We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

Sinusoidal Trajectory Generation Methods for Spacecraft Feedforward Control

Kyle A. Baker

IntechOpen

Abstract

The following is a brief walkthrough of material related to the modeling of spacecraft dynamics with feedforward control as the self-awareness declaration for deterministic artificial intelligence. Specifically, the focus will be on the analysis of various sinusoidal trajectory methods. The methods utilized are the basic MATLAB sine generation function, a Taylor series implementation, and two alternate algorithms for higher speed, lower precision and lower speed, higher precision implementations. The chapter features a brief summary of previous work investigating the impact of step size on Euler and Body angles. This is followed by a high level overview of Euler angle theory, quaternions, direction cosine matrices, kinematics, and dynamics to form a mathematical basis for the core material. With the numerical basis for the modeling efforts outlined, the results of running a SIMULINK model of spacecraft dynamics with feedforward control will be briefly analyzed and explored. The analysis will cover the impacts of varying step size with various sinusoidal trajectory generation methodologies.

Keywords: sine wave approximation, sinusoidal trajectory generation, feedforward control, Taylor Series approximation, spacecraft torque generation, space vehicle rotational mechanics, SIMULINK

IntechOpen

1. Introduction

The study of spacecraft rotational mechanics includes three core functional areas: kinematics, dynamics, and disturbances. This chapter will be primarily focused on the trajectory generation methodology used to drive the space vehicle dynamics. In the model shown in **Figure 1**, a commanded spacecraft body movement is taken in and translated into both an input torque (T) and an angular velocity (ω) for the spacecraft body. A quaternion and direction cosine matrix are then calculated and employed to find Euler Angles for the spacecraft's attitude with respect to an inertial frame.

Previously unpublished academic work, based on the topology found in **Figure 1**, delved into the effects of varying time steps and maneuver time in feedforward control (everything to the left of dynamics) but considered only a single method of sinusoid generation. Eq. (1) is the idealized feedforward control for the simulation and it can additionally function as the self-awareness declaration for deterministic artificial intelligence [1–9]. In this chapter, the Torque generator

IntechOpen

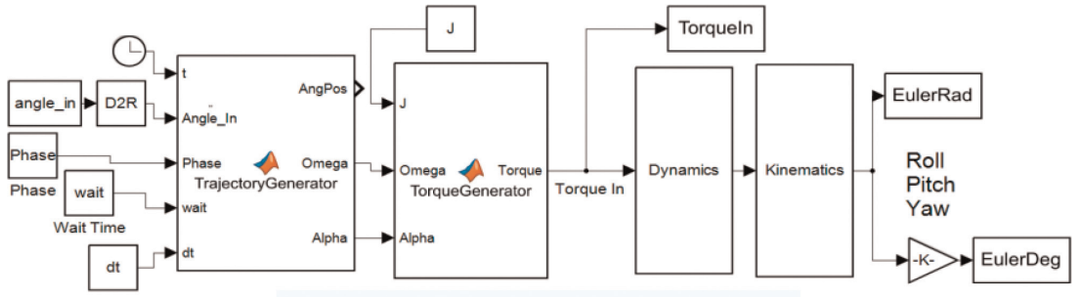


Figure 1.
Simulink model topology.

driven by Eq. (1) will remain unchanged but we will investigate the impact of four different sinusoid generation methods in the Trajectory generator, namely:

- MATLAB's sine function
- 4th order Taylor series approximation
- Low precision approximation algorithm
- High precision approximation algorithm

These methodologies require examination due to the inherent errors in an actual spacecraft's measurement apparatus. No onboard system can ever measure a spacecraft's angular position, velocity, or acceleration at infinite precision. Additionally, any disturbances accounted for in the coarse control from feedforward design implementations will still require additional feedback control for robust, fine tuning. As such, it may be more prudent to embrace a small amount of course control error in order to speed up overall computation time. In order to assess these methodologies, a SIMULINK model for each Trajectory generator method was created and their outputs were compared for error and computation time.

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} J_{xx}\dot{\omega}_x + J_{xy}\dot{\omega}_y + J_{xz}\dot{\omega}_z - J_{xy}\omega_x\omega_z - J_{yy}\omega_y\omega_z - J_{yz}\omega_z^2 + J_{xz}\omega_x\omega_y + J_{zz}\omega_z\omega_y + J_{yz}\omega_y^2 \\ J_{yx}\dot{\omega}_x + J_{yy}\dot{\omega}_y + J_{yz}\dot{\omega}_z - J_{yz}\omega_x\omega_y - J_{zz}\omega_x\omega_z - J_{xz}\omega_x^2 + J_{xx}\omega_x\omega_z + J_{xy}\omega_z\omega_y + J_{xz}\omega_z^2 \\ J_{zx}\dot{\omega}_x + J_{zy}\dot{\omega}_y + J_{zz}\dot{\omega}_z - J_{xx}\omega_x\omega_y - J_{xz}\omega_y\omega_z - J_{xy}\omega_y^2 + J_{yy}\omega_x\omega_y + J_{yz}\omega_z\omega_x + J_{xy}\omega_x^2 \end{bmatrix} \tag{1}$$

2. Sinusoidal trajectory generation

In the previous work referred to in Section 1 of this chapter it was found that a sufficiently small time step size must be utilized in order to adequately model the commanded input as a sinusoid. This can be seen with a snapshot of the experimental results as shown in **Figure 2**.

