Application of Taylor-Series Integration to Reentry Problems with Wind

Bergsma, Michiel; Mooij, Erwin

DOI
10.2514/1.G000378

Publication date
2016

Document Version
Accepted author manuscript

Published in
Journal of Guidance, Control, and Dynamics: devoted to the technology of dynamics and control

Citation (APA)
Bergsma, M., & Mooij, E. (2016). Application of Taylor-Series Integration to Reentry Problems with Wind. *Journal of Guidance, Control, and Dynamics: devoted to the technology of dynamics and control*, *39*, 2324-2335. https://doi.org/10.2514/1.G000378

This is an Accepted Manuscript of an article published in:
JOURNAL OF GUIDANCE,CONTROL, AND DYNAMICS Vol. 39, No. 10, October 2016
Available online: https://arc.aiaa.org/doi/10.2514/1.G000378

Application of Taylor Series Integration to Reentry Problems With Wind

M.C.W. Bergsma[1] and E. Mooij[2]
Delft University of Technology, Faculty of Aerospace Engineering,

Kluyverweg 1, 2629 HS Delft, The Netherlands

Taylor Series Integration is a numerical integration technique that computes the Taylor series of state variables using recurrence relations and uses this series to propagate the state in time. A Taylor Series Integration reentry integrator was developed and compared with the fifth-order Runge-Kutta-Fehlberg integrator to determine whether Taylor Series Integration is faster than traditional integration methods for reentry applications. By comparing the CPU times of the integrators, Taylor Series Integration was indeed found to be faster for integration without wind and slower with wind, unless the error tolerance was 10^{-8} or lower. Furthermore, it was found that reducing step sizes to prevent integration over discontinuities is not only needed for Taylor Series Integration to obtain maximum accuracy, but also for Runge-Kutta-Fehlberg methods. In that case, the Runge-Kutta-Fehlberg integrator does become several times slower than Taylor Series Integration.

I. Introduction

According to Barrio, Taylor Series Integration (TSI) dates back to the time of Newton (late 17^{th} or early 18^{th} century), who used recursive computation of Taylor series for the solution of ordinary differential equations (ODEs) [1]. Among the first to use TSI in the 20^{th} century was Moore, who, for some time, has been the main promoter of TSI as replacement for the, in his opinion, less efficient

[1] MSc. Graduate, section Astrodynamics and Space Missions.
[2] Assistant Professor, section Astrodynamics and Space Missions, e.mooij@tudelft.nl, Associate Fellow AIAA.

traditional numerical integrators. He argued that the main objection to TSI, namely the need to evaluate derivatives up to a high order, is overrated, as a computer can do this recursively, and Runge-Kutta integrators of the same order require more arithmetic operations than TSI [2].

The conclusion of Hull et al. was that TSI fares better than Runga-Kutta and variable-order Adams methods when the accuracy requirements are more stringent [3]. Irvine and Savageau found that TSI can, in fact, be up to 20 times faster for high accuracy, i.e., about 16 significant digits [4]. In a more recent comparison, Jorba and Zou found that TSI is capable of achieving higher levels of accuracy then other integrators and that TSI is about 1.5 to 5 times faster for a given achieved level of accuracy [5]. The results of Barrio also show that TSI is capable of greater levels of accuracy and that it is faster for an accuracy of 10^{-8} or lower [1].

From the 1970s onwards, TSI has been analyzed thoroughly, especially by Chang and Corliss, focusing on elements such as step-size control, and expanding the range of problems that TSI could be applied to [6–10]. In the 1990s it was first used as a method to solve differential algebraic equations [9, 11]. TSI was recently found to be a particularly fast method for integrating celestial mechanics, yielding on average 16.6 times lower CPU times than the Runge-Kutta-Fehlberg (RKF) integrators that are typically used for this type of problems [12].

Reentry trajectory propagation is typically done using numerical integrators, such as the methods of Runge-Kutta, Adams-Bashforth, or Adams-Moulton. The strength of TSI is that its step size and order can be changed with little computational (and programming) effort, allowing it, in theory, to solve any problem with the highest efficiency. As a result TSI can have both a higher order and larger step sizes than its competitors. Therefore, TSI is best applied to problems, where the step sizes are not kept artificially small by guidance, navigation or control systems. For such problems, the RKF5(6) integrator [13] was found to be the fastest integration method for reentry applications for a given level of accuracy.

If TSI can outperform RKF5(6) in terms of CPU time, whilst yielding the same results, then it can also replace the traditional methods in applications, such as trajectory optimization (without wind) and Monte Carlo analysis (with or without wind), which require the integration of a large number of trajectories, potentially yielding large reductions in the total CPU time. A TSI integra-

tion program was developed, and its speed and results were compared with those of an RKF5(6) integrator to determine whether TSI is faster. The integration program builds on the models developed in [14], where an extensive performance analysis was done for gliding reentry trajectories without wind. In this paper, the TSI framework is extended with a generic wind model, and the effects of wind are subsequently analyzed.

The remainder of the paper is organized as follows. In Sec. II, a brief overview of the basics and the step-size and order selection of TSI will be given. The reference vehicle, reentry equations, environment models, controls and constraints are given Sec. III. Section IV will explain methods used to deal with discontinuities in the equations of motion. In Sec. V, the results of the determination of the optimal step-size and order controllers for TSI, and the comparison of CPU time and results of the two integrators will be given. Section VI concludes the paper with some final remarks.

II. Taylor Series Integration

Here, the basics of TSI are discussed, together with its application to ODEs and the methods for step-size and order control.

A. Basics

Consider the following initial value problem:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(t, \boldsymbol{x}(t)), \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{1}$$

where $\boldsymbol{x}(t)$ is a vector containing the values of state variables. The Taylor-series expansion of $\boldsymbol{x}(t)$ at t_n can be used to propagate it to t_{n+1}:

$$\boldsymbol{x}(t_{n+1}) = \sum_{k=0}^{\infty} \frac{\boldsymbol{x}^{(k)}(t_n)}{k!} h_n^k \tag{2}$$

where $\boldsymbol{x}^{(k)}$ are the k^{th} derivatives of $\boldsymbol{x}$, and h_n is the step size from t_n to t_{n+1}. The Taylor-series expansion can be rewritten in terms of Taylor coefficients, defined as [2]:

$$(x_n)_k = \frac{x^{(k)}(t_n)}{k!} \tag{3}$$

In a practical application, it is not possible to compute an infinite number of terms, so the Taylor series is truncated at order K:

$$\boldsymbol{x}(t_{n+1}) = \sum_{k=0}^{K} (\boldsymbol{x}_n)_k h_n^k + \mathcal{O}(h_n^{K+1}) \tag{4}$$

where $\mathcal{O}(h_n^{K+1})$ marks the truncation error. Thus, to propagate the state to t_{n+1}, one has to determine its Taylor coefficients $(\boldsymbol{x}_n)_k$. From Eq. (3), $(\boldsymbol{x}_n)_0 = \boldsymbol{x}(t_n)$. The higher-order coefficients are obtained from the differential equation $\boldsymbol{f}(t, \boldsymbol{x}(t))$ by rewriting it in terms of recurrence relations. The recurrence relations of a number of basic mathematical operations on two arbitrary analytical functions $y(t)$ and $z(t)$ are (for $k \geq 1$) [2]:

$$(y \pm z)_k = (y)_k \pm (z)_k \tag{5}$$

$$(yz)_k = \sum_{j=0}^{k} (y)_j (z)_{k-j} \tag{6}$$

$$(y/z)_k = \frac{1}{(z)_0} \left[(y)_k - \sum_{j=1}^{k} (z)_j (y/z)_{k-j} \right] \tag{7}$$

$$(y^\alpha)_k = \frac{1}{(y)_0} \sum_{j=1}^{k} \left(\frac{j}{k}(\alpha + 1) - 1 \right) (y)_j (y^\alpha)_{k-j} \tag{8}$$

$$(e^y)_k = \frac{1}{k} \sum_{j=1}^{k} j (y)_j (e^y)_{k-j} \tag{9}$$

$$(\sin y)_k = \frac{1}{k} \sum_{j=1}^{k} j (y)_j (\cos y)_{k-j} \tag{10}$$

$$(\cos y)_k = -\frac{1}{k} \sum_{j=1}^{k} j (y)_j (\sin y)_{k-j} \tag{11}$$

Equation (10) can be reordered to obtain the recurrence relation for an angle θ using the coefficients of its sine and cosine:

$$(\theta)_k = \frac{1}{(\cos\theta)_0} \left[(\sin\theta)_k - \frac{1}{k} \sum_{j=1}^{k-1} j (\theta)_j (\cos\theta)_{k-j} \right] \tag{12}$$

B. Application to Ordinary Differential Equations

As an example of the application of TSI, consider the following ODE:

$$\dot{x} = \frac{x^2 + 1}{\cos x} \tag{13}$$

To obtain the recurrence relation for $\dot{x}$, its expression is decomposed into auxiliary variables. In that way no nested multiplications appear in the expression of any variable. In this case, the first two auxiliary variables are the numerator and denominator of Eq. (13). Furthermore, according to Eq. (11), the recurrence relation of $\cos x$ requires the Taylor coefficients of $\sin x$, so that will be the third variable.

