3$, (76) is

$$g'_{i,q} = \left(\alpha_{i-1}(n+1) \frac{-h_n}{h_{n+1}} + \frac{\psi_{i-2}(n)}{\psi_{i-1}(n+1)}\right) (q-1)! c_{i-1,n}^{(-q)} \left(\frac{-h_n}{h_{n+1}}\right) - \alpha_{i-1}(n+1) q! c_{i-1,n}^{(-q-1)} \left(\frac{-h_n}{h_{n+1}}\right). \tag{77}$$

Using the definition of α, (29), and noting that $-h_n + \psi_{i-2}(n) = \psi_{i-3}(n-1)$, the expression simplifies,

$$g'_{i,q} = \frac{\psi_{i-3}(n-1)}{\psi_{i-1}(n+1)} g'_{i-1,q} - \alpha_{i-1}(n+1) g'_{i-1,q+1}, \tag{78}$$

for $i \geq 3$. To implement a recursive formula for g', the expressions for $i = 1$ and $i = 2$ are needed. When $i = 1$, change the limit of integration in (42),

$$g'_{1,q} = (q-1)! \int_0^{\frac{-h_n}{h_{n+1}}} \cdots \int_0^{s_1} ds_0 \ldots ds_{q-1} = \frac{(q-1)!}{q!} \left(\frac{h_n}{h_{n+1}}\right)^q = \frac{1}{q} \left(\frac{h_n}{h_{n+1}}\right)^q . \tag{79}$$

For $i = 2$, change the integration limit in (43),

$$g'_{2,q} = (q-1)! \int_0^{\frac{-h_n}{h_{n+1}}} \cdots \int_0^{s_1} s \, ds_0 \ldots ds_{q-1} = \frac{(q-1)!}{(q+1)!} \left(\frac{h_n}{h_{n+1}}\right)^{q+1} = \frac{1}{q(q+1)} \left(\frac{h_n}{h_{n+1}}\right)^{q+1} . \tag{80}$$

Now a recursive formula for $g'_{i,q}$ is available, similar to (44) for $g_{i,q}$,

$$g'_{i,q} = \begin{cases} \frac{1}{q} \left(\frac{h_n}{h_{n+1}}\right)^q & i = 1, \\ \frac{1}{q(q+1)} \left(\frac{h_n}{h_{n+1}}\right)^{q+1} & i = 2, \\ \frac{\psi_{i-3}(n-1)}{\psi_{i-1}(n+1)} g'_{i-1,q} - \alpha_{i-1}(n+1) g'_{i-1,q+1} & i \geq 3. \end{cases} \tag{81}$$

Note that for $i = 3$, $\psi_{i-3}(n-1) = 0$.

Using the coefficients g and g', the double-integration predictor formula (75) can be written

$$p_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + h_{n+1}^2 \sum_{i=1}^{k} \phi_i^*(n) g_{i,2} + \frac{h_{n+1}^3}{h_n} \sum_{i=1}^{k} \phi_i^*(n) g'_{i,2}, \tag{82}$$

or, combining the terms,

$$p_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + h_{n+1}^2 \sum_{i=1}^{k} \left(g_{i,2} + \frac{h_{n+1}}{h_n} g'_{i,2}\right) \phi_i^*(n). \tag{83}$$

This expression is the predictor formula for double integration.

Corrector

As in single integration, the corrector uses the interpolating polynomial $P^*_{k+1,n}$, given in (46). The corrected value at point $(n+1)$, y_{n+1}, is found by replacing $P_{k,n}$ with $P^*_{k+1,n}$ in (72),

$$\begin{aligned} y_{n+1} = \left(1 + \frac{h_{n+1}}{h_n}\right) y_n - \frac{h_{n+1}}{h_n} y_{n-1} + \int_{x_n}^{x_{n+1}} \int_{x_n}^{\tilde{x}} \left[P_{k,n}(x) + c_{k+1,n}(s) \phi^p_{k+1}(n+1)\right] dx \, d\tilde{x} \\ + \frac{h_{n+1}}{h_n} \int_{x_n}^{x_{n-1}} \int_{x_n}^{\tilde{x}} \left[P_{k,n}(x) + c_{k+1,n}(s) \phi^p_{k+1}(n+1)\right] dx \, d\tilde{x}. \end{aligned} \tag{84}$$