These graphs were generated when the model shown in **Figure 1** was tested with a commanded input of $[0, 0, 30]^T$ which corresponds to zero roll, zero pitch, and a 30° yaw. This allowed analysis to focus on time step impacts to one axis of rotation of MATLAB sinusoidal generation methodology. The quiescent and post-maneuver timeframes were both set to 5 s and the maneuver was conducted over a 10 s interval. **Figure 2** shows the results of a time step of 1 s and a time step of 0.01 s,

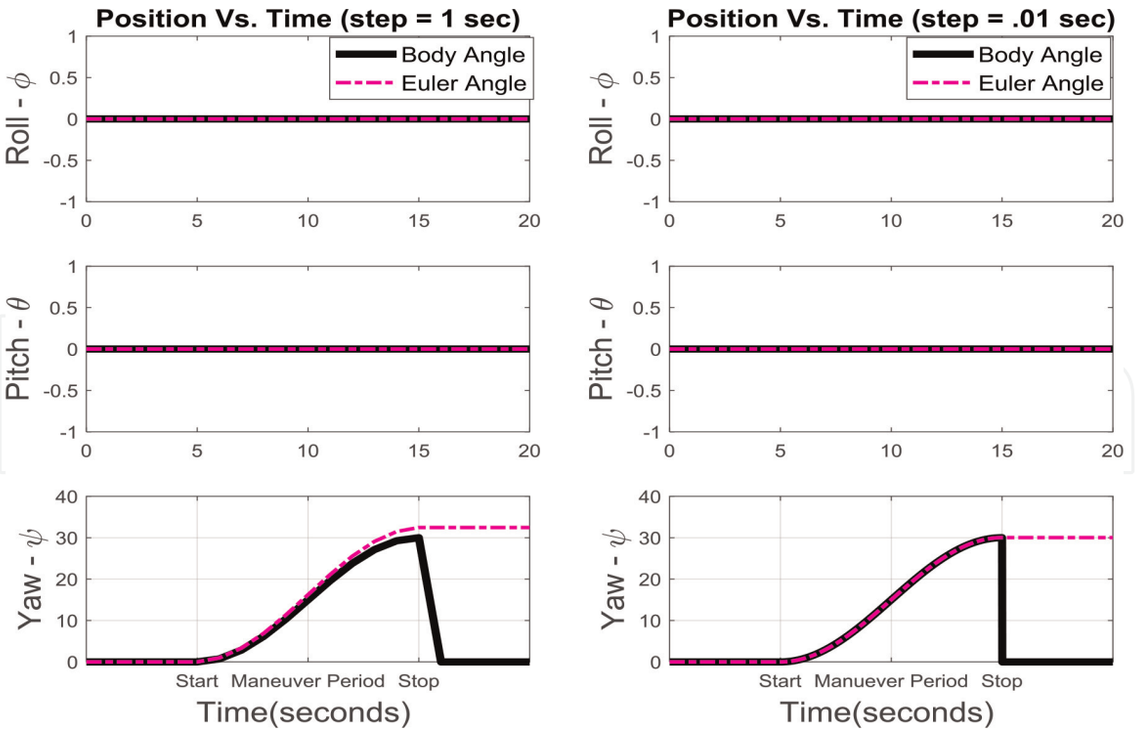


Figure 2.
Position vs. time graphs as a function of time step.

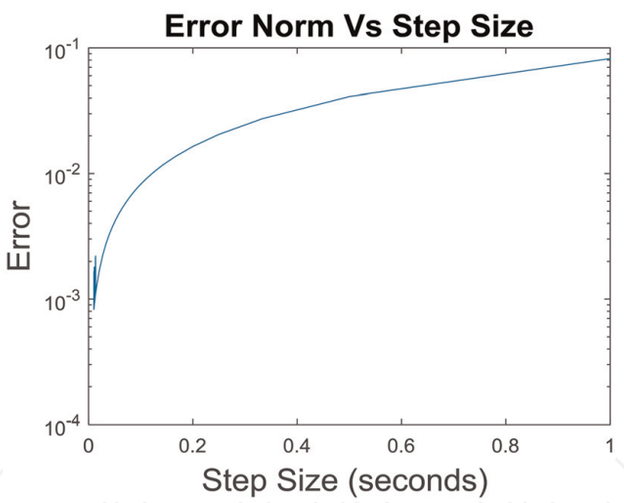


Figure 3.
Error vs. step size.

both using the Runge-Kutta solver. With the larger step size of 1 s, the Euler angles lost tracking and went slightly above the commanded angle while the smaller step size of 0.01 s afforded better tracking between Euler angle and Body angle. The main takeaway was that with finer step size resolution one could essentially equate the Euler and Body angles on a given model even though the former is in the inertial reference frame and the latter is in the spacecraft body reference frame.