$$f_1 = x^2 + 1 \tag{14}$$

$$f_2 = \cos x \tag{15}$$

$$f_3 = \sin x \tag{16}$$

Next, the recurrence relations are computed. For the recurrence relation of f_1, note that the 1 is a constant, so $(1)_k = 0$ for $k \geq 1$.

$$(f_1)_k = \sum_{j=0}^{k} (x)_j (x)_{k-j} \tag{17}$$

$$(f_2)_k = -\frac{1}{k} \sum_{j=1}^{k} j (x)_j (f_3)_{k-j} \tag{18}$$

$$(f_3)_k = \frac{1}{k} \sum_{j=1}^{k} j (x)_j (f_2)_{k-j} \tag{19}$$

$$(\dot{x})_k = (f_1/f_2)_k = \frac{1}{(f_2)_0} \left[(f_1)_k - \sum_{j=1}^{k} (f_2)_j (\dot{x})_{k-j} \right] \tag{20}$$

The Taylor coefficient $(x)_{k+1}$ can then be determined from $(\dot{x})_k$ [2]:

$$(x)_{k+1} = (k+1)^{-1} (\dot{x})_k \tag{21}$$

and this formula can be used to obtain the order $k+1$ Taylor coefficients of the other variables. Once the Taylor coefficients up to order K are computed, one can determine the step size and propagate x to the next step.

C. Step-Size and Order Control

The goal of step-size control is to set the step sizes h_n as large as possible without violating error tolerances $\boldsymbol{\epsilon}_n$. If ϵ_{abs} and ϵ_{rel} are the user-set absolute and relative error tolerances, then $\boldsymbol{\epsilon}_n$ is given by:

$$\boldsymbol{\epsilon}_n = \max\left\{\epsilon_{rel}|\boldsymbol{x}_n|, \epsilon_{abs}\right\} \tag{22}$$

To determine the step size, the truncation error of a Taylor-series expansion is written as a Lagrange remainder, R:

$$\boldsymbol{R}_{K,n+1} = \frac{\boldsymbol{x}^{(K+1)}(t_n+\xi)}{(K+1)!}h^{K+1} \tag{23}$$

with $\xi \in [0, h]$. To obtain an error approximation, which uses the last two terms of the Taylor series, the Lagrange remainder of order $K-2$ is used [1]:

$$\begin{aligned}\boldsymbol{R}_{K-2,n+1} &= \frac{\boldsymbol{x}^{(K-1)}(t_n+\xi)}{(K-1)!}h^{K-1} \\ &= \frac{h^{K-1}}{(K-1)!}\left[\boldsymbol{x}^{(K-1)}(t_n) + \boldsymbol{x}^{(K)}(t_n)\xi + \cdots + \frac{\boldsymbol{x}^{(K-1+i)}(t_n)}{i!}(\xi)^i + \cdots\right]\end{aligned}$$

Taking only the first two terms of this expansion and approximating ξ by h gives:

$$\boldsymbol{R}_{K-2,n+1} \approx \frac{\boldsymbol{x}^{(K-1)}(t_n)}{(K-1)!}h^{K-1} + \frac{\boldsymbol{x}^{(K)}(t_n)}{(K-1)!}h^K \tag{24}$$

This approximation should be smaller than or equal to the error tolerances. Writing Eq. (24) in terms of Taylor coefficients gives the following error-tolerance formula:

$$|(\boldsymbol{x}_n)_{K-1}|h^{K-1} + K|(\boldsymbol{x}_n)_K|h^K \leq \boldsymbol{\epsilon}_n \tag{25}$$

From this formula, an exponential fixed-point iteration scheme was derived which converges close enough to the root of Eq. (25) in one iteration [12]:

$$h_n = \eta \exp\left[\frac{1}{K-1}\ln\left(\min\left\{\frac{\boldsymbol{\epsilon}_n}{|(\boldsymbol{x}_n)_{K-1}| + K|(\boldsymbol{x}_n)_K|h_{n,0}}\right\}\right)\right] \tag{26}$$

where η is a safety factor, set equal to 0.95. The term $h_{n,0}$ in Eq. (26) is found by ignoring the term $K(\boldsymbol{x}_n)_K h^K$ in Eq. (25), which results in:

$$h_{n,0} = \left(\min\left\{\frac{\boldsymbol{\epsilon}_n}{|(\boldsymbol{x}_n)_{K-1}|}\right\}\right)^{1/(K-1)} \tag{27}$$

With the errors handled by the step-size control, the order K can be selected to minimize the CPU time of the integration. For the order control, it was found in Ref. [14] that using a variable order does not yield significantly better results for reentry integration than using fixed orders. Hence, in Sec. V, different orders will be compared for CPU time.

III. The Reentry Problem

This section will introduce the reentry vehicle used for the simulations, HORUS, and will briefly explain its reference mission and aerodynamics, the environment models, controls, trajectory constraints and equations of motion. The recurrence relations for the equations of this section will be given in the appendix.

In this section, groundspeed, i.e., the velocity with respect to the rotating planet, is indicated by V_G and airspeed, i.e., the velocity with respect to the atmosphere, by V_A. Furthermore, all angles based on the groundspeed have the subscript G, and those based on the airspeed have the subscript A. When wind is absent, the groundspeed-based variables equal the airspeed-based ones.

A. HORUS

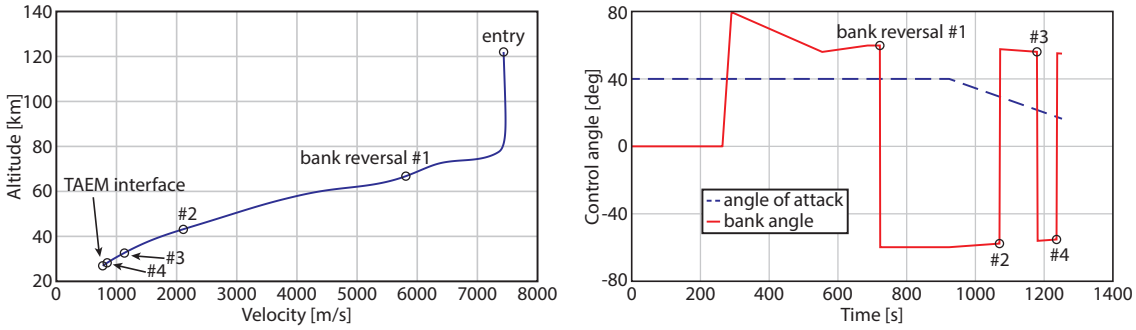
HORUS is a manned, Space-Shuttle-like reentry vehicle. The original purpose of HORUS was to serve as a reusable second stage to the Ariane-5 launcher. HORUS was later redesigned as the second stage of Sänger II, a two-stage-to-orbit concept, for which HORUS would be fitted with a rocket engine to bring it up to orbit. Both projects were canceled for budgetary reasons, but a complete aerodynamic database of HORUS (the version without rocket engine) is available [15], making it possible to be used as a reference vehicle for reentry simulations.

The relevant mission details of HORUS are shown in Table 1. HORUS' entry flight starts above the Pacific Ocean, and it ends at the runway at Kourou, French Guiana. The trajectory is split into two parts, the hypersonic entry phase and the Terminal Area Energy Management (TAEM) phase before landing. In this case, only the hypersonic phase is of interest, and it ends when the vehicle reaches a distance along the surface of the Earth of 0.75° to the target point, which is located 12

km north of the runway. This angular distance, d, is given by:

$$d = \arccos\left(\sin\delta \sin\delta_T + \cos\delta \cos\delta_T \cos(\tau - \tau_T)\right) \tag{28}$$

The altitude-velocity profile and control profile of the reference mission are shown in Fig. 1. HORUS flies an equilibrium-glide trajectory, i.e., its flight-path angle changes only very little over time. Its angle of attack is set to the maximal value (40°) for most of the time to limit the maximum heat flux. Since the large angle of attack implies a high lift force, the vehicle requires a large bank angle to prevent it from skipping and to keep it flying an equilibrium-glide trajectory. To remain flying towards the target, HORUS performs a number of bank reversals, during which the sign of the bank angle is instantaneously flipped.



(a) Altitude versus velocity

(b) Controls

Fig. 1 Reference trajectory and control profile for HORUS.

B. Environment Models

Here, the models for the shape, gravity field and atmosphere of the Earth are listed. Only analytical functions can be transformed into recurrence relations, so all data tables will be fit with analytical functions.

Planet Shape

The shape of the Earth is approximated by an ellipsoid, flattened at the poles. To determine the altitude, H, exactly above such a shape, an iterative process is required. Here, an approximation is used, given by:

$$H \approx r - R_E \sqrt{\frac{1 - e^2}{1 - e^2 \cos^2\delta}} \tag{29}$$

Table 1 Main characteristics of HORUS' reference mission [16].

