AE4878 - Mission Geometry and Orbit Design
Part 8 - Full sky geometry Dual axis spiral for ground coverage

Ali Nawaz
3ME, LR, Delft University of Technology.

September 2, 2018

1 Introduction

The objective of this report is to outline the working principles two fundamental yet insightful orbit design tools. First full sky geometries are analysed with the aid of simple spherical triangle trigonometry. The resourcefulness of such a simple tool is demonstrated with the aid of an airplane problem. The tool constructed can provide insight into all parameters of a spherical triangle if three other parameters are known. These parameters can later be used to provide information such as the area projected on a sphere. This bears special importance in the coverage analysis of a remote sensing constellation.

Later part of this report, focuses on designing a tool for a dual axis spiral. The dual axis problem has several applications in providing insight into the mission geometry e.g. ground coverage of a rotating sensor on a spinning spacecraft, satellite ground track and so on.

First the reason as to why we need full sky spherical geometry and the dual axis spiral problems are defined.

Why full sky spherical geometry?
The answer to this is simple. It presents a quick way of providing insight into the unknown geometric elements of satellites/points on a sphere. The problem is purely geometry. It looks into three points one a sphere. These three points can give rise to 8 possible spherical triangles as shown in Figure 1[4][Fig8-2]. The area of the projected spherical triangle can easily be estimated as shown in Figure 2[4][Fig8-3], this can provide quick but valuable insight into coverage analysis of a satellite.

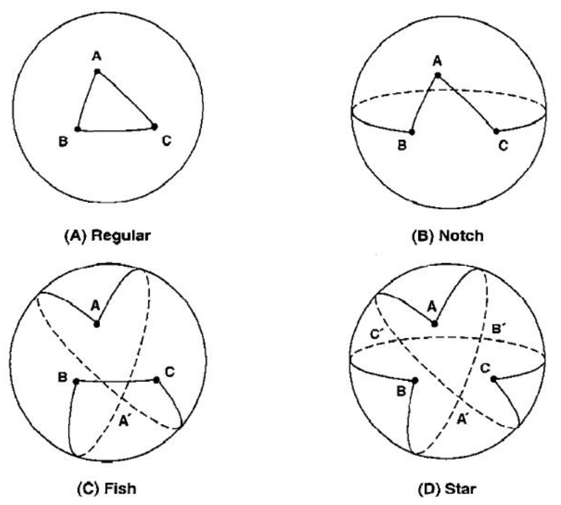


Fig. 8-2. Any 3 Vertices Define 8 Possible Shapes for Spherical Triangles. Each pair of vertices can be connected by going either the short way (< 180 deg) or the long way (> 180 deg). Either 0, 1, 2, or 3 sides can be longer than 180 deg and the "inside" and "outside" are interchangeable. This results in 8 possible types of triangles and in 16 unique shapes because for the notch and fish any of the 3 sides can be the one which is a short or long arc.

Figure 1: Three points on a sphere can define 8 possible shapes for a spherical triangle.

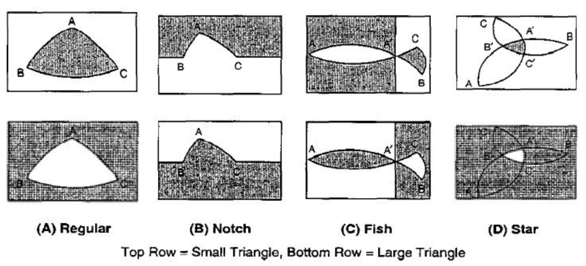


Fig. 8-3. An Alternative "Flat" Sketch of the 8 Possible Spherical Triangles. In the notch any of the 3 sides can be greater than 180 deg and in the fish any of the sides can be less than 180 deg. This results in the 8 possible sets of sides and 16 possible shapes because either area defined by a particular shape can be taken as the inside. In traditional spherical geometry only the shaded triangle in the top diagram of (A) is allowed.

Figure 2: Area projected by different spheres.

Why dual-axis spiral?

Before describing why "Dual Axis Spiral" (DAS) is used, it is important to understand what it actually is. As the name represents a dual axis problem is comprised of two rotating axis. This is explained with the aid of Figure 3[4][Fig. 8-8].

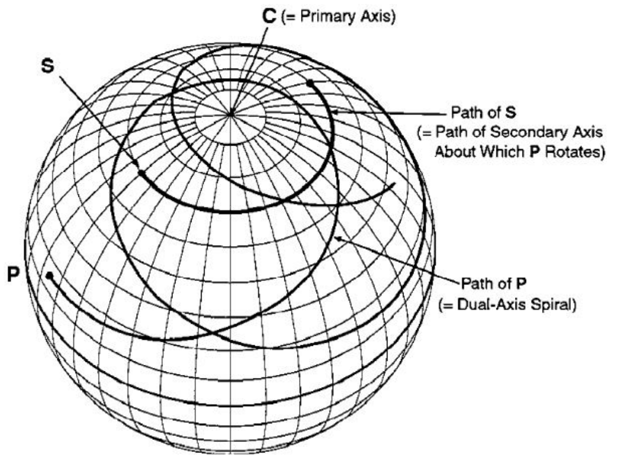

C (= Primary Axis)
S
Path of S (= Path of Secondary Axis About Which P Rotates)
P
Path of P (= Dual-Axis Spiral)

Fig. 8-8. The Dual-Axis Spiral. The spiral is created by a point, **P**, rotating about a secondary axis, **S**, which is itself rotating about a primary or central axis, **C**.