Additionally, 100 iterations were run over a range of step sizes between 1 and 0.01 s to see what impact a chosen step size would have on error between Body and Euler Angles at the end of the maneuver phase. The results of this study can be seen below in **Figure 3**.

From this graph you can clearly see the benefit of reducing step size. It should be noted, however, that once step size was reduced to 0.01119 s the model reached a

point where CPU and MATLAB precision comes into play and shrinking step size did not necessarily result in reducing error. This is noticeable with the erratic nature of the curve on the left side of the graph when approaching a much small step size.

Since it was found that we could adequately model our system with a small enough step size this allowed further investigation into other areas within the model which could be changed and optimized. This line of thought led us to create a new model which could be utilized to compare and contrast various sinusoidal generation methodologies. This new model's numerical basis and simulation results will be shown in the following two subsections.

2.1 Model creation

Before adding changes for various sinusoidal generation methodologies the mathematical basis and assumptions used in the previous model were re-verified. The first step used in model creation (and re-verification) was the implementation of a direction cosine matrix to numerically represent rotations about a set of axes to project a starting frame onto a desired reference frame in order to outline the system kinematics [10–14]. **Figures 4** and **5** provide visual depictions of the process driven by Eq. (2) through Eq. (4). Each direction cosine matrix equation takes an axial rotation as depicted by the series of rotations in **Figure 4** and represents it as an orthonormal matrix consisting of sines, cosines, zeroes, and ones per the trigonometry rules shown in **Figure 5**.

$$1\,RotationDCM = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\Phi & \sin\Phi \\ 0 & -\sin\Phi & \cos\Phi \end{bmatrix} \tag{2}$$

$$2\,RotationDCM = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3}$$

$$3\,RotationDCM = \begin{bmatrix} \cos\Psi & \sin\Psi & 0 \\ -\sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

For more complicated movements, direction cosine matrices are multiplied together resulting in a more intricate matrix. Eq. (5) is an example of this for a 3-2-1 Rotation. This direction cosine matrix sequence and others are provided a more in depth treatment in Refs. [14-16] and as well as other chapters in this book. Note that in Eq. (5), the "C" and "S" characters are utilized as shorthand for the sine and cosine trigonometric functions.

$$321\,DCM\,Rotation = \begin{bmatrix} C\theta C\Psi & C\Phi S\Psi & -S\theta \\ S\Phi S\theta C\Psi - C\Phi S\Psi & S\Phi S\theta S\Psi + C\Phi C\Psi & S\Phi C\theta \\ C\Phi S\theta C\Psi + S\Phi S\Psi & C\Phi S\theta S\Psi - S\Phi \mathrm{C}0 & C\Phi C\theta \end{bmatrix} \tag{5}$$

The other major workhorse of the model's kinematics is the orthonormal quaternion matrix [17–19], which accomplishes the same feat as the direction cosine matrix but with intrinsic divide by zero protection. The particular quaternion calculation format implemented in the model can be seen in Eq. (6).

$$\dot{q} = \begin{bmatrix} 0 & \omega 3 & -\omega 2 & \omega 1 \\ -\omega 3 & 0 & \omega 1 & \omega 2 \\ \omega 2 & -\omega 1 & 0 & \omega 3 \\ -\omega 1 & -\omega 2 & -\omega 3 & 0 \end{bmatrix} \begin{bmatrix} q1 \\ q2 \\ q3 \\ q4 \end{bmatrix} \tag{6}$$

To find the quaternion from Eq. (6), it is a simple matter of initializing the quaternion vectors with an orthonormal set (i.e., $[0,0,0,1]^T$) and integrating the output.

The final matrix utilized, shown in Eq. (7), is an equation equivalent to the previously shown 3-2-1 rotational matrix Eq. (5) but written in terms of quaternion elements. This quaternion based 3-2-1 rotation is depicted below in Eq. (7).

$$321\ DCMRotation = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_3q_4) & 2(q_1q_3-q_2q_4) \\ 2(q_1q_2-q_3q_4) & 1-2(q_1^2+q_3^2) & 2(q_2q_3+q_1q_4) \\ 2(q_1q_3+q_2q_4) & 2(q_3q_2-q_1q_4) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{7}$$

In actual implementation, the SIMULINK model uses Eq. (7) but for calculation purposes takes advantage of Eq. (5). Combining terms leaves us with Eq. (8) through Eq. (10) for Euler Angles:

$$\theta = \sin^{-1}(1-2(q_2^2+q_3^2)) \tag{8}$$

$$\Phi = \text{atan2}(2(q_2q_3+q_1q_4)/1-2(q_1^2+q_2^2)) \tag{9}$$

$$\Psi = \text{atan2}(2(q_1q_2+q_3q_4)/1-2(q_2^2+q_3^2)) \tag{10}$$

Referring back to **Figure 1**, the Dynamics block is driven by Eq. (11) (the Euler moment equation) when a torque is provided by the Feedforward Control section's Torque generator.