Property	Value
Wing area (S_{ref}) [m^2]	110
Reentry mass [kg]	26029
Nose Radius [m]	0.8
Initial altitude [km]	122
Initial longitude [°]	-106.7
Initial latitude [°]	-22.3
Initial velocity [m/s]	7435.5
Initial flight-path angle [°]	-1.43
Initial heading [°]	70.75
Maximum dynamic pressure [N/m^2]	1×10^4
Maximum heat flux [kW/m^2]	530
Maximum g-load [-]	2.5
Terminal altitude [km]	24.86
Terminal Mach number [-]	2.5
Terminal distance [°]	0.75
Runway longitude [°]	-53.0
Runway latitude [°]	5.0

where r is distance to the center of the planet, δ is the latitude, and R_E and e are the equatorial radius and eccentricity of the Earth, which are equal to 6378.137 km and 0.081819190842, respectively, according to the World Geodetic System, 1984 [17].

Gravity Field

The gravity field of the Earth can be expressed in terms of spherical harmonics. Here, the spherical harmonics are truncated at degree 2 and order 0, retaining only the central gravity and J_2 terms. The downward and northward gravity accelerations are in that case equal to, respectively [18]:

$$g_{down} = \frac{\mu_E}{r^2}\left[1 - \frac{3}{2}J_2\left(\frac{R_E}{r}\right)^2\left(3\sin^2\delta - 1\right)\right] \tag{30}$$

$$g_{north} = -3J_2 \frac{\mu_E}{r^2} \left(\frac{R_E}{r} \right)^2 \sin\delta \cos\delta \tag{31}$$

with μ_E and J_2 equal to $3.986004415 \times 10^{14}$ $\mathrm{m}^3/\mathrm{s}^2$ and 1.0826357×10^{-3}, respectively [19].

Atmosphere

For the atmosphere modeling, the United States Standard Atmosphere 1976 (US76) [20] is used. The outputs of interest of this model are the air density, ρ, and speed of sound, a. For altitudes above 86 km, data tables are used to determine ρ. In this case, the table for ρ is fitted using exponential polynomials, given by:

$$\rho = \exp\left(\sum_{i=0}^{p} a_{\rho,i} H^i \right) \tag{32}$$

The coefficients $a_{\rho,i}$ are given in Ref. [14].

The computation of the speed of sound requires the molecular mass, M_M, which, above 86 km, also has to be obtained from tables. Here, these tables were fit up to 125 km by:

$$M_M = a_{M,0} + a_{M,1}\sin(b_{M,0}H) + a_{M,2}\cos(b_{M,0}H) + a_{M,3}\sin(b_{M,1}H) + a_{M,4}\cos(b_{M,1}H) \tag{33}$$

with the coefficients $a_{M,i}$ and $b_{M,i}$ also given in Ref. [14].

Wind Model

For the wind model, the Earth Global Reference Atmosphere model 1999 (GRAM99) was used for the raw wind data. The input of this model consists of position and time, from which the model can generate the wind-velocity components in the vertical reference frame:

$$\boldsymbol{V}_W = \begin{pmatrix} V_{W,north} & V_{W,east} & V_{W,down} \end{pmatrix}^T \tag{34}$$

GRAM99 is set to generate random wind velocities. It is sampled with the unperturbed trajectory at a rate of once every 2 km of altitude to obtain a wind profile as a function of only altitude. These profiles were then smoothed using a 5-point moving average scheme and interpolated with a cubic spline. The cubic polynomials defining the spline segments then form the analytical expressions needed for the recurrence relations. This process is repeated multiple times to obtain different random wind profiles. The result is shown in Fig. 2 for an arbitrary profile.

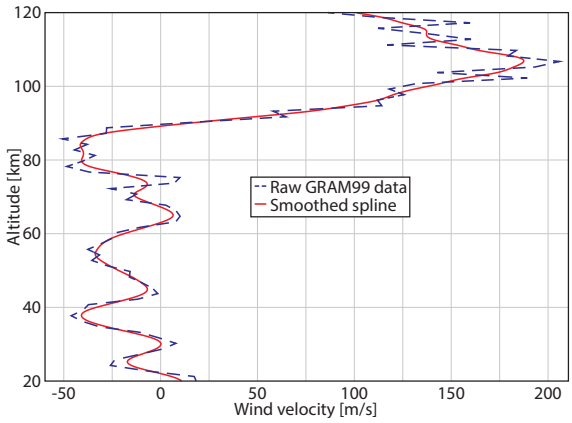


Fig. 2 Sampling and processing of the wind profile.

C. Aerodynamics

The aerodynamics database of HORUS gives the lift, drag and pitch-moment coefficients as function of angle of attack α_A, angle of side-slip β_A, Mach number M, and deflection of its body flap and elevons [15]. Here, it is assumed that the onboard guidance system keeps β_A and β_G at values close to zero, also due to the large flight velocity. Therefore, the vehicle experiences only a very small aerodynamic side-force that is successively ignored. HORUS is then pitch-trimmed for each of the data points in the range of Mach 2.5 to 20. The resulting data tables of $C_{D,trim}$ and $C_{L,trim}$ are fitted with analytical equations for the α_A range of 15° to 40°. The resulting fits are:

$$\begin{aligned} C_{D,trim} = {} & a_{D,0} + a_{D,1}\alpha_A + a_{D,2}\alpha_A^2 + a_{D,3}\alpha_A^3 + a_{D,4}M + a_{D,5}M^2 \\ & + a_{D,6}\alpha_A M + a_{D,7}\frac{\alpha_A}{M} + a_{D,8}\frac{\alpha_A^2}{M} + a_{D,9}\frac{\alpha_A}{M^2} \end{aligned} \tag{35}$$

$$\begin{aligned} C_{L,trim} = {} & a_{L,0} + a_{L,1}M + \frac{a_{L,2}}{M + a_{L,3}} + \left(a_{L,4} + \frac{a_{L,5}}{M + a_{L,6}}\right) \sin\left(a_{L,7}\alpha_A + \frac{a_{L,8}\alpha_A}{M + a_{L,9}}\right) \\ & + \left(a_{L,10} + a_{L,11}M + \frac{a_{L,12}}{M + a_{L,13}}\right) \cos\left(a_{L,7}\alpha_A + \frac{a_{L,8}\alpha_A}{M + a_{L,9}}\right) \end{aligned} \tag{36}$$

The coefficients $a_{D,i}$ and $a_{L,i}$ are given in Ref. [14]. Finally, the aerodynamic drag, side-force and lift accelerations are:

$$\begin{pmatrix} f_D \\ f_S \\ f_L \end{pmatrix} = - \begin{pmatrix} C_{D,trim} \\ 0 \\ C_{L,trim} \end{pmatrix} \frac{\rho V_A^2 S_{ref}}{2m} \tag{37}$$

D. Controls

The reentry vehicle is modeled as a point mass, so rotational dynamics are not taken into account. The attitude of the vehicle is defined by the (commanded) angle of attack, α_G and bank angle, σ_G, which in turn are defined by a control profile. These profiles are given as a set of nodes, each defined for a certain amount of specific energy E:

$$E = g_0 H + \frac{1}{2} V_G^2 \tag{38}$$

where g_0 is the gravitational acceleration at sea level, 9.81 m/s^2. The control profile is then created by interpolating the nodes with a Hermite spline, in a similar fashion as the *pchip*-function of Matlab [21]. Here, all control profiles are given by 6 nodes, of which two are fixed at the initial and terminal energy levels of HORUS, 2.884×10^7 J/kg and 5.219×10^5 J/kg, respectively.

For the control profiles, $\alpha_G \in [15°, 40°]$ and $\sigma_G \in [0°, 90°]$. The magnitude of σ_G is then determined by the control profile, and the sign is controlled by performing a bank reversal each time the heading error, χ_e, exceeds a predefined dead band, χ_{db}. A bank reversal should only be executed if the sign of σ_G is not correct already, which can be expressed in mathematical terms as:

$$-\chi_e \operatorname{sign}(\sigma_G) > \chi_{db} \tag{39}$$

with χ_e defined as $\chi_G - \chi_T$, where χ_G is the (groundspeed-based) heading, and χ_T is the heading to target, given by [16]:

$$\chi_T = \operatorname{atan2}\left(\sin(\tau_T - \tau), \tan\delta_T \cos\delta - \cos(\tau_T - \tau)\sin\delta\right) \tag{40}$$

where τ is the latitude, and τ_T and δ_T are the longitude and latitude of the target. The definition of the dead band for HORUS is given in Ref. [16].