This expression is simplified by combining the terms known to be p_{n+1} and by changing the integration variable to s,

$$\begin{aligned} y_{n+1} = & p_{n+1} + h_{n+1}^2 \int_0^1 \int_0^{\tilde{s}} c_{k+1,n}(s) \phi^p_{k+1}(n+1) \, ds \, d\tilde{s} \\ & + \frac{h_{n+1}^3}{h_n} \int_0^{\frac{-h_n}{h_{n+1}}} \int_0^{\tilde{s}} c_{k+1,n}(s) \phi^p_{k+1}(n+1) \, ds \, d\tilde{s}. \end{aligned} \tag{85}$$

The integrals can be written in terms of the coefficients g and g',

$$y_{n+1} = p_{n+1} + h_{n+1}^2 \left(g_{k+1,2} + \frac{h_{n+1}}{h_n} g'_{k+1,2}\right) \phi^p_{k+1}(n+1), \tag{86}$$

giving a corrector formula for double integration. To reduce run-time, a second evaluation is not performed, so only a PEC implementation is used. Results show this implementation to be stable. Without the second evaluation the differences used by the predictor in the next step are simply $\phi_i(n+1) = \phi_i^p(n+1)$.

Step-Size Control

The step size is controlled by estimating the local error at each step, through (53). For double integration, the value of $y_{n+1}(k)$ is found by replacing $P^*_{k+1,n}(x)$ with $P^*_{k,n}(x)$, (54), in (84),

$$\begin{aligned} y_{n+1}(k) = & p_{n+1} + h_{n+1}^2 \int_0^1 \int_0^{\tilde{s}} c_{k,n}(s)\phi^p_{k+1}(n+1)\, ds\, d\tilde{s} \\ & + \frac{h_{n+1}^3}{h_n} \int_0^{\frac{-h_n}{h_{n+1}}} \int_0^{\tilde{s}} c_{k+1,n}(s)\phi^p_k(n+1)\, ds\, d\tilde{s}. \end{aligned} \tag{87}$$

The integrals may be written in terms of the coefficients g and g',

$$y_{n+1}(k) = p_{n+1} + h_{n+1}^2 \left(g_{k,2} + \frac{h_{n+1}}{h_n} g'_{k,2} \right) \phi^p_{k+1}(n+1). \tag{88}$$

The local error is estimated by subtracting (88) from (86),

$$le_{n+1}(k) \approx h_{n+1}^2 \left(g_{k+1,2} - g_{k,2} + \frac{h_{n+1}}{h_n} (g'_{k+1,2} - g'_{k,2}) \right) \phi^p_{k+1}(n+1). \tag{89}$$

To choose the step size at the next step, $h_{n+2} = rh_{n+1}$, the local error at that step is approximated, analogous to (57),

$$le_{n+2}(k) \approx h_{n+2}^2 \left(g_{k+1,2} - g_{k,2} + \frac{h_{n+2}}{h_{n+1}} (g'_{k+1,2} - g'_{k,2}) \right) \phi^p_{k+1}(n+2). \tag{90}$$

As with single integration, this expression is approximated assuming that the differences are slowly varying and the previous steps were also taken at rh_{n+1}, analogous to (61),

$$le_{n+2}(k) \approx r^2 h_{n+1}^2 (\lambda_k - \lambda_{k-1}) r^k \sigma_{k+1}(n+1) \phi^p_{k+1}(n+1), \tag{91}$$

where λ_i are the Störmer-Cowell predictor coefficients (Ref. 10). Introducing $\lambda^*_k = \lambda_k - \lambda_{k-1}$ simplifies the expression,

$$le_{n+2}(k) \approx r^{k+2} h_{n+1}^2 \lambda^*_k \sigma_{k+1}(n+1) \phi^p_{k+1}(n+1). \tag{92}$$