$$\sum T = \dot{H} = J\dot{\omega} + \omega \times J\omega \tag{11}$$

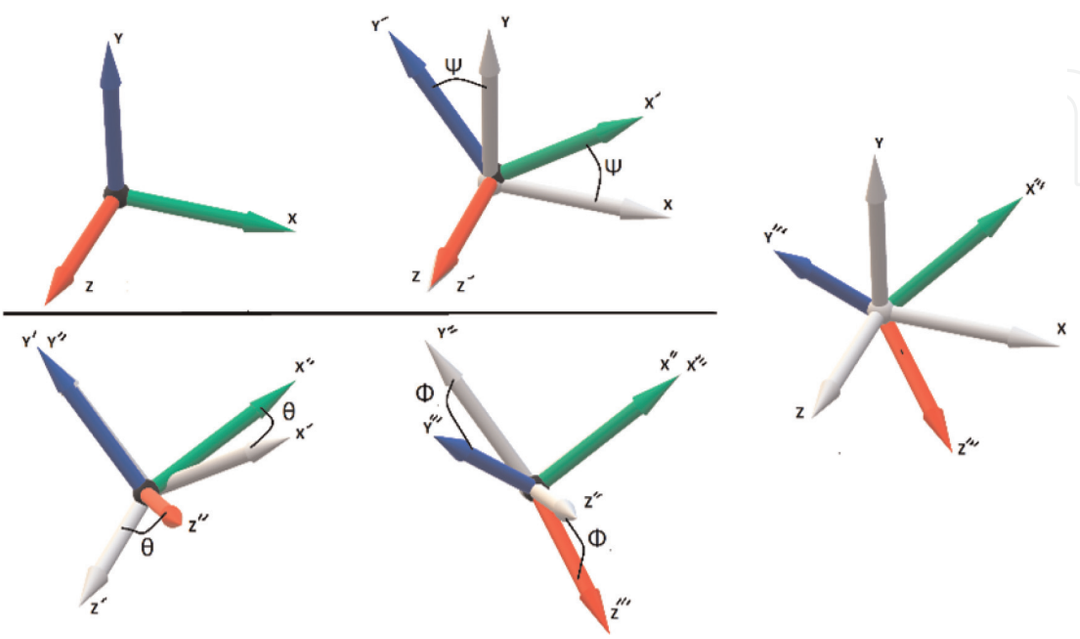


Figure 4.
Simple 3-2-1 rotation.

To derive the angular velocity of the body (ω_{BODY}) fed from dynamics to kinematics, we multiply moment of inertia (J) by the time derivative of angular velocity, i.e., angular acceleration ($\dot{\omega}$). This $J\dot{\omega}$ term is multiplied by the inverse of J and then integrated as per Eq. (12).

$$\int \left(J^{-1} * J\dot{\omega}\right)dt = \omega_{BODY} \tag{12}$$

Afterwards, the ω_{BODY} is used to obtain the spacecraft attitude's Euler angles.

The feedforward control blocks in **Figure 1** contain the Trajectory generator and the Torque generator. The Trajectory generator takes in a commanded body angle and then uses a sine wave in order to approximate the commanded maneuver. To help elaborate on this point, **Figure 6** shows a square wave and a sine wave, both shifted up in amplitude such that they operate from zero to two instead of between negative and positive one.

We can see that if a square wave was used to approximate the commanded maneuver, the spacecraft would essentially be expected to transition from the zero position up to its max amplitude at the two positions instantaneously. This is a physical impossibility. With the sine wave in the same figure we can see that the

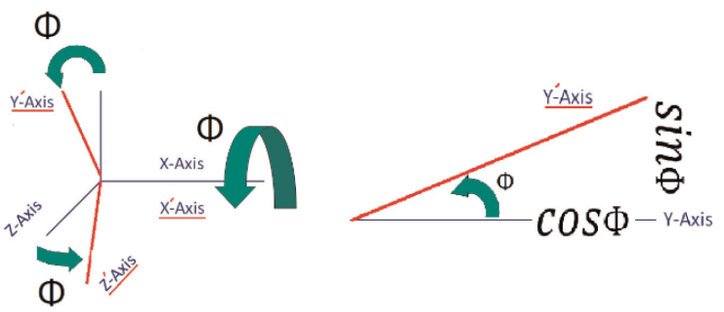


Figure 5.
Depiction of a Φ rotation about the X axis.