E. The Inclusion of Wind

In this section, it is explained how the wind velocities of Eq. (34) impact the aerodynamic accelerations computed with Eq. (37). The first step is to compute the groundspeed vector components in the vertical frame:

$$\boldsymbol{V}_G = \begin{pmatrix} V_G \cos\chi_G \cos\gamma_G & V_G \sin\chi_G \cos\gamma_G & -V_G \sin\gamma_G \end{pmatrix}^T \tag{41}$$

where γ_G is the (groundspeed-based) flight-path angle. The airspeed vector is computed by subtracting the wind-velocity vector:

$$\boldsymbol{V}_A = \boldsymbol{V}_G - \boldsymbol{V}_W \tag{42}$$

From the components of the airspeed, one can obtain the airspeed-based angles γ_A and χ_A. The next step is to compute the transformation matrix $\boldsymbol{C}_{B,TA}$, i.e., the matrix for the transformation from the airspeed-based trajectory frame (subscript TA) to the body frame (subscript B), assuming that β_G is zero:

$$\boldsymbol{C}_{B,TA} = \boldsymbol{C}_y(\alpha_G)\boldsymbol{C}_x(-\sigma_G)\boldsymbol{C}_y(\gamma_G)\boldsymbol{C}_z(\chi_G - \chi_A)\boldsymbol{C}_y(-\gamma_A) \tag{43}$$

with $\boldsymbol{C}_i(\theta)$ being a unit-axis transformation matrix of an angle θ about axis i. An explanation of transformation matrices and the reference frames used is given by Mooij [18].

$\boldsymbol{C}_{B,TA}$ can also be computed using the angles α_A, β_A and σ_A:

$$\boldsymbol{C}_{B,TA} = \begin{bmatrix} \mathrm{c}\alpha_A\mathrm{c}\beta_A & (\mathrm{s}\alpha_A\mathrm{s}\sigma_A - \mathrm{c}\alpha_A\mathrm{s}\beta_A\mathrm{c}\sigma_A) & (\mathrm{c}\alpha_A\mathrm{s}\beta_A\mathrm{s}\sigma_A - \mathrm{s}\alpha_A\mathrm{c}\sigma_A) \\ \mathrm{s}\beta_A & \mathrm{c}\beta_A\mathrm{c}\sigma_A & -\mathrm{c}\beta_A\mathrm{s}\sigma_A \\ \mathrm{s}\alpha_A\mathrm{c}\beta_A & (\mathrm{c}\alpha_A\mathrm{s}\sigma_A - \mathrm{s}\alpha_A\mathrm{s}\beta_A\mathrm{c}\sigma_A) & (\mathrm{c}\alpha_A\mathrm{c}\sigma_A + \mathrm{s}\alpha_A\mathrm{s}\beta_A\mathrm{s}\sigma_A) \end{bmatrix} \tag{44}$$

where s and c indicate the sine and cosine of each angle. From Eq. (44), one can obtain expressions for the angles α_A, β_A and σ_A:

$$\sin\alpha_A = \frac{\boldsymbol{C}_{B,TA}(3,1)}{\cos\beta_A} \qquad \cos\alpha_A = \frac{\boldsymbol{C}_{B,TA}(1,1)}{\cos\beta_A} \tag{45}$$

$$\sin\beta_A = \boldsymbol{C}_{B,TA}(2,1) \qquad \cos\beta_A = \sqrt{\boldsymbol{C}_{B,TA}(2,2)^2 + \boldsymbol{C}_{B,TA}(2,3)^2} \tag{46}$$

$$\sin\sigma_A = -\frac{\boldsymbol{C}_{B,TA}(2,3)}{\cos\beta_A} \qquad \cos\sigma_A = \frac{\boldsymbol{C}_{B,TA}(2,2)}{\cos\beta_A} \tag{47}$$

where the matrix coefficients on the right-hand side are provided by Eq. (43). α_A can be used to calculate the aerodynamic accelerations. Note that these accelerations are given in an airspeed based reference frame, whereas the equations of motion, which are given in the next section, are defined in the groundspeed-based trajectory frame (subscript TG). So the final step is to transform

the aerodynamic accelerations to this frame:

$$\begin{pmatrix} f_{D,TG} \\ f_{S,TG} \\ f_{L,TG} \end{pmatrix} = \boldsymbol{C}_y(\gamma_G)\boldsymbol{C}_z(\chi_G - \chi_A)\boldsymbol{C}_y(-\gamma_A)\boldsymbol{C}_x(\sigma_A) \begin{pmatrix} f_D \\ f_S \\ f_L \end{pmatrix} \tag{48}$$

Without wind, the groundspeed-based and airspeed-based angles are equal, simplifying this transformation to:

$$\begin{pmatrix} f_{D,TG} \\ f_{S,TG} \\ f_{L,TG} \end{pmatrix} = \boldsymbol{C}_x(\sigma_G) \begin{pmatrix} f_D \\ f_S \\ f_L \end{pmatrix} \tag{49}$$

F. Equations of Motion

For the translational motion, different state-variable sets have been compared for the average CPU time when integrating with TSI, including Cartesian and spherical state variables, and the combination of spherical position with Cartesian velocity. The outcome of this comparison was that spherical state variables are the fastest for TSI. The equations of motion for this set are [18]:

$$\dot{r} = V_G \sin\gamma_G \tag{50}$$

$$\dot{\tau} = \frac{V_G \sin\chi_G \cos\gamma_G}{r\cos\delta} \tag{51}$$

$$\dot{\delta} = \frac{V_G \cos\chi_G \cos\gamma_G}{r} \tag{52}$$

$$\dot{V}_G = f_{D,TG} - g_{down}\sin\gamma_G + g_{north}\cos\gamma_G\cos\chi_G + \omega_E^2 r\cos\delta\left(\sin\gamma_G\cos\delta - \cos\gamma_G\sin\delta\cos\chi_G\right) \tag{53}$$

$$\begin{aligned} V_G\dot{\gamma}_G = &-f_{L,TG} - g_{down}\cos\gamma_G - g_{north}\sin\gamma_G\cos\chi_G + 2\omega_E V_G\cos\delta\sin\chi_G \\ &+\frac{V_G^2}{r}\cos\gamma_G + \omega_E^2 r\cos\delta\left(\cos\delta\cos\gamma_G + \sin\gamma_G\sin\delta\cos\chi_G\right) \end{aligned} \tag{54}$$

$$\begin{aligned} V_G\cos\gamma_G\dot{\chi}_G = &f_{S,TG} - g_{north}\sin\chi_G + 2\omega_E V_G\left(\sin\delta\cos\gamma_G - \cos\delta\sin\gamma_G\cos\chi_G\right) \\ &+\frac{V_G^2}{r}\cos^2\gamma_G\tan\delta\sin\chi_G + \omega_E^2 r\cos\delta\sin\delta\sin\chi_G \end{aligned} \tag{55}$$

The reader is referred to the appendix for the complete formulation of the TSI model with recurrence relations.

G. Trajectory Constraints

In this research, three different trajectory constraints are considered. The first is the convective heat flux, $\dot{q}_c$, given by [16]

$$\dot{q}_c = \frac{C_1}{\sqrt{R_N}}\sqrt{\rho}V_A^{3.15} \leq 5.3\times10^5 \text{ W/m}^2 \tag{56}$$

with R_N and C_1 equal to 0.8 m and 5.28137×10^{-5} $\text{Js}^{2.15}/\text{kg}^{0.5}/\text{m}^{2.15}$, respectively. The other two constraints are the load factor, n_g and the dynamic pressure, $\bar{q}$:

$$n_g = \frac{\sqrt{f_D^2 + f_L^2}}{g_0} \leq 2.5 \tag{57}$$

$$\bar{q} = \frac{1}{2}\rho V_A^2 \leq 10^4 \text{ N/m}^2 \tag{58}$$

In Ref. [14], it is explained how to incorporate constraint-violation penalties in TSI. This explanation includes a mechanism for finding constraint violations, even when the constraint is only violated briefly during a time step and the violation cannot be detected by only checking the values of the constraint variables at the start and end of a step.

IV. Discontinuities

A Taylor-series expansion before a discontinuity contains no information on the new equation set after the discontinuity. Thus TSI will likely accumulate errors if there is a discontinuity somewhere during a time step. The solution is to reduce the step size such that the step ends at the discontinuity, and the next step can be performed with the new set of equations. This practice will here be called the *G-stop* procedure.

The reentry equations of the previous section contains six types of discontinuities: i) the atmosphere layers of US76, which are determined by the current altitude, ii) the spline segments with which the wind profiles are defined for each 2 km of altitude, iii) the control profile, which is a piecewise function of E, iv) the regression fits of the aerodynamics, which are only defined in the range of Mach 2.5 to 20, v) bank reversals, and vi) reaching the terminal distance. The G-stop procedure then involves finding the root of a function g, for instance, for a US76 layer change at altitude H_{layer}:

$$g = H - H_{layer} \tag{59}$$

A. Altitude, Mach Number and Energy

Of the variables H, M and E, the Taylor coefficients are available at the end of an integration step, as they are needed to propagate the state. To find the root of function g, the Taylor series of its reciprocal $1/g$ is computed. The poles of this function are the roots of g, so an estimate of the radius of convergence of $1/g$ is also an approximation of the nearest root of g. The Taylor coefficients of $1/g$ can be computed with Eq. (7), and the radius of convergence r is then found using the ratio test [8]:

$$r = \frac{(1/g)_{K-1}}{(1/g)_K} \tag{60}$$

The procedure for finding a root in the search space $[t_n, t_n + h]$ is as follows [8]:

1. Initially, the expansion point t_{exp} is equal to t_n, and multiplication factor η_{step} is equal to 0.8.

2. Generate the Taylor coefficients of $1/g$ at t_{exp} and compute its radius of convergence r. Set h_{step} equal to $\eta_{step} r$.