To calculate the value of r, the error using a step size of h_{n+1} is found, as in (63),

$$\text{ERK} = |h^2 \lambda^*_k \sigma_{k+1}(n+1) \phi^p_{k+1}(n+1)|. \tag{93}$$

Using a chicken factor of 0.5, the factor r for the next step is found similarly to (65),

$$r = \left(\frac{0.5\epsilon}{\text{ERK}} \right)^{\frac{1}{k+2}}. \tag{94}$$

For stability, the calculated value of r is bounded between 0.5 and 2. However, no other restrictions are placed on r, so that unlike Shampine-Gordon step-size increases between 1 and 2 can be made. This technique allows the step size to increase as soon as possible, which reduces overall run-time. Because of the expensive force model it is better to make a small increase in the step size, even though the coefficients g and g' must be recomputed.

IMPLEMENTATION

The double-integration variable-step integrator is implemented with a Shampine-Gordon style single-integration integrator to solve the differential equation $y''(x) = f(x, y, y')$. The double-integration integrator is used to find y and the single-integration integrator is used to find y'. The integrators are implemented together to use the same step size, which is the smaller of the two step sizes given by their respective step-size control algorithms. The single-integration integrator differs from Shampine-Gordon in that the order of the method remains fixed, the step size is allowed to change by a factor between 0.9 and 2, and only one evaluation is performed per step, just as with the double-integration integrator.

Because the method is not variable order, a start-up method is required. A fourth-order Runge-Kutta integrator is first used to take $(k-1)$ steps forward, so k backpoints are available when the method begins. Because Runge-Kutta is a lower order method, the step size used must be adequately small so the values meet the desired tolerance. An alternative to using Runge-Kutta for start-up would be to use a start-up routine similar to Shampine-Gordon in which the method starts as a first-order method and increases the order at each step, until the desired order is reached. A second evaluation per step would be required only during this start-up phase.

After the start-up has been performed, the predictor-corrector cycle begins. At each step $(n+1)$,

1. The new step size is calculated, $h_{n+1} = rh_n$.
2. The values of $\psi_i(n+1)$, $\psi_i(n)$, and $\psi_i(n-1)$ are calculated, (14).
3. The values of $\alpha_i(n+1)$ are calculated, (29).
4. The coefficients are calculated, g, (44), and g', (81).
5. The values of β_i are calculated, (18), and the values of ϕ_i^*, (23).
6. The predicted values of p_{n+1}, (83), and p'_{n+1}, (51), are found.
7. The function is evaluated at the predicted point.
8. The new differences $\phi_i(n+1)$ are calculated, (52).
9. The corrected values of y_{n+1}, (86), and y'_{n+1}, (51), are found.
10. The error is estimated for y, (89), and y', (56).
11. If either error estimate is above the tolerance, the step fails, the differences are reset, r is set to 0.5, and the procedure returns to step 1.
12. The value of ERK is calculated for double integration, (93), and single integration, (63).
13. The factor r recommended for double integration, (94), and single integration, (65) is calculated.
14. The factor r is set to the lower of the two recommended values.
15. The factor r is bounded between 0.5 and 2.
16. The step n is incremented and the procedure returns to step 1.

RESULTS

Two separate implementations show the integrator to be effective. The first implementation is in Matlab. The Matlab implementation is used to integrate the second order differential equation $y'' = -y$, with initial conditions $y(0) = 0$, $y'(0) = 1$, over $0 \leq x \leq 10\pi$. The exact solution of this problem is $y(x) = \sin(x)$. Because this problem has no dependence on y', only the double-integration method has been implemented. The method is started using Runge-Kutta, with a step size of 0.1. Nine backpoints are used in the implementation, $k = 9$. A tolerance of $\epsilon = 1 \times 10^{-13}$ is used in the integration. Figure 2 shows a plot of the numerical solution, the step sizes used, and the total error at each step, $|y_n - sin(x_n)|$.