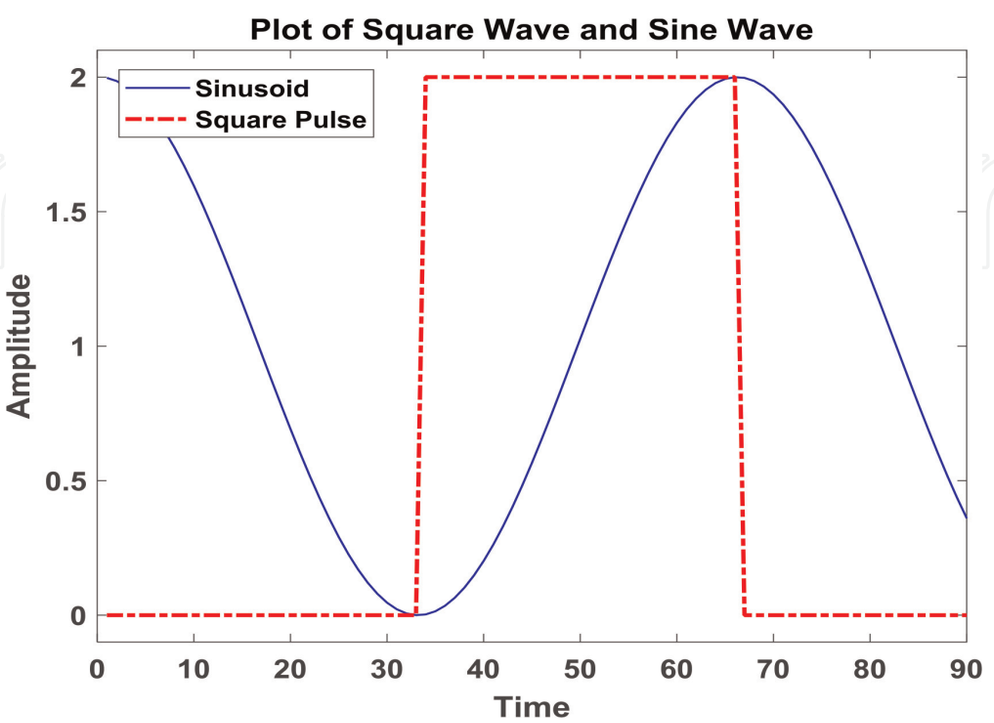


Figure 6.
Visual depiction of square and sine waves.

spacecraft machinery is afforded a ramp up and slow down period along with a relatively linearly slope in the middle. Basically, the half period of sinusoid within the square pulse of **Figure 6** is used to provide a smooth, achievable input to the system.

Our original model's Trajectory generator follows Eq. (13) through Eq. (15) to approximate the commanded maneuver for angular position, angular velocity (ω), and angular acceleration ($\dot{\omega}$) via the MATLAB sine wave function.

$$AngularPosition = \frac{1}{2}\left(A + A * \sin\left(\left(\frac{\pi}{\Delta t}\right)(t - t_{wait}) - \frac{\pi}{2}\right)\right) \tag{13}$$

$$AngularVelocity = \frac{1}{2}A * \left(\frac{\pi}{\Delta t}\right)\cos\left(\left(\frac{\pi}{\Delta t}\right)(t - t_{wait}) - \frac{\pi}{2}\right) \tag{14}$$

$$AngularAcceleration = -\frac{1}{2}A * \left(\frac{\pi}{\Delta t}\right)^2 \sin\left(\left(\frac{\pi}{\Delta t}\right)(t - t_{wait}) - \frac{\pi}{2}\right) \tag{15}$$

Eq. (13) models the input command, where "A" is the maneuver's commanded angle. The base frequency of the sinusoid (ω_{sine}) is $\left(\frac{\pi}{\Delta t}\right)$ where (Δt) is the desired maneuver time. The t_{wait} term allows for a quiescent period and $-\frac{\pi}{2}$ term allows for a proper phase shift to implement the sinusoidal half period of **Figure 4**. The effect of this can be seen later on in **Figure 7**. Eqs. (14) and (15) are just successive derivatives of Eq. (13) used to generate angular velocity and acceleration which are fed into Eq. (1) in the Torque generator from **Figure 1**. This produces an output torque which drives the dynamics.

From Eq. (13) through Eq. (15) it can be clearly seen that the argument of the sine and cosine terms always follows the form of Eq. (16). We can use this to implement our Taylor series and the other two other algorithms on equal footing.

$$Arg = \left(\frac{\pi}{\Delta t}\right)(t - t_{wait}) - \frac{\pi}{2} \tag{16}$$

The Taylor series, as detailed in Ref. [20], is a numerical method that can be used to approximate other functions. In our model we substitute the sine and cosine in Eq. (13) through Eq. (15) with Eqs. (17) and (18).

$$Taylor\ Sin = Arg - \frac{Arg^3}{3!} - \frac{Arg^5}{5!} - \frac{Arg^7}{7!} \tag{17}$$

$$Taylor\ Cos = 1 - \frac{Arg^2}{2!} - \frac{Arg^4}{4!} - \frac{Arg^6}{6!} \tag{18}$$

The Taylor series is a power series and additional terms could be included ad infinitum for greater precision; however, initial testing found that four series terms provided a reasonable approximation while maintaining a viable runtime. Additionally, it was found that pre-calculating the factorial terms sped up computation time.