3. If $r < 0$, the nearest root lies before the expansion point. In that case, set h_{step} equal to $0.6(h - t_{exp})$ to force the method to look for a root within the search space, and skip step 4. If $r > h - t_{exp}$ (which, during simulation, only happened when a root was very close to h), set h_{step} equal to $0.95(h - t_{exp})$ and skip step 4.

4. The estimate of the root is equal to $t_0 + r$, so the method has converged if $|g(t_0 + r)| \leq \epsilon_{abs}$.

5. While $t_0 + h_{step}$ is located behind the root, reduce the size of h_{step} by a factor 0.25.

6. Re-expand the Taylor series of g at $t_0 + h_{step}$ and set t_0 equal to $t_0 + h_{step}$. Set η_{step} equal to $\sqrt{\eta_{step}}$ and go to step 2.

It was found that for H, M and E, one should not aim for the root, but rather slightly beyond it, to avoid cases where the root-finding method is triggered again or the integrator ends up in an infinite loop. The values of these "safety distances" for H, M and E are 10^{-9} m, $M \times \epsilon_{rel}$ and $E \times \epsilon_{rel}$, respectively.

B. Bank Reversals

For bank reversals, the function g of which the root is to be found is obtained from Eq. (39):

$$g = -\text{sign}(\sigma)\chi_e - \chi_{db} \tag{61}$$

Since the Taylor coefficients of χ_e and χ_{db} are not needed for the rest of the integration process, the Muller method [22] is used to find the roots. For this method, one needs to sample g at three points $t_{0,i}$, $t_{1,i}$ and $t_{2,i}$ and then fit a line through these points, given by:

$$\hat{g}_i(h) = a_i h^2 + b_i h + f_{0,i} \tag{62}$$

with h being the step size measured from $t_{0,i}$. b_i and a_i are the first and second derivatives of the line at $t_{0,i}$, respectively. They are given by:

$$a_i = \frac{1}{t_{2,i} - t_{1,i}} \left(\frac{f_{2,i} - f_{0,i}}{t_{2,i} - t_{0,i}} - \frac{f_{1,i} - f_{0,i}}{t_{1,i} - t_{0,i}} \right) \tag{63}$$

$$b_i = \frac{f_{1,i} - f_{0,i}}{t_{1,i} - t_{0,i}} - a_i(t_{1,i} - t_{0,i}) \tag{64}$$

Of $\hat{g}$, only the positive root is needed, as a bank reversal is only needed when g is larger than zero, in which case the left values are always negative, and the right are always positive. The positive root is:

$$t_{new} = t_{0,i} + \frac{-b_i + \sqrt{b_i^2 + 4a_i f_{0,i}}}{2a_i} \tag{65}$$

To find the value for g for a particular value of h, one first computes the state at $t_{0,i} + h$, using its Taylor coefficients, and one then computes g. The root-finding procedure for bank reversals is:

1. Initially, the value at the start of a time step and at the end are used, so $t_{0,0} = t_n$ and $t_{2,0} = t_n + h_n$. The middle point is found using the false position method [22] with the two outer points.

2. Using Eq. (65), a new estimate for the root, t_{new}, can be found.

3. The method has converged if $|f(t_{new})| \leq \epsilon_{abs}$.

4. t_{new} will become the new middle point, $t_{1,i+1}$, so if it is larger than $t_{1,i}$, $t_{0,i+1} = t_{1,i}$, else $t_{2,i+1} = t_{1,i}$. Go to step 2.

The sign of σ is flipped when the root-finding is done, which means that Eq. (39) is no longer satisfied and the root-finding will not be triggered for this particular bank reversal again, and the integrator cannot end up in an infinite loop. Therefore, the root-finding for bank reversals is allowed to target the root precisely, i.e., there is no safety distance needed.

C. Terminal Distance

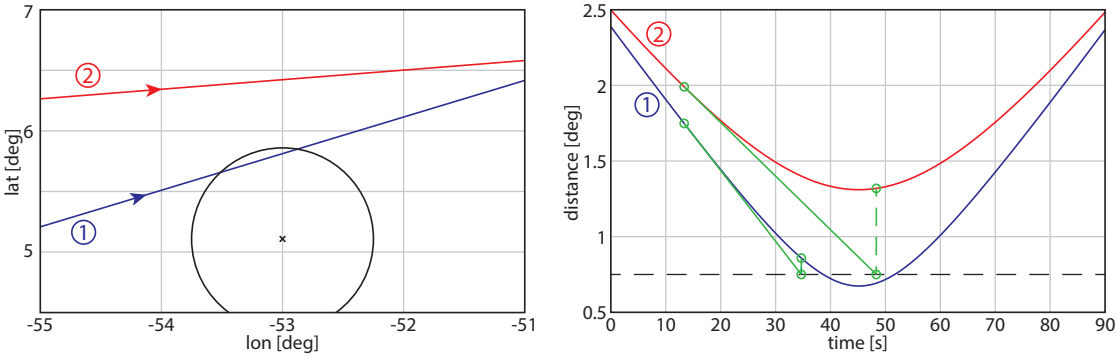
The integration of a trajectory can be stopped when the terminal distance to target is reached, but also when the vehicle misses the target. To find both cases and distinguish between them, Newton's method [22] is used:

$$h_{i+1} = h_i - \frac{d(h_i) - d_{terminal}}{\dot{d}(h_i)} \tag{66}$$

This method is used because the distance to target for trajectories that miss the TAEM interface first decreases asymptotically, then increases asymptotically, as is shown in Fig. 3. In this figure, one iteration of Newton's method for trajectory 1 stays on the left of the minimum, but for trajectory 2, the method eventually (in this case in one iteration) ends up on the right of the minimum. One can determine whether the method ends up on the left or the right by evaluating the sign of the first derivative of the distance, given by:

$$\dot{d} = \frac{\left(\dot{\delta}\sin\delta\cos(\tau_T - \tau) - \dot{\tau}\sin(\tau_T - \tau)\cos\delta\right)\cos\delta_T - \dot{\delta}\cos\delta\sin\delta_T}{\sin d} \tag{67}$$

To compute the values of d and $\dot{d}$, one first computes the state, just like for the bank reversals.



(a) Longitude versus latitude

(b) Distance to target

Fig. 3 Two high-speed trajectories and their distance to target.

V. Simulation Results

In this section, the results of simulations with TSI and RKF5(6) are presented. Both integrators were developed in C++, and the simulations were performed on a HP® EliteBook 8540w, with an Intel® Core™ i7 clocked at 1.60 GHz. First the optimal orders for TSI are determined, followed by a comparison of the integration results of TSI and RKF5(6). Finally, the CPU times of the two integrators are compared. For all simulations, the relative and absolute error tolerances have identical values. It is noted that the RKF integrator has been optimized as much as possible for this problem, i.e., the most optimal state-variable set has been selected for this integrator and the integrator has been programmed as efficiently as possible, so no general-purpose integration package has been used.

A. Optimal Order Control

To determine the optimal fixed orders for integrating reentry trajectories, TSI was timed for batches of 5,000 random trajectories with orders varying from 5 to 25 and error tolerances ranging from 10^{-5} to 10^{-14}. The random trajectories were created by randomizing the controls and wind profiles. The optimal orders and their respective times for each error tolerance are given in Table 2. As can be seen, the optimal order mostly increases as the tolerance decreases. Furthermore, the CPU times are higher for integration with wind, and the optimal orders are lower, mostly because the wind profiles are defined by spline segments for every 2 km of altitude. As a result the integration has a discontinuity every 2 km of altitude, which both decreases the step sizes the integrator can take and increases the number of times root-finding has to be used to locate each discontinuity.

B. Comparison of Integration Results

Here, the results of the two integrators are compared to establish that the integrators indeed yield results with small relative differences when they are applied to the same reentry problem. The integrators were set to $\epsilon = 10^{-14}$ and the comparison was performed for a single trajectory, which was the result of an optimization of the flight to the TAEM interface and had the lowest heat load.

Table 2 Optimal orders and their average time to integrate 5,000 trajectories.

	without wind		with wind	
ϵ	order	CPU time [s]	order	CPU time [s]
10^{-5}	8	1.491	6	5.358
10^{-6}	8	1.785	6	5.976
10^{-7}	10	2.187	8	6.656
10^{-8}	10	2.596	8	7.309
10^{-10}	12	3.463	10	9.157
10^{-12}	16	4.527	11	12.010
10^{-14}	18	5.891	13	15.821

The integrated heat load Q is:

$$Q = \int_0^{t_f} \dot{q}_c dt \tag{68}$$

The relative differences between a number of parameters at the end of flight for the two integrators are shown in Table 3 (relative difference here means $(Value_{RKF} - Value_{TSI})/Value_{TSI}$) for four different wind profiles. As can be seen, some of the differences have an order of magnitude of 10^{-5}, which is large compared to the error tolerance.

Table 3 Relative differences between the final conditions found with TSI and RKF5(6).