Figure 3: Dual axis spiral coordinate overview.

The primary axis (which is the rotation axis of the sphere in this case) is first axis. A point S (which is the secondary axis) is rotating around the primary axis C. A sub-satellite point P, is in turn rotating around point S. The scenario is best described with the aid of a hypothetical spinning spacecraft with an imaging sensor on board. Let's assume that the sphere represents the Earth, spinning on it's axis C. The spinning spacecraft is point S. The spacecraft is going around point C. The imaging sensor on board of the spacecraft is pointing at point P on the sphere. But since the spacecraft is also spinning about point S, point P leaves behind a trail on the sphere which is very different than than trail left behind by the spacecraft S. This is clearly pointed out with the aid of Figure 3. If a remote sensing company relies solely on the ground track left behind by the spacecraft, it is going to provide it's customers with image data from a wrong location. The track left behind by the line of sight of the spacecraft, denoted by P is of true value.

This is where the dual axis spiral comes in, if one knows the angular velocity of S about C (ω_1), angular velocity of P about S (ω_2), angular distance of S from C (ρ_1) and angular distance of P from S (ρ_2), the dual axis problem can provide valuable insight into the Azimuth/Elevation (α/δ) of P about C, velocity of P and the direction of motion of P (Ψ). This report will focus on the GPS groundtrack application of dual axis spiral problem.

First the full sky problem is discussed in Section 2. A general tool for the "Dual Axis Spiral" problem is defined and verified in Section 3. The general DAS tool is extended to a handle a GPS groundtrack problem in Section 4. The results of this GPS problem is outlined in Section 5. Multiple scripts defining the algorithms behind the tools is outlined in Section 6.

2 Full sky geometry problem

The objective of this section is to outline 2 different combinations of angles and sides provided for a spherical triangle. The spherical triangle system is outlined with the aid of Figure 4 [2]. The first subsection deals with an Angle-Side-Angle problem, where a side and two adjacent angle are provided. The second subsection deals with an Angle-Angle-Side problem, where two angles and a side are provided. The results are verified with the aid of an airplane problem outlined in [3][Slide20/48].

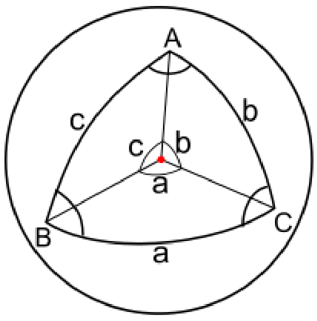


Figure 4: Nomenclature/ Overview of a spherical triangle.

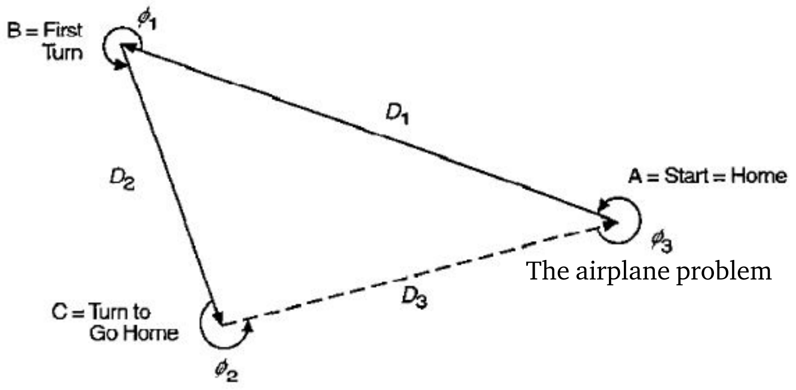


Fig. 8-1. The Airplane Problem. This exemplifies a problem that is conceptually simple yet computationally complex for both vectors and traditional spherical geometry because any of the sides or angles can range from 0 to 360 deg.

Figure 5: Airplane problem coordinate overview.

is outlined with the aid of Figure 5[4]. Solution to the airplane problem is tabulated in the results section along with the results of Angle-Side-Angle and Angle-Angle-Side problem.

2.1 Angle-Side-Angle

The first problem under consideration is the Angle-Side-Angle problem. This is outlined with the aid of Figure 6[4][Fig.A-8]. The problem has two solutions of angle C and sides a,b for given angles A,B and side c.

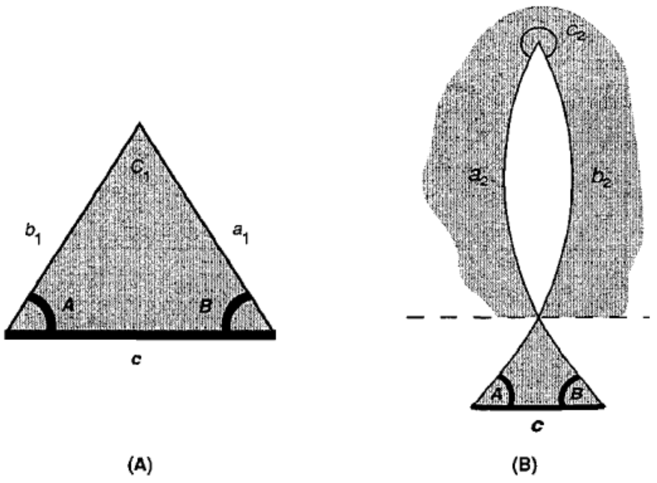


Fig. A-8. The Two Solutions for the Angle-Side-Angle Triangle.