The last set of sinusoidal generation methodologies tested was the low precision (LP) and high precision (HP) algorithms. These were found to be used in many applications, especially ones which limited by lower processing power such as mobile gaming. Refs. [21, 22] provided the baseline for adaptation of the baseline Eqs. (19) and (20). Note that the same equation is used for sine and cosine except that the argument term is given a positive $\frac{\pi}{2}$ phase shift when applied to the cosine in Eq. (14). In Eq. (19), $(+/-)$ indicates that a plus is used if Arg is less than zero and a minus is used otherwise.

$$LP = 1.27323954 * Arg\ (+/-)\ 0.405284735 * Arg^2 \tag{19}$$

$$HP = .225 * [LP * abs(LP) - LP] + LP \tag{20}$$

Eq. (20) provides additional smoothing to Eq. (16) at the cost of computation time to implement a high precision mode. This equation could be implemented such that "$LP * abs(LP)$" essentially become the magnitude of LP^2. While this analysis used "if then" statements, there may be more efficient ways to implement Eqs. (19) and (20) depending on the software package being utilized.

2.2 Model simulation and analysis

For the purpose of model verification, commands are initially held constant at zero resulting in zero torque to allow us to evaluate model operation during a quiescent period. With zero input torque, Eq. (11) shows us that the change in angular momentum should be zero as well; therefore the model should not change. After a 5 s quiescent period, a 30° yaw maneuver was input into each model and the maneuver was conducted over a 10 s period at which point the commanded body angle went to zero (indicating no more change required). Afterwards the output Euler angles should remain constant with the spacecraft's new attitude. The results of this test were plotted in **Figure 7**. Additionally, note that the model was setup with an arbitrary diagonalized inertia matrix (J) per Eq. (21). The roll and pitch Euler angles remained at zero, as they should have, and as such are not shown.

$$J = \begin{bmatrix} 150 & 0 & 0 \\ 0 & 90 & 0 \\ 0 & 0 & 35 \end{bmatrix} \tag{21}$$

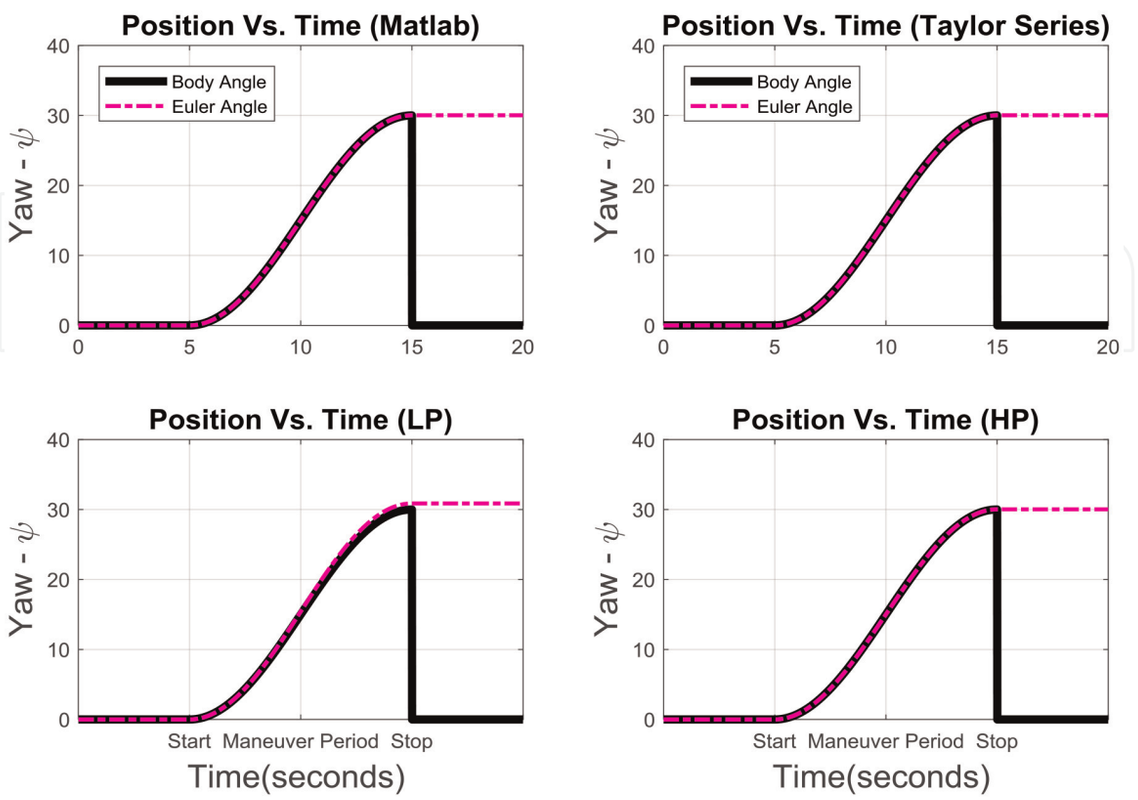


Figure 7.
Feedforward control for 30° yaw maneuver.