Trajectory	H [-]	τ [-]	δ [-]	V_G [-]	Q [-]
1	-2.242×10^{-5}	-5.157×10^{-7}	-5.431×10^{-6}	-7.031×10^{-5}	4.197×10^{-7}
2	5.387×10^{-6}	-6.160×10^{-8}	-1.033×10^{-6}	8.524×10^{-6}	-1.860×10^{-7}
3	-6.280×10^{-6}	3.763×10^{-8}	5.505×10^{-7}	-1.283×10^{-5}	2.753×10^{-8}
4	-3.808×10^{-6}	-2.370×10^{-7}	-2.908×10^{-6}	-1.122×10^{-5}	4.059×10^{-8}

To show the main cause for the large differences in Table 3, the unperturbed trajectory is integrated with integrators set to $\epsilon = 10^{-15}$. The relative differences in altitude and velocity are shown by the dashed and solid lines in Fig. 4. The most noticeable features of these lines are the jumps in the solid line and certain points at which both lines suddenly increase. As has been marked in the graph, these events coincide with discontinuities such as US76 layer changes and bank

reversals. These jumps are removed when a G-stop facility is also build into RKF5(6), which was in this case done using the Muller method that was described in Sec. IV B. The result is shown by the dotted line in Fig. 4; the difference in velocity is now less than 10^{-12} at the end of the trajectory.

Finally, the differences of Table 3 were recomputed with the G-stop procedure for RKF5(6), the results of which are shown in Table 4. As can be seen, the differences are now of magnitude 10^{-11} or smaller, from which it can be concluded that the two integrators yield very similar results. Thus, when computing reentry trajectories with high accuracy, a G-stop facility should also be added to RKF5(6).

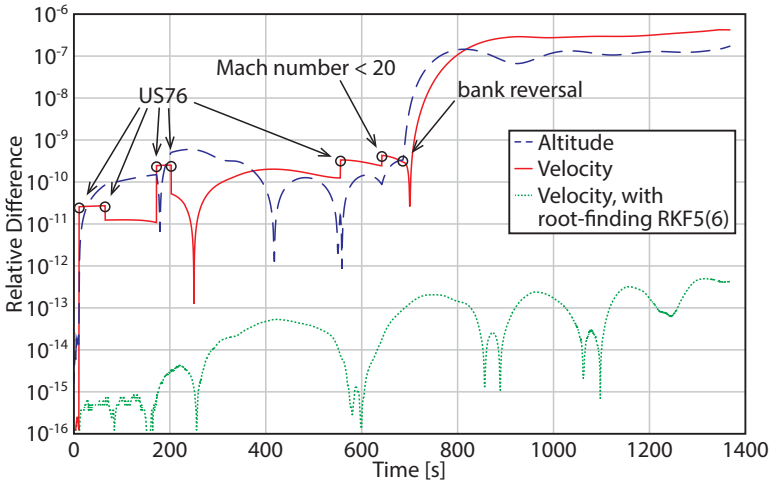


Fig. 4 Differences in H and V between integrations with RKF5(6) and TSI set to $\epsilon = 10^{-15}$.

Table 4 Relative differences between the final conditions found with TSI and RKF5(6) with G-stop facility.

Trajectory	H [-]	τ [-]	δ [-]	V_G [-]	Q [-]
1	-4.384×10^{-12}	6.456×10^{-14}	5.805×10^{-13}	-1.143×10^{-11}	-9.198×10^{-14}
2	-7.266×10^{-12}	5.481×10^{-14}	7.861×10^{-13}	-1.229×10^{-11}	-1.764×10^{-14}
3	-6.305×10^{-12}	6.350×10^{-14}	7.624×10^{-13}	-1.192×10^{-11}	-5.129×10^{-14}
4	-3.016×10^{-12}	1.532×10^{-14}	2.041×10^{-13}	-6.114×10^{-12}	2.085×10^{-14}

C. Comparison of Integration Speed

Finally, the CPU times of the two integrators are compared. The state variable sets used for RKF5(6) are Cartesian for integration without wind and spherical for integration with wind, as these settings yielded the highest performance for this integrator. RKF5(6) was also timed for batches of 5,000 random trajectories and the results are shown in Table 5. As can be concluded from this table, integration with wind does not cause a large increase in CPU times for RKF5(6), as it does not take into account the discontinuities. As a result it is almost as fast as TSI for integration with wind and an error tolerance of 10^{-8} and faster for higher tolerances.

The CPU times when discontinuities are taken into account can be seen in Table 6. The Muller method was used for root-finding for RKF5(6), the reason being that it always converged in about 4 steps, which is considered to be close to optimal. As can be seen in Table 6, RKF5(6) is especially penalized for high tolerances, whereas for low tolerances without wind, root-finding is faster, because the integrator no longer has to check in the state-derivative function in which atmosphere layer it is and whether it should execute a bank reversal. Furthermore, bank reversals tend to cause the integrator to reject the step size a few times, because of the large discontinuity in derivatives. In the case of wind, there are many discontinuities, adding 50 to 60 seconds to the integration time. Overall, including a G-stop facility in RKF5(6) makes it several times slower than TSI.

Table 5 CPU times of RKF5(6) for 5,000 trajectories.

	without wind		with wind	
ϵ	CPU time RKF [s]	CPU time ratio $\frac{\text{RKF}}{\text{TSI}}$	CPU time RKF [s]	CPU time ratio $\frac{\text{RKF}}{\text{TSI}}$
10^{-5}	3.276	2.20	3.603	0.67
10^{-6}	4.228	2.37	4.817	0.81
10^{-7}	5.819	2.66	6.374	0.96
10^{-8}	8.034	3.09	8.596	1.18
10^{-10}	16.21	4.68	17.33	1.89
10^{-12}	32.93	7.27	35.97	3.00
10^{-14}	70.57	11.98	75.72	4.79

Table 6 CPU times of RKF5(6) for 5,000 trajectories with G-stop facility.

	without wind		with wind	
ϵ	CPU time RKF [s]	CPU time ratio $\frac{\text{RKF}}{\text{TSI}}$	CPU time RKF [s]	CPU time ratio $\frac{\text{RKF}}{\text{TSI}}$
10^{-5}	6.162	4.13	56.03	10.46
10^{-6}	7.119	3.99	59.72	9.99
10^{-7}	8.299	3.79	61.42	9.23
10^{-8}	9.927	3.82	63.94	8.75
10^{-10}	15.46	4.46	69.96	7.64
10^{-12}	28.41	6.28	95.89	7.98
10^{-14}	57.54	9.77	139.2	8.80

VI. Conclusion

A Taylor Series Integration (TSI) reentry simulator was developed to determine whether TSI is faster than traditional integration methods for reentry applications with wind. Discrete data tables of environment and aerodynamic models were fitted with regression lines to obtain analytical approximations, as TSI can only handle analytical expressions. To cope with discontinuites in the equations of motion, time steps were shortened whenever TSI integrates over a discontinuity, so that TSI does not accumulate errors by integrating a function that is no longer valid after the discontinuity. To optimize the CPU times for TSI, the optimal integration orders were determined for different error tolerances.

TSI was compared with the fifth-order Runge-Kutta-Fehlberg (RKF) for CPU times and was found to be between 2.20 and 11.98 times faster for integration without wind, and between 0.67 and 4.79 times faster for integration with wind. The lesser performance for the cases with wind is mostly due to the fact that the way the wind profiles are defined drastically increases the number of discontinuities and TSI has to adapt its step sizes for each of these discontinuities, whereas the RKF method does not. However, when comparing the results of the integrators for low error tolerances, it was found that RKF methods can also accumulate significant errors when integrating over discontinuities. Thus a G-stop facility is also need for RKF methods when integrating reentry

trajectories with high accuracy, albeit that this is not common practice. In that case, TSI is 3.79 to 9.77 times faster for integration without wind and 7.64 to 10.46 times faster for integration with wind.

The use of TSI is advised for reentry applications such as integration, optimization and sensitivity analysis, as long as the application does not involve a guidance/navigation/control system that limits the step sizes to very small values. Otherwise, TSI cannot use a high order and large step sizes, which are the main reasons it can outperform other numerical integrators.

Appendix: Reentry Recurrence Relations

In the appendix of Ref. [14], the recurrence relations for the reentry equations of motion without wind were given. With these relations one could create his own TSI reentry simulator. In this appendix, the relations for the inclusion of wind are added. Similar to the process in Sec. II B, the equations of interest are first decomposed into auxiliary variables and then the recurrence relations for these variables are given. Sines and cosines are here denoted by s and c, respectively.

Auxiliary Variables

The appendix of Ref. [14] starts with the computation of the sines and cosines of δ, γ_G and χ_G, which are the first auxiliary variables. Next, the variables for the gravity field model, planet shape and atmosphere model are listed, followed by those for the controls (α_G and σ_G).