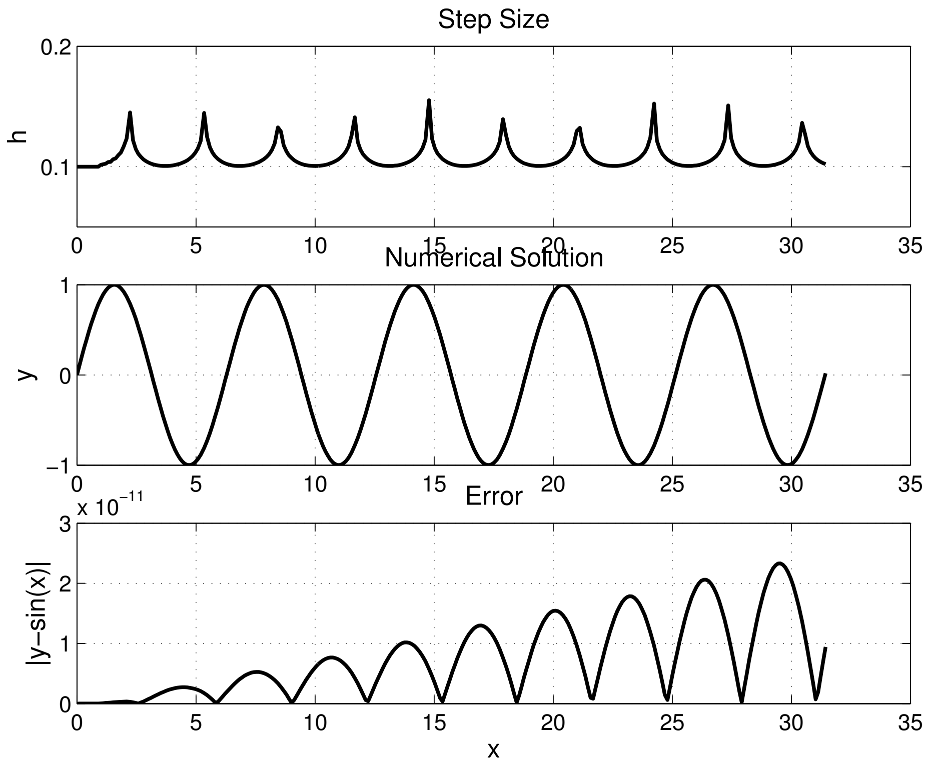

Figure 2: Results of integrating $y'' = -y$.

Figure 2 shows that the step size fluctuates periodically between approximately 0.1 and 0.15. There is a slight offset between the peaks of the solution and the peaks of the step-size curve. The error curve also behaves periodically, with peaks of the error correlating to peaks of the solution. The error grows as the integration progresses, which is expected. The maximum error is 2.33×10^{-11}.

A Fortran implementation has also been used to test the integration method on orbit propagation. The full implementation of both the variable-step double and single-integration integrators described in the Implementation Section has been implemented into Special-K (Ref. 5). A fourth-order Runge-Kutta integrator is used to start the method. Again, nine backpoints are used. Table 2 shows error ratios in a two-body test of various test case orbits. A tolerance of $\epsilon = 1 \times 10^{-12}$ is used in the tests. These preliminary results show the integrator to be effective.