From a qualitative standpoint you can see that each sine generation algorithm created the intended Euler angle movement shown earlier in **Figure 4**. Upon closer examination the only algorithm that appeared to lose tracking on the body angle was the low precision algorithm. Since we achieved the basic shape from each method, we now turn our attention to analyzing the error and computation time associated with each for a given time step interval.

In the previous work addressed in the beginning of Section 2 of this we found that for a Runge-Kutta solver with step size of 0.01119, the MATLAB sine function reached a point where CPU and MATLAB precision started to affect the gains offered by shrinking step size. With this in mind, step sizes of 0.1, 0.05, and 0.1 s were chosen for analysis, results of which can be seen in **Figure 8** and **Table 1**.

One major inference to be drawn from **Table 1** is that the normalized error columns show Taylor, LP, and HP algorithms are roughly step-size invariant within our test range; only the MATLAB function's error reduces with step size in our model. Interestingly, the LP and HP functions can be more accurate than Taylor but we are more concerned with the final angle in coarse control (corresponding to the peak of sine curve) rather than the intermediate steps—so in this case, the Taylor series is considered more accurate.

The other major item of note is the difference in average computation time. While the MATLAB command is ultimately the most accurate at all step sizes, we can see that the LP and HP algorithms generally run faster. It should be noted that results were relatively inconsistent when run below 200 iterations but at 500

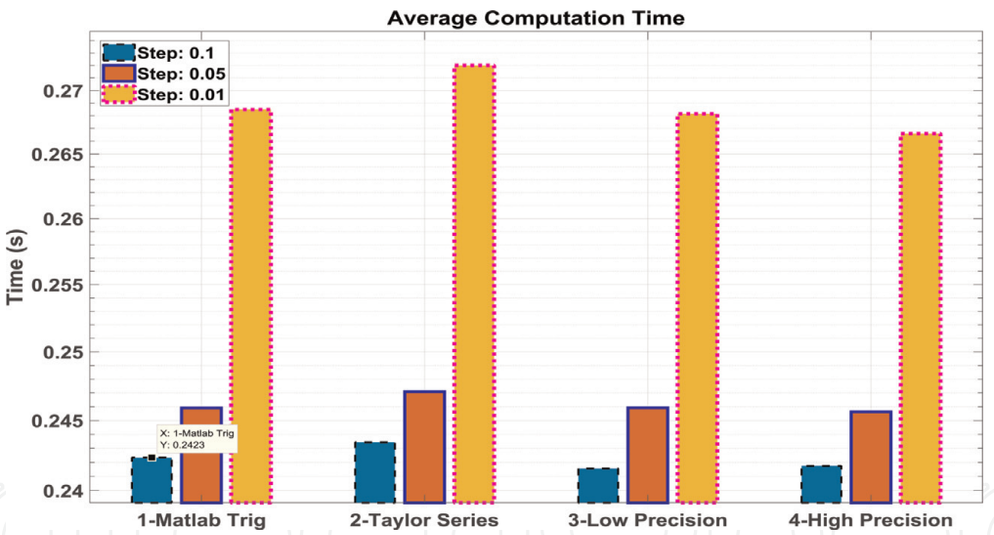


Figure 8.
Computation time results.

Method	Step size		Step size		Step size	
	0.01 s		0.5 s		0.1 s	
	Time(s)	Error	Time(s)	Error	Time(s)	Error
1-MATLAB	0.2423	1.02E-9	0.2459	6.36E-11	0.2685	1.02E-13
2-Taylor	0.2434	3.53E-5	0.2471	3.53E-5	0.2720	3.53E-5
3-LP	0.2416	2.81E-2	0.2459	2.81E-2	0.2682	2.81E-2
4-HP	0.2417	3.25E-4	0.2456	3.25E-4	0.2666	3.25E-4

Time and error results averaged over 500 iterations per model.

Table 1.
Timing and error results.

iterations they stabilized with HP and LP usually running faster during a given test. Since HP would run faster than LP sometimes and vice versa, even at greater iterations; it is believed that this may come down to specific CPU architecture used and how a dual core processor handles calculations.

3. Conclusions

When it comes to sinusoidal trajectory generation, if your spacecraft design has accurate fine control (e.g., robust feedback mechanisms) and can handle coarse control error on the order of 10^{-4}, then it may be more prudent to utilize the HP algorithm. This would allow you to reduce processing power and time resulting in lower power requirements and faster onboard calculations. A fourth order Taylor series would not be beneficial due to longer computation times and less accuracy than the MATLAB function; extending the order of the series would increase accuracy but also extend its runtime, thereby, losing viability. Additionally, simulation results reveal that the fastest methodology may vary with CPU architecture indicating that there may not be a definitive answer for "best" trajectory generation method. This should be evaluated within the specific spacecraft design trade space. Very little information is available on the proprietary MATLAB sine calculations but future testing will investigate the LP and HP algorithms against MATLAB on a variety of CPU architectures and organic sinusoidal generation on other platforms such as PYTHON.