Now, before computing the aerodynamic accelerations, the variables for the inclusion of wind are computed. To distinguish these variables from those in Ref. [14], the auxiliary variables here are labeled with the numbers 40 to 62.

$$f_{40} = V_G \mathrm{c}\gamma_G - V_{W,north} \mathrm{c}\chi_G - V_{W,east} \mathrm{s}\chi_G \tag{69}$$

$$f_{41} = V_{W,north} \mathrm{s}\chi_G - V_{W,east} \mathrm{c}\chi_G \tag{70}$$

$$f_{42} = -V_G \mathrm{s}\gamma_G - V_{W,down} \tag{71}$$

$$f_{43} = f_{40}^2 + f_{41}^2 \qquad f_{44} = \sqrt{f_{43}} \tag{72}$$

$$f_{45} = f_{43} + f_{42}^2 \qquad V_A = \sqrt{f_{45}} \tag{73}$$

$$f_{46} = -\frac{f_{41}}{f_{44}} \qquad f_{47} = \frac{f_{40}}{f_{44}} \qquad \mathrm{s}\gamma_A = -\frac{f_{42}}{V_A} \qquad \mathrm{c}\gamma_A = \frac{f_{44}}{V_A} \tag{74}$$

$$f_{48} = \mathrm{s}\gamma_G \mathrm{s}\gamma_A \tag{75}$$

$$f_{49} = \mathrm{s}\gamma_G \mathrm{c}\gamma_A \tag{76}$$

$$f_{50} = \mathrm{c}\gamma_G \mathrm{c}\gamma_A \tag{77}$$

$$f_{51} = \mathrm{c}\gamma_G \mathrm{s}\gamma_A \tag{78}$$

$$f_{52} = f_{47} f_{49} - f_{51} \tag{79}$$

$$f_{53} = -f_{46} \mathrm{c}\gamma_A \tag{80}$$

$$f_{54} = f_{46} \mathrm{s}\gamma_G \tag{81}$$

$$f_{55} = f_{48} + f_{47} f_{50} \tag{82}$$

$$f_{56} = f_{50} + f_{47} f_{48} \tag{83}$$

$$f_{57} = -f_{46} \mathrm{s}\gamma_A \tag{84}$$

$$f_{58} = f_{52} \mathrm{c}\sigma_G + f_{53} \mathrm{s}\sigma_G \tag{85}$$

$$\mathrm{s}\beta_A = f_{53} \mathrm{c}\sigma_G - f_{52} \mathrm{s}\sigma_G \tag{86}$$

$$\mathrm{c}\beta_A = \sqrt{1 - (\mathrm{s}\beta_A)^2} \tag{87}$$

$$\mathrm{s}\alpha_A = \frac{f_{55} \mathrm{s}\alpha_G + f_{58} \mathrm{c}\alpha_G}{\mathrm{c}\beta_A} \tag{88}$$

$$c\alpha_A = \sqrt{1 - (s\alpha_A)^2} \tag{89}$$

$$s\sigma_A = \frac{f_{56}s\sigma_G - f_{57}c\sigma_G}{c\beta_A} \tag{90}$$

$$c\sigma_A = \frac{f_{47}c\sigma_G + f_{54}s\sigma_G}{c\beta_A} \tag{91}$$

$$\alpha_A = \arcsin(s\alpha_A) \tag{92}$$

Now that α_A is known, f_D and f_L can then be computed as is done in the section "Aerodynamic Accelerations" of the appendix of Ref. [14], but using α_A instead of α. Next, the aerodynamic accelerations are transformed to the groundspeed-based trajectory frame before computing the translational equations of motion.

$$f_{59} = c\gamma_G f_{46} \tag{93}$$

$$f_{60} = f_{52}f_{47} - f_{49} \tag{94}$$

$$f_{61} = f_L s\sigma_A \tag{95}$$

$$f_{62} = f_L c\sigma_A \tag{96}$$

$$f_{D,TG} = f_D f_{55} + f_{61}f_{53} + f_{62}f_{52} \tag{97}$$

$$f_{S,TG} = f_D f_{59} + f_{61}f_{47} + f_{62}f_{54} \tag{98}$$

$$f_{L,TG} = f_D f_{60} - f_{61}f_{57} + f_{62}f_{56} \tag{99}$$

The equations for the translational equations of motion are the same as in the appendix of Ref. [14], safe for those of f_{34}, f_{35} and $\dot{V}_G$, which are now a function of $f_{D,TG}$, $f_{S,TG}$ and $f_{L,TG}$:

$$f_{34} = \frac{-f_{L,TG} - g_{down}c\gamma_G - g_{north}f_{30} + f_{29}f_{32}}{V_G} \tag{100}$$

$$f_{35} = \frac{f_{S,TG} + f_{33}s\chi_G}{V_G} \tag{101}$$

$$\dot{V}_G = f_{D,TG} - g_{down}s\gamma_G + g_{north}f_{28} + f_{29}f_{31} \tag{102}$$

Recurrence Relations

$$(f_{40})_k = \sum_{j=0}^{k} (V_G)_j (\mathrm{c}\gamma_G)_{k-j} - (V_{W,north})_j (\mathrm{c}\chi_G)_{k-j} - (V_{W,east})_j (\mathrm{s}\chi_G)_{k-j} \tag{103}$$

$$(f_{41})_k = \sum_{j=0}^{k} (V_{W,north})_j (\mathrm{s}\chi_G)_{k-j} - (V_{W,east})_j (\mathrm{c}\chi_G)_{k-j} \tag{104}$$

$$(f_{42})_k = -(V_{W,down})_k - \sum_{j=0}^{k} (V_G)_j (\mathrm{s}\gamma_G)_{k-j} \tag{105}$$

$$(f_{43})_k = \sum_{j=0}^{k} (f_{40})_j (f_{40})_{k-j} + (f_{41})_j (f_{41})_{k-j} \tag{106}$$

$$(f_{44})_k = \frac{1}{2(f_{44})_0} \left[(f_{43})_k - \sum_{j=1}^{k-1} (f_{44})_j (f_{44})_{k-j} \right] \tag{107}$$

$$(f_{45})_k = (f_{43})_k + \sum_{j=0}^{k} (f_{42})_j (f_{42})_{k-j} \tag{108}$$

$$(V_A)_k = \frac{1}{2(V_A)_0} \left[(f_{45})_k - \sum_{j=1}^{k-1} (V_A)_j (V_A)_{k-j} \right] \tag{109}$$

$$(f_{46})_k = \frac{-1}{(f_{44})_0} \left[(f_{41})_k + \sum_{j=1}^{k} (f_{44})_j (f_{46})_{k-j} \right] \tag{110}$$

$$(f_{47})_k = \frac{1}{(f_{44})_0} \left[(f_{40})_k - \sum_{j=1}^{k} (f_{44})_j (\mathrm{c}\chi_A)_{k-j} \right] \tag{111}$$

$$(\mathrm{s}\gamma_A)_k = -\frac{1}{(V_A)_0} \left[(f_{42})_k + \sum_{j=1}^{k} (V_A)_j (\mathrm{s}\gamma_A)_{k-j} \right] \tag{112}$$

$$(\mathrm{c}\gamma_A)_k = \frac{1}{(V_A)_0} \left[(f_{44})_k - \sum_{j=1}^{k} (V_A)_j (\mathrm{c}\gamma_A)_{k-j} \right] \tag{113}$$

$$(f_{48})_k = \sum_{j=0}^{k} (\mathrm{s}\gamma_G)_j (\mathrm{s}\gamma_A)_{k-j} \tag{114}$$

$$(f_{49})_k = \sum_{j=0}^{k} (\mathrm{s}\gamma_G)_j (\mathrm{c}\gamma_A)_{k-j} \tag{115}$$

$$(f_{50})_k = \sum_{j=0}^{k} (\mathrm{c}\gamma_G)_j (\mathrm{c}\gamma_A)_{k-j} \tag{116}$$

$$(f_{51})_k = \sum_{j=0}^{k} (\mathrm{c}\gamma_G)_j (\mathrm{s}\gamma_A)_{k-j} \tag{117}$$

$$(f_{52})_k = \sum_{j=0}^{k} (f_{47})_j (f_{49})_{k-j} - (f_{51})_k \tag{118}$$

$$(f_{53})_k = -\sum_{j=0}^{k} (f_{46})_j (\mathrm{c}\gamma_A)_{k-j} \tag{119}$$

$$(f_{54})_k = \sum_{j=0}^{k} (f_{46})_j (\mathrm{s}\gamma_G)_{k-j} \tag{120}$$

$$(f_{55})_k = f_{48} + \sum_{j=0}^{k} (f_{47})_j (f_{50})_{k-j} \tag{121}$$

$$(f_{56})_k = f_{50} + \sum_{j=0}^{k} (f_{47})_j (f_{48})_{k-j} \tag{122}$$

$$(f_{57})_k = -\sum_{j=0}^{k} (f_{46})_j (\mathrm{s}\gamma_A)_{k-j} \tag{123}$$

$$(f_{58})_k = \sum_{j=0}^{k} (f_{52})_j (\mathrm{c}\sigma_G)_{k-j} + (f_{53})_j (\mathrm{s}\sigma_G)_{k-j} \tag{124}$$

$$(\mathrm{s}\beta_A)_k = \sum_{j=0}^{k} (f_{53})_j (\mathrm{c}\sigma_G)_{k-j} + (f_{52})_j (\mathrm{s}\sigma_G)_{k-j} \tag{125}$$