Figure 6: Spherical triangle and projected area overview for an Angle-Side-Angle problem.

The first and second solutions are obtained using the guidelines outlined in Figure 7[4][pg.788]. It is important to note that the term H defines the Hemisphere function. While function *acos2()* is not built in Matlab. Both of the functions are constructed in line with the descriptions outlined in Figure 8[4][pg.792].

First Solution:

If a non-singular solution exists, then there are two solutions. One is given by:

$$C_1 = \text{acos2}\,[-\cos A \cos B + \sin A \sin B \cos c, \text{H}(c)] \qquad \text{(A-35a)}$$

$$a_1 = \text{acos2}\left[\frac{\cos A + \cos B \cos C_1}{\sin B \sin C_1}, \text{H}(A)\right] \qquad \text{(A-35b)}$$

$$b_1 = \text{acos2}\left[\frac{\cos B + \cos A \cos C_1}{\sin A \sin C_1}, \text{H}(B)\right] \qquad \text{(A-35c)}$$

where the acos2 function is defined in Sec. A.7.7 and discussed further in Sec. 8.1.

Second Solution:

The second solution is given by:

$$C_2 = 360° - C_1 \qquad \text{(A-36a)}$$

$$a_2 = (a_1 + 180°) \bmod 360° \qquad \text{(A-36b)}$$

$$b_2 = (b_1 + 180°) \bmod 360° \qquad \text{(A-36c)}$$

Figure 7: Expressions used to evaluate the two solutions to Angle-Side-Angle problem.

$$\text{acos2}[\cos\phi, \text{H}(\phi)] \equiv [\text{H}(\phi)\ \text{acos}(\cos\phi)]\text{mod}_{360\ \text{deg}} \qquad \text{(A-42)}$$

where

$$0\ \text{deg} \le \text{acos}(\phi) < 180\ \text{deg} \qquad \text{(A-43)}$$

and

$$\begin{aligned} \text{H}(\phi) &\equiv +1 \text{ if } 0\ \text{deg} \le \phi\ \text{mod}_{360\ \text{deg}} < 180\ \text{deg} \\ \text{H}(\phi) &\equiv -1 \text{ if } 180\ \text{deg} \le \phi\ \text{mod}_{360\ \text{deg}} < 360\ \text{deg} \end{aligned} \qquad \text{(A-44)}$$

Figure 8: Expressions used to evaluate the hemisphere() and acos2() function.

A verification table and results table in presented in Section 2.3. The script for all the above algorithms are presented in Section 6.

2.2 Angle-Angle-Side

The second problem under consideration is the Angle-Angle-Side problem. This is outlined with the aid of Figure 9[4][pg.786]. The problem has two solutions of angle C and sides b,c for given angles A,B and side a.

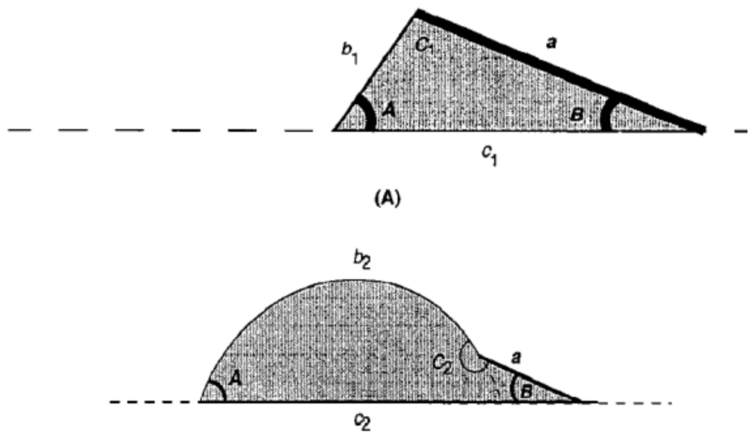
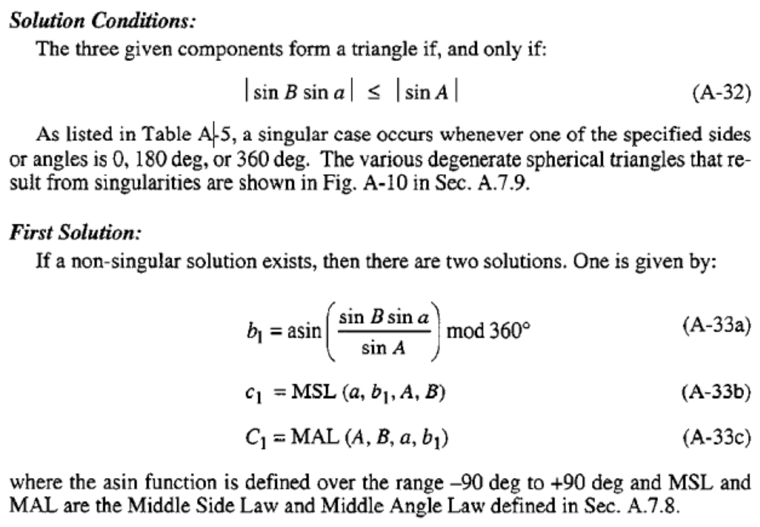


Fig. A-7. The Two Solutions for the Angle-Angle-Side Triangle.