Author details

Kyle A. Baker
Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA, USA

*Address all correspondence to: kabaker@nps.edu

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. Distributed under the terms of the Creative Commons Attribution - NonCommercial 4.0 License (https://creativecommons.org/licenses/by-nc/4.0/), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited. BY–NC

References

[1] Lobo K, Lang J, Starks A, Sands T. Analysis of deterministic artificial intelligence for inertia modifications and orbital disturbances. International Journal of Control Science and Engineering. 2018;**8**(3):53-62

[2] Sands T. Electric vehicle sales catastrophe averted by deterministic artificial intelligence methods. Applied Sciences. 2018;**8**:2081

[3] Nakatani S. Simulation of spacecraft damage tolerance and adaptive controls. In: 2014 IEEE Aerospace Conference. 2014. pp. 1-16. DOI: 10.1109/AERO.2014.6836260

[4] Nakatani S. Autonomous damage recovery in space. International Journal of Automation, Control and Intelligent Systems. 2016;**2**(2):22-36. ISSN Print: 238175

[5] Nakatani S. Battle-damage tolerant automatic controls. Electronics and Electrical Engineering. 2018;**8**:10-23. DOI: 10.5923/j.eee.20180801.02.

[6] Heidlauf P, Cooper M. Nonlinear Lyapunov control improved by an extended least squares adaptive feed forward controller and enhanced Luenberger observer. In: Proceedings of the International Conference and Exhibition on Mechanical & Aerospace Engineering; 2-4 October 2017; Las Vegas, NV, USA. 2017

[7] Cooper M, Heidlauf P, Sands T. Controlling chaos—Forced van der pol equation. Mathematics on Automation Control Systems, A Special Issue of Mathematics. 2017;**5**:70-80. DOI: 10.3390/math5040070

[8] Sands T. Phase lag elimination at all frequencies for full state estimation of spacecraft attitude. Physics Journal. 2017;**3**(1):1-12

[9] Sands T. Nonlinear-adaptive mathematical system identification. Computation. 2017;**5**:47-59. DOI: 10.3390/computation5040047

[10] Wright T. Elements of Mechanics Including Kinematics, Kinetics, and Statics, With Applications. New York: Nostrand; 1909

[11] Eduard Study (D.H. Delphenich Translator). Foundations and goals of analytical kinematics. Berlin Mathematical Society. Sitzber. d. Berl. Math. Ges. 1913;**13**:36-60. Available from: http://neo-classical-physics.info/uploads/3/4/3/6/34363841/study-analytical_kinematics.pdf [Accessed: 14 April 2017]

[12] Gray A. A Treatise on Gyrostatics and Rotational Motion. London: MacMillan; 1918. ISBN 978-1-4212-5592-7

[13] Fung AC, Zimmermun BG. Digital simulation of rotational kinematics. In: NASA Technical Report NASA TN D-5302; October 1969; Washington, DC. 1969. Available from: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19690029793.pdf

[14] Smeresky B, Rizzo A, Sands T. Kinematics in the information age. Mathematical Engineering, A Special Issue of Mathematics. 2018;**6**(9):148. DOI: 10.3390/math6090148

[15] Wie B. Space Vehicle Dynamics and Control. 1st ed. Reston, VA, USA: AIAA, McGraw-Hill; 1998. pp. 310-327. DOI: 10.2514/4.103803

[16] Slabaugh GG. Computing Euler Angles From a Rotation Matrix. 1999;6 (2000).pp. 39-63. Available form: h ttp://www.close-range.com/Docspacec raftomputing_Euler_angles_from_a_rotation_matrix.pdf [Accessed: January 1999]

[17] Henderson DM. Euler Angles, Quaternions, and Transformation Matrices—Working Relationships. McDonnell Douglas Technical Services Co. Inc., as NASA Technical Report NASA-TM-74839. 1977. Available from: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770024290.pdf

[18] Henderson DM. Euler Angles, Quaternions, and Transformation Matrices for Space Shuttle Analysis. McDonnell Douglas Technical Services Co. Inc., Houston Astronautics Division as NASA Design Note 1.4-8-020. 1977. Available from: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19770019231.pdf

[19] Kuipers J. Quaternions and Rotation Sequences, Geometry, Integrability, and Quantization; 1-10 September 1999; Varna, Bulgaria. Sofia: Coral Press; 2000

[20] Weisstein EW. Taylor Series. From MathWorld—A Wolfram Web Resource [Internet]. 2018. Available from: http://www.mathworld.wolfram.com/TaylorSeries.html [Accessed: 21 June 2018]

[21] Baczynski M. Fast and Accurate Sine/Cosine Approximation [Internet]. 2007. Available from: http://www.lab.polygonal.de/2007/07/18/fast-and-accurate-sinecosine-approximation/ [Accessed: 21 June 2018]

[22] Climatiano M. Faster Sine Approximation Using Quadratic Curve [Internet]. 2014. Available from: http://www.mclimatiano.com/faster-sine-approximation-using-quadratic-curve/ [Accessed: 21 June 2018]