$$(\mathrm{c}\beta_A)_k = \frac{-1}{2(\mathrm{c}\beta_A)_0} \left[\sum_{j=0}^{k} (\mathrm{s}\beta_A)_j (\mathrm{s}\beta_A)_{k-j} + \sum_{j=1}^{k-1} (\mathrm{c}\beta_A)_j (\mathrm{c}\beta_A)_{k-j} \right] \tag{126}$$

$$(\mathrm{s}\alpha_A)_k = \frac{1}{(\mathrm{c}\beta_A)_0} \left\{ \sum_{j=0}^{k} \left[(f_{55})_j (\mathrm{s}\alpha_G)_{k-j} - (f_{58})_j (\mathrm{c}\alpha_G)_{k-j} \right] - \sum_{j=1}^{k} (\mathrm{c}\beta_A)_j (\mathrm{s}\alpha_A)_{k-j} \right\} \tag{127}$$

$$(\mathrm{c}\alpha_A)_k = \frac{-1}{2(\mathrm{c}\alpha_A)_0}\left[\sum_{j=0}^{k}(\mathrm{s}\alpha_A)_j(\mathrm{s}\alpha_A)_{k-j} + \sum_{j=1}^{k-1}(\mathrm{c}\alpha_A)_j(\mathrm{c}\alpha_A)_{k-j}\right] \tag{128}$$

$$(\mathrm{s}\sigma_A)_k = \frac{1}{(\mathrm{c}\beta_A)_0}\left\{\sum_{j=0}^{k}\left[(f_{56})_j(\mathrm{s}\sigma_G)_{k-j} - (f_{57})_j(\mathrm{c}\sigma_G)_{k-j}\right] - \sum_{j=1}^{k}(\mathrm{c}\beta_A)_j(\mathrm{s}\sigma_A)_{k-j}\right\} \tag{129}$$

$$(\mathrm{c}\sigma_A)_k = \frac{1}{(\mathrm{c}\beta_A)_0}\left\{\sum_{j=0}^{k}\left[(f_{47})_j(\mathrm{c}\sigma_G)_{k-j} - (f_{54})_j(\mathrm{s}\sigma_G)_{k-j}\right] - \sum_{j=1}^{k}(\mathrm{c}\beta_A)_j(\mathrm{s}\sigma_A)_{k-j}\right\} \tag{130}$$

$$(\alpha_A)_k = \frac{1}{(\mathrm{c}\alpha_A)_0}\left[(\mathrm{s}\alpha_A)_k - \frac{1}{k}\sum_{j=1}^{k} j(\alpha_A)_j(\mathrm{c}\alpha_A)_{k-j}\right] \tag{131}$$

$$(f_{59})_k = \sum_{j=0}^{k}(\mathrm{c}\gamma_G)_j(f_{40})_{k-j} \tag{132}$$

$$(f_{60})_k = -(f_{49})_k + \sum_{j=0}^{k}(f_{52})_j(f_{47})_{k-j} \tag{133}$$

$$(f_{61})_k = \sum_{j=0}^{k}(f_L)_j(\mathrm{s}\sigma_A)_{k-j} \tag{134}$$

$$(f_{62})_k = \sum_{j=0}^{k}(f_L)_j(\mathrm{c}\sigma_A)_{k-j} \tag{135}$$

$$(f_{D,TG})_k = \sum_{j=0}^{k}(f_D)_j(f_{55})_{k-j} + (f_{61})_j(f_{53})_{k-j} + (f_{62})_j(f_{52})_{k-j} \tag{136}$$

$$(f_{S,TG})_k = \sum_{j=0}^{k}(f_D)_j(f_{59})_{k-j} + (f_{61})_j(f_{47})_{k-j} + (f_{62})_j(f_{54})_{k-j} \tag{137}$$

$$(f_{L,TG})_k = \sum_{j=0}^{k}(f_D)_j(f_{60})_{k-j} - (f_{61})_j(f_{57})_{k-j} + (f_{62})_j(f_{56})_{k-j} \tag{138}$$

$$(f_{34})_k = \frac{-1}{(V_G)_0}\left[(f_{L,TG})_k + \sum_{j=0}^{k}\left[(g_{down})_j(\mathrm{c}\gamma_G)_{k-j} + (g_{north})_j(f_{30})_{k-j} - (f_{29})_j(f_{32})_{k-j}\right] + \sum_{j=1}^{k}(V_G)_j(f_{34})_{k-j}\right] \tag{139}$$

$$(f_{35})_k = \frac{1}{(V_G)_0}\left[(f_{S,TG})_k + \sum_{j=0}^{k}(f_{33})_j(\mathrm{s}\chi_G)_{k-j} - \sum_{j=1}^{k}(V_G)_j(f_{35})_{k-j}\right] \tag{140}$$

$$(\dot{V}_G)_k = (f_{D,TG})_k + \sum_{j=0}^{k}\left[-(g_{down})_j(\mathrm{s}\gamma_G)_{k-j} + (g_{north})_j(f_{28})_{k-j} + (f_{29})_j(f_{31})_{k-j}\right] \tag{141}$$

References

[1] Barrio, R., "Performance of the Taylor Series Method for ODEs/DAEs," *Applied Mathematics and Computation*, Vol. 163, 2005, pp. 525–545.

[2] Moore, R.E., *Interval Analysis*, Prentice-Hall, Englewod Cliffs, N.J., 1966.

[3] Hull, T.E., Enright, W.H., Fellen, B.M. and Sedgwick, A.E., "Comparing Numerical Methods for Ordinary Differential Equations," *SIAM Journal on Numerical Analysis*, Vol. 9:4, 1972, pp. 603–637.

[4] Irvine, D.H. and Savageau, M.A., "Efficient Solution of Nonlinear Ordinary Differential Equations Expressed in S-System Canonical Form," *SIAM Journal on Numerical Analysis*, Vol. 27:3, 1990, pp. 704–735.

[5] Jorba, A. and Zou, M., "A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods," *Experimental Mathematics*, Vol. 14:1, 2005, pp. 99–117.

[6] Corliss, G.F. and Chang, Y.F., "Ratio-Like and Recurrence Relation Tests for Convergence of Series," *IMA Journal of Applied Mathematics*, Vol. 25:4, 1980, pp. 349–359.

[7] Corliss, G.F. and Chang, Y.F., "Solving Ordinary Differential Equations Using Taylor Series," *ACM Transactions on Mathematical Software*, Vol. 8:2, 1982, pp. 114–144.

[8] Corliss, G.F. and Chang, Y.F., "G-Stop Facility in ATOMFT, a Taylor Series Ordinary Differential Equation Solver," *Computational Ordinary Differential Equations*, edited by Fatunla, S.O., University Press, Champaign, Ill., 1992, pp. 37–77.

[9] Corliss, G.F. and Chang, Y.F., "ATOMFT: Solving ODEs and DAEs Using Taylor Series," *Computers & Mathematics with Applications*, Vol. 28:10, 1994, pp. 209–233.

[10] Corliss, G.F. et al., "High-order Stiff ODE Solvers via Automatic Differentiation and Rational Prediction," *Lecture Notes in Computational Science*, Vol. 1196, Springer, Berlin, 1997, pp. 114–125.

[11] Pryce, J.D., "Solving High-index DAEs by Taylor Series," *Numerical Algorithms*, Vol. 19, 1998, pp. 195–211.

[12] Scott, J.R. and Martini, M.C., "High Speed Solution of Spacecraft Trajectory Problems Using Taylor Series Integration," *Journal of Spacecraft and Rockets*, Vol. 47:1, 2010, pp. 199–202.

[13] Fehlberg, E., "Classical Fifth-, Sixth-, Seventh-, and Eight-Order Runge-Kutta Formulas with Stepsize Control," Tech. Rep. NASA-TR-R-287, NASA, Washington, D.C., 1968.

[14] Bergsma, M.C.W. and Mooij, E., "Application of Taylor Series Integration to Reentry Problems," *AIAA SciTech 2016*, No. AIAA-2016–0024, AIAA, 2016.

[15] Mooij, E., "The HORUS-2B Reference Vehicle," Delft University of Technology, Memorandum M-682, 1995.

[16] Mooij, E., *Aerospace-Plane Flight Dynamics: Analysis of Guidance and Control Concepts*, Ph.D. thesis, Delft University of Technology, 1998.

[17] NIMA, "World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems," Tech. rep., NIMA, 2000.

[18] Mooij, E., *The Motion of a Vehicle in a Planetary Atmosphere*, Delft University Press, Delft, 1997.

[19] Tapley, B. et al., "GGM02 - An improved Earth gravity field model from GRACE," *Journal of Geodesy*, Vol. 79:8, 2005, pp. 467–478.

[20] NOAA, NASA, and USAF, *U.S. Standard Atmosphere, 1976*, Washington, D.C., 1976.

[21] Moler, C., *Numerical Computing with MATLAB*, SIAM, Philadelphia, 2004.

[22] Press, W.H. et al., *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1992.