Figure 9: Spherical triangle and projected area overview for an Angle-Angle-Side problem.

The solution conditions and first solution is outlined with the aid of Figure 10[4][pg.786]. The second solution is outlined with the aid of Figure 11[4][pg.787].



Solution Conditions:

The three given components form a triangle if, and only if:

$$|\sin B \sin a| \leq |\sin A| \quad \text{(A-32)}$$

As listed in Table A-5, a singular case occurs whenever one of the specified sides or angles is 0, 180 deg, or 360 deg. The various degenerate spherical triangles that result from singularities are shown in Fig. A-10 in Sec. A.7.9.

First Solution:

If a non-singular solution exists, then there are two solutions. One is given by:

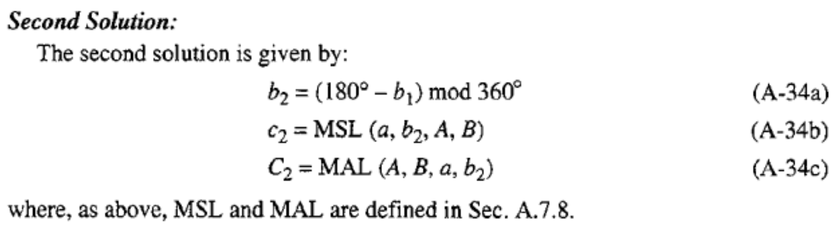
$$b_1 = \text{asin}\left(\frac{\sin B \sin a}{\sin A}\right) \bmod 360° \quad \text{(A-33a)}$$

$$c_1 = \text{MSL}(a, b_1, A, B) \quad \text{(A-33b)}$$

$$C_1 = \text{MAL}(A, B, a, b_1) \quad \text{(A-33c)}$$

where the asin function is defined over the range –90 deg to +90 deg and MSL and MAL are the Middle Side Law and Middle Angle Law defined in Sec. A.7.8.

Figure 10: Solution conditions and expressions used to evaluate the first round of solutions to Angle-Angle-Side problem.



Second Solution:

The second solution is given by:

$$b_2 = (180° - b_1) \bmod 360° \quad \text{(A-34a)}$$

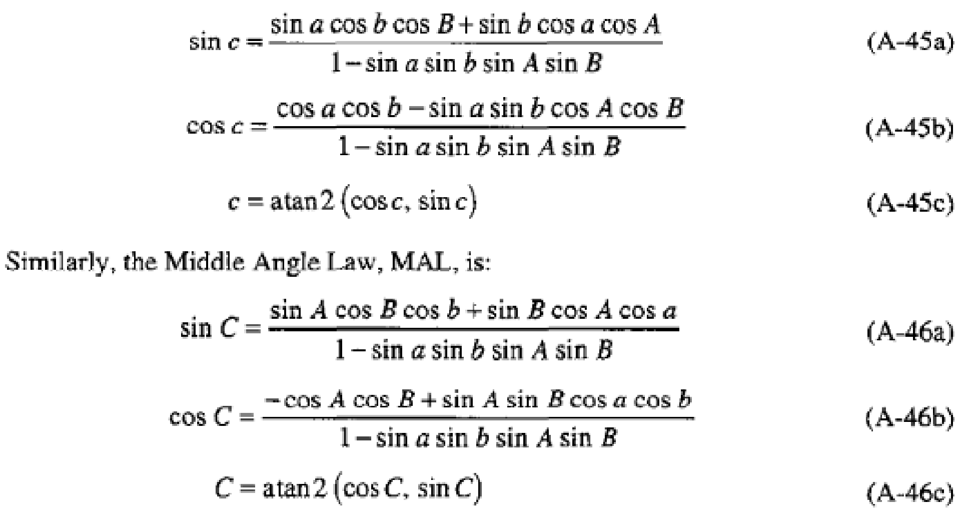
$$c_2 = \text{MSL}(a, b_2, A, B) \quad \text{(A-34b)}$$

$$C_2 = \text{MAL}(A, B, a, b_2) \quad \text{(A-34c)}$$

where, as above, MSL and MAL are defined in Sec. A.7.8.

Figure 11: Expressions used to evaluate the second round of solutions to Angle-Side-Angle problem.

The conditions outlined as MSL (Mean Side Law) and MAL (Middle Angle Law) are estimated with as described in Figure 12[4][pg.792].



$$\sin c = \frac{\sin a \cos b \cos B + \sin b \cos a \cos A}{1 - \sin a \sin b \sin A \sin B} \quad \text{(A-45a)}$$

$$\cos c = \frac{\cos a \cos b - \sin a \sin b \cos A \cos B}{1 - \sin a \sin b \sin A \sin B} \quad \text{(A-45b)}$$

$$c = \text{atan2}\left(\cos c, \sin c\right) \quad \text{(A-45c)}$$

Similarly, the Middle Angle Law, MAL, is:

$$\sin C = \frac{\sin A \cos B \cos b + \sin B \cos A \cos a}{1 - \sin a \sin b \sin A \sin B} \quad \text{(A-46a)}$$

$$\cos C = \frac{-\cos A \cos B + \sin A \sin B \cos a \cos b}{1 - \sin a \sin b \sin A \sin B} \quad \text{(A-46b)}$$

$$C = \text{atan2}\left(\cos C, \sin C\right) \quad \text{(A-46c)}$$

Figure 12: Expressions outlining Middle Side Law and the Middle Angle Law.

Matlab's built in *atan2()* resulted in -ve angles, thus a function in line with the definition of *atan2()* outlined in Figure 13[4][pg.791]. This function is labelled as *myatan2d()*.

$$
\begin{aligned}
\operatorname{atan2}(x,y) &\equiv \operatorname{atan}(y/x) && \text{if } 0 \le y \le |x| \text{ and } x > 0\\
&\equiv 360^\circ + \operatorname{atan}(y/x) && \text{if } -|x| \le y \le 0 \text{ and } x > 0\\
&\equiv 180^\circ + \operatorname{atan}(y/x) && \text{if } |y| \le |x| \text{ and } x < 0 \qquad \text{(A-41)}\\
&\equiv 90^\circ - \operatorname{atan}(x/y) && \text{if } y \ge |x|\\
&\equiv 270^\circ - \operatorname{atan}(x/y) && \text{if } y < -|x|
\end{aligned}
$$

Figure 13: Expressions outlining the definition of function *myatan2d()*.

A script implementing the above algorithms along with the functions *mytan2d()*, *MSL()* and *MAL()* is outlined in Section 6.

2.3 Results

Table 1 presents the verification results of scripts outlined earlier against the given airplane problem. The scripts are verified to be correct.

Table 1: Verification of the Angle-Side-Angle and Angle-Angle-Side methods.

Solution [3][Slide 20/48]		**Angle-Side-Angle**		**Angle-Angle-Side**	
Parameter	**Value [deg]**	**Input Value [deg]**	**Output Value [deg]**	**Input Value [deg]**	**Output Value [deg]**
ϕ_2 = A	300.64, 120.64	300.64		300.64	
ϕ_3 = B	323.95,143.95	323.95		323.95	
D_3 = c	35.53, 324.47	35.53			35.53, 194.52
ϕ_1 = C	270		90,270		270, 154.45
D_1 = a	30		210, 30	30	
D_2 = b	20		200, 20		20, 160

Now that the scripts are verified, the given Angle-Side-Angle and Angle-Angle-Side problem can be solved. The results of this is presented with the aid of Table 2.

Table 2: Results of Angle-Side-Angle and Angle-Angle-Side method for the given problems.

	Angle-Side-Angle		**Angle-Angle-Side**	
Parameter	**Input Value [deg]**	**Output Value [deg]**	**Input Value [deg]**	**Output Value [deg]**
A	45		45	
B	60		20	
c	22			28.2124, 193.3671
C		77.6297, 282.3703		116.8303, 205.8740
a		15.7347, 195.7347	22	
b		19.3981, 199.3981		10.4393, 169.5607

This concludes the section on full sky geometry. The tool is verified to be correct. The method illustrated is an efficient approach to find key parameters with just a few unknown, though simple trigonometry yet quite powerful.

3 General dual axis spiral

First a general algorithm is written to solve the dual axis problem. The coordinate system of a ”Dual Axis Spiral” was outlined in Introduction. However, a new axis is defined as the Euler axis/ instantaneous rotation axis E. This axis will be instantaneously at rest. Throughout the motion of point P, E is a point along the great circle connecting S and C. This is where the rotations ω_1 and ω_2 cancel each other out.Figure 14[4][pg.398] outlines the new coordinate system with Euler axis, E. Inclusion of E is insightful for ground track analysis, it’s impact on higher orbit is quite significant. The satellite rotates about E at fixed angular rate ω_E.

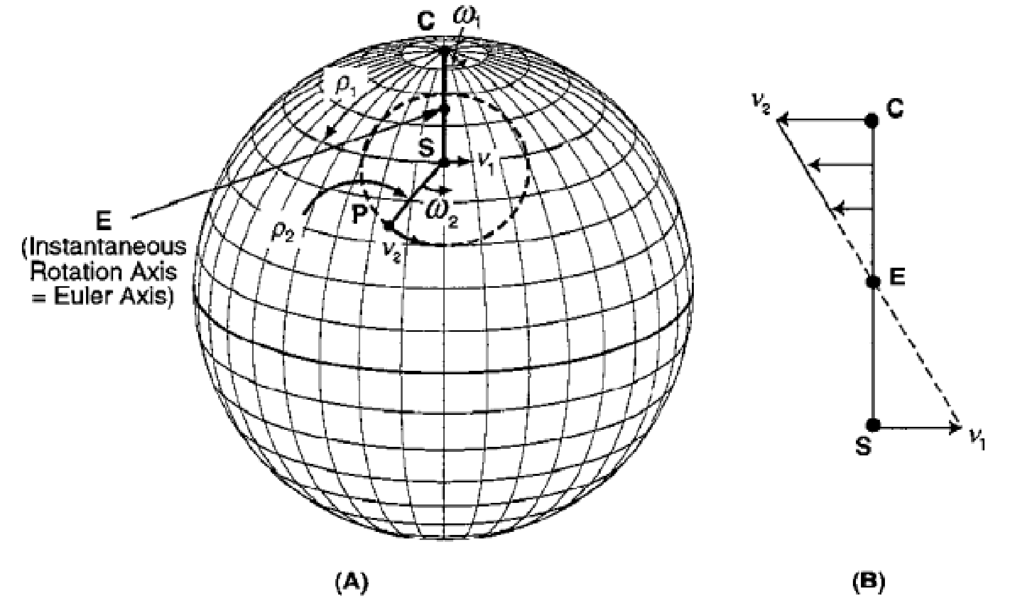
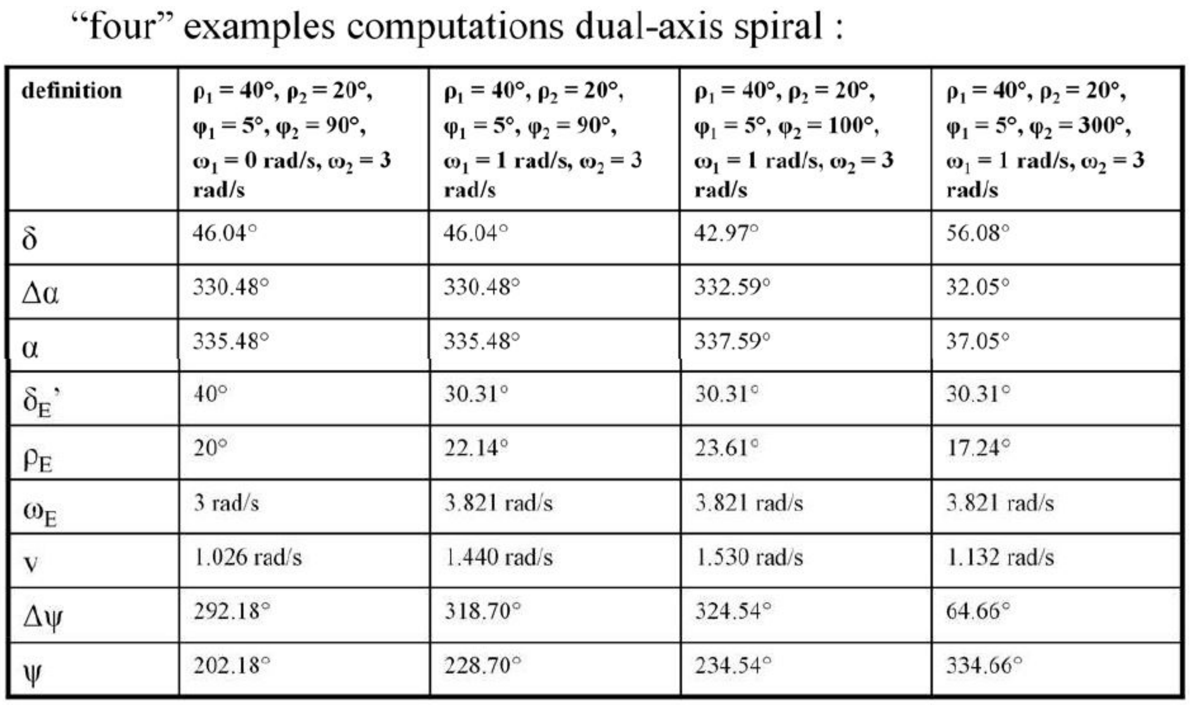


Fig. 8-11. Geometry of the Dual-Axis Spiral. (A) The point **P** rotates with angular velocity ω_2 about the axis **S** which, in turn, is rotating with angular velocity ω_1 about axis **C**. In the combined rotation of the two coordinate frames, there is one point, the instantaneous rotation axis or Euler axis, **E**, where the two rotations cancel and which is instantaneously at rest as shown by figure (B).

Figure 14: Dual axis spiral updated coordinate overview.

The process of defining the general dual axis spiral is outlined with the aid of Table presented in Figure16[4][pg.403]. A general script is presented in Section . Table 3 presents the results obtained with this method and Figure 15 presents the benchmark results[3][Slide38/48].



"four" examples computations dual-axis spiral :

definition	$\rho_1 = 40°$, $\rho_2 = 20°$, $\varphi_1 = 5°$, $\varphi_2 = 90°$, $\omega_1 = 0$ rad/s, $\omega_2 = 3$ rad/s	$\rho_1 = 40°$, $\rho_2 = 20°$, $\varphi_1 = 5°$, $\varphi_2 = 90°$, $\omega_1 = 1$ rad/s, $\omega_2 = 3$ rad/s	$\rho_1 = 40°$, $\rho_2 = 20°$, $\varphi_1 = 5°$, $\varphi_2 = 100°$, $\omega_1 = 1$ rad/s, $\omega_2 = 3$ rad/s	$\rho_1 = 40°$, $\rho_2 = 20°$, $\varphi_1 = 5°$, $\varphi_2 = 300°$, $\omega_1 = 1$ rad/s, $\omega_2 = 3$ rad/s
δ	46.04°	46.04°	42.97°	56.08°
$\Delta\alpha$	330.48°	330.48°	332.59°	32.05°
α	335.48°	335.48°	337.59°	37.05°
δ_E'	40°	30.31°	30.31°	30.31°
ρ_E	20°	22.14°	23.61°	17.24°
ω_E	3 rad/s	3.821 rad/s	3.821 rad/s	3.821 rad/s
v	1.026 rad/s	1.440 rad/s	1.530 rad/s	1.132 rad/s
$\Delta\psi$	292.18°	318.70°	324.54°	64.66°
ψ	202.18°	228.70°	234.54°	334.66°

Figure 15: Benchmark values presented in [3][Slide 38/48] for validation of script.

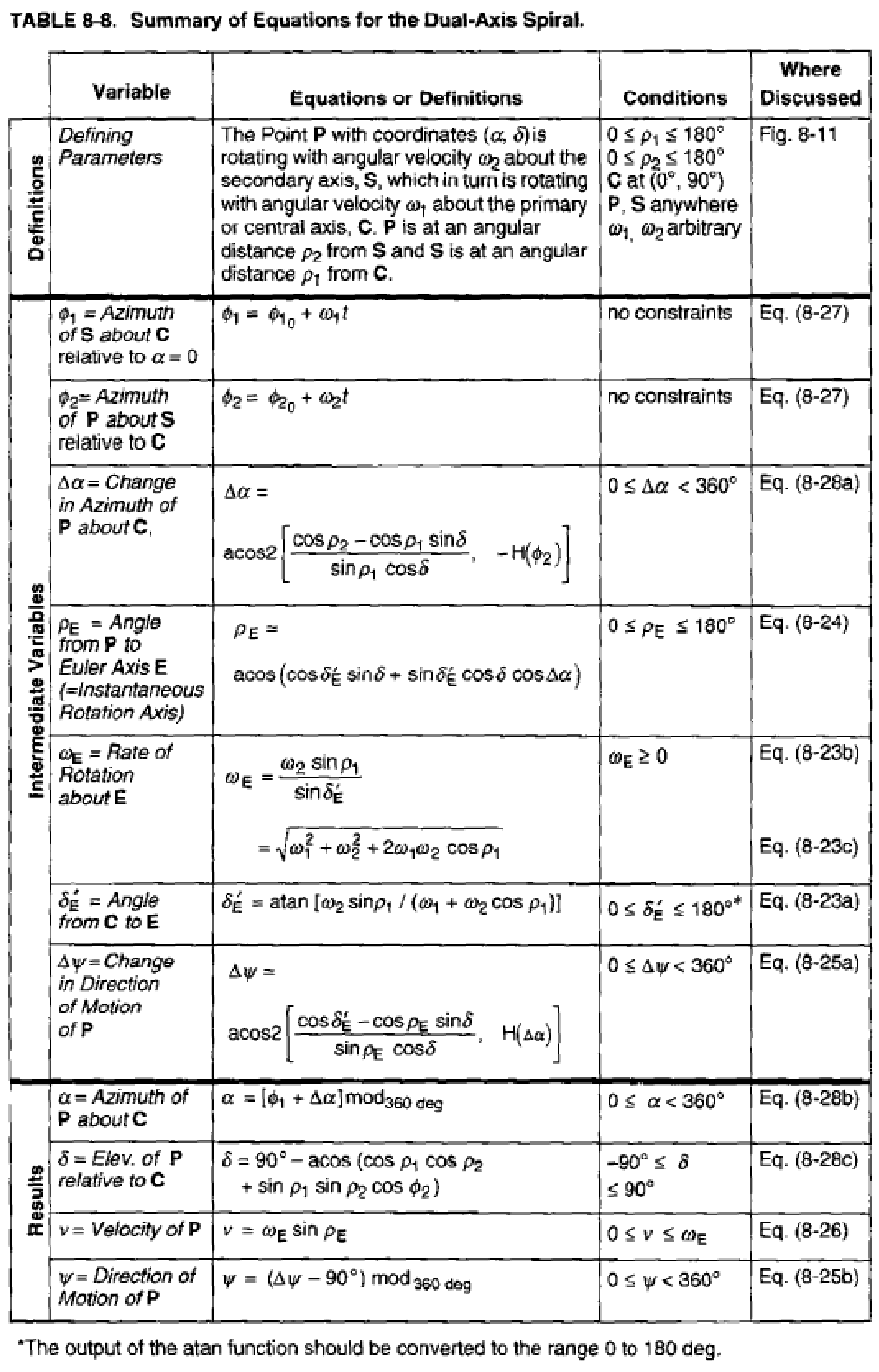


TABLE 8-8. Summary of Equations for the Dual-Axis Spiral.

	Variable	Equations or Definitions	Conditions	Where Discussed
Definitions	*Defining Parameters*	The Point **P** with coordinates (α, δ) is rotating with angular velocity ω_2 about the secondary axis, **S**, which in turn is rotating with angular velocity ω_1 about the primary or central axis, **C**. **P** is at an angular distance ρ_2 from **S** and **S** is at an angular distance ρ_1 from **C**.	$0 \le \rho_1 \le 180°$ $0 \le \rho_2 \le 180°$ **C** at $(0°, 90°)$ **P**, **S** anywhere ω_1, ω_2 arbitrary	Fig. 8-11
Intermediate Variables	ϕ_1 = *Azimuth of* **S** *about* **C** relative to $\alpha = 0$	$\phi_1 = \phi_{1_0} + \omega_1 t$	no constraints	Eq. (8-27)
	ϕ_2 = *Azimuth of* **P** *about* **S** relative to **C**	$\phi_2 = \phi_{2_0} + \omega_2 t$	no constraints	Eq. (8-27)
	$\Delta\alpha$ = *Change in Azimuth of* **P** *about* **C**,	$\Delta\alpha =$ $\mathrm{acos2}\left[\dfrac{\cos\rho_2 - \cos\rho_1 \sin\delta}{\sin\rho_1 \cos\delta},\ -H(\phi_2)\right]$	$0 \le \Delta\alpha < 360°$	Eq. (8-28a)
	ρ_E = *Angle from* **P** *to Euler Axis* **E** *(=Instantaneous Rotation Axis)*	$\rho_E =$ $\mathrm{acos}\,(\cos\delta_E' \sin\delta + \sin\delta_E' \cos\delta \cos\Delta\alpha)$	$0 \le \rho_E \le 180°$	Eq. (8-24)
	ω_E = *Rate of Rotation about* **E**	$\omega_E = \dfrac{\omega_2 \sin\rho_1}{\sin\delta_E'}$ $= \sqrt{\omega_1^2 + \omega_2^2 + 2\omega_1\omega_2 \cos\rho_1}$	$\omega_E \ge 0$	Eq. (8-23b) Eq. (8-23c)
	δ_E' = *Angle from* **C** *to* **E**	$\delta_E' = \mathrm{atan}\,[\omega_2 \sin\rho_1 / (\omega_1 + \omega_2 \cos\rho_1)]$	$0 \le \delta_E' \le 180°$*	Eq. (8-23a)
	$\Delta\psi$ = *Change in Direction of Motion of* **P**	$\Delta\psi =$ $\mathrm{acos2}\left[\dfrac{\cos\delta_E' - \cos\rho_E \sin\delta}{\sin\rho_E \cos\delta},\ H(\Delta\alpha)\right]$	$0 \le \Delta\psi < 360°$	Eq. (8-25a)
Results	α = *Azimuth of* **P** *about* **C**	$\alpha = [\phi_1 + \Delta\alpha]\mathrm{mod}_{360\ \mathrm{deg}}$	$0 \le \alpha < 360°$	Eq. (8-28b)
	δ = *Elev. of* **P** *relative to* **C**	$\delta = 90° - \mathrm{acos}\,(\cos\rho_1 \cos\rho_2 + \sin\rho_1 \sin\rho_2 \cos\phi_2)$	$-90° \le \delta \le 90°$	Eq. (8-28c)
	v = *Velocity of* **P**	$v = \omega_E \sin\rho_E$	$0 \le v \le \omega_E$	Eq. (8-26)
	ψ = *Direction of Motion of* **P**	$\psi = (\Delta\psi - 90°)\ \mathrm{mod}_{360\ \mathrm{deg}}$	$0 \le \psi < 360°$	Eq. (8-25b)

*The output of the atan function should be converted to the range 0 to 180 deg.

Figure 16: Expressions underlying the general dual axis spiral problem.

Table 3: Results generated by the script for verification.

	Set 1	**Set 2**	**Set 3**	**Set 4**
δ [deg]	46.0418	46.0418	42.9743	56.0751
$\Delta\alpha$ [deg]	330.4798	330.4798	332.5899	32.05411
α [deg]	335.4798	335.4798	337.5899	37.05411
δ'_E [deg]	40	30.31410	30.31410	30.31410
ρ_E [deg]	20.0000	22.13521	23.60828	17.24050
ω_E [rad/s]	3.0000	3.8205	3.8205	3.8205
v [rad/s]	1.026060	1.439542	1.530042	1.132334
Δ_Ψ [deg]	292.17599	318.6968	324.53521	64.66419
Ψ [deg]	202.17599	228.6968	234.53521	334.66419

Comparing the benchmark and the script results, the script is verified to be correct.

4 Extension to GPS application

Now that the general problem script is verified, the same script can be used to extend it to generate ground track of GPS satellites. The approach is outlined with the aid of Table presented in Figure 17[4][pg.411]. The primary axis is taken as the rotation axis of the Earth's pole. The secondary axis is taken as the satellite orbit pole. The primary radius ρ_1 is taken as the inclination of the orbit. For GPS satellite the orbit inclination is 55 deg[4]. The secondary radius is taken as ρ_2 = 90 deg. The α and δ parameters indicate the longitude and latitude of the satellite and α_0 and δ_0 parameters indicate the longitude and latitude of the orbit pole at any time t[4][pg.410]. The ground track angular velocity is given by n_G. ω_