Table 5: Variable-Step Double-Integration Results, Two Body

Height (km)	Eccentricity	Error Ratio
300	0.00	6.41×10^{-10}
300	0.25	7.49×10^{-11}
300	0.50	2.04×10^{-11}
300	0.75	1.98×10^{-11}
500	0.00	6.23×10^{-10}
500	0.25	5.99×10^{-11}
500	0.50	2.20×10^{-11}
500	0.75	2.04×10^{-11}
1000	0.00	5.81×10^{-10}
1000	0.25	5.97×10^{-11}
1000	0.50	2.14×10^{-11}
1000	0.75	2.31×10^{-11}

FUTURE WORK

Now that this integrator has been developed, extensive testing comparing it to other integration methods is required to show where it has an advantage. In particular, speed testing on a range of eccentricities will indicate at what eccentricities the method should be used, and how much computation time can be saved.

Several issues regarding the implementation of this integrator remain. One issue is the start-up method. Though using Runge-Kutta for start-up is effective, starting up by using a variable-order version would reduce the complexity of the implementation because an additional integrator would not be required.

A second issue is interpolation of the solution values. In orbit propagation and orbit determination, solution values are required at specific time values. These time values do not normally correspond to the integration steps, so an interpolator is used to find the solution at the required time. In Special-K, as in other programs, that interpolation is done outside the integration, and in fact is of lower order than the integrator, causing a loss in accuracy. Because Shampine-Gordon is derived based on integrating an interpolating polynomial, the method allows values to be found at times other than the integration steps, without a loss of accuracy or additional evaluations (Ref. 4, pp. 91-93). Such a technique could also be used for the double-integration method.

Another implementation issue involves choosing the factor r by which to modify the step size. Two values are available, one for single integration and one for double integration. The conservative approach used here is to choose the smaller of the two values. However, in problems where the velocity does not give a significant contribution to the force model, always using the value for double integration may give adequately accurate results at a faster run-time. Also, because the tolerance is set as an absolute value, yet the units are different for position and velocity, it may be appropriate to use different tolerances for single and double integration. Further study into this topic is needed.

ACKNOWLEDGEMENTS

We thank Chris Hall for his comments on the paper.

REFERENCES

[1] Berry, M. and Healy, L., "Implementation of Gauss-Jackson Integration for Orbit Propagation," In *Advances in Astronautics*, San Diego, CA, August 2001. American Astronautical Society AAS 01–426.

[2] Berry, M. and Healy, L., "Accuracy and Speed Effects of Variable Step Integration for Orbit Determination and Propagation," In *Advances in Astronautics*, San Diego, CA, August 2003. American Astronautical Society AAS 03–664.

[3] Berry, M. and Healy, L., "The Generalized Sundman Transformation for Propagation of High-Eccentricity Elliptical Orbits," In *Advances in Astronautics*, San Diego, CA, February 2002. American Astronautical Society AAS 02–109.

[4] Shampine, L. F. and Gordon, M. K., *Computer Solution of Ordinary Differential Equations*, W. H. Freeman and Company, San Francisco, 1975.

[5] Neal, H. L., Coffey, S. L., and Knowles, S., "Maintaining the Space Object Catalog with Special Perturbations," In Hoots, F., Kaufman, B., Cefola, P., and Spencer, D., editors, *Astrodynamics 1997 Part II*, volume 97 of *Advances in the Astronautical Sciences*, pp. 1349–1360, San Diego, CA, August 1997. American Astronautical Society AAS 97–687.

[6] Merson, R. H., *Numerical Integration of the Differential Equations of Celestial Mechanics*, Technical Report TR 74184, Royal Aircraft Establishment, Farnborough, Hants, UK, January 1975. Defense Technical Information Center number AD B004645.

[7] Berry, M. and Healy, L., "Comparison of Accuracy Assessment Techniques for Numerical Integration," In *Advances in Astronautics*, San Diego, CA, February 2003. American Astronautical Society AAS 03–171.

[8] Danby, J. M. A., *Computing Applications to Differential Equations*, Reston Publishing Company, Reston, Virginia, 1985.

[9] Henrici, P., *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley and Sons, New York, 1962.

[10] Maury, J. L. and Segal, G. P., *Cowell type numerical integration as applied to satellite orbit computation*, Technical Report X-553-69-46, NASA, 1969. NTIS #N6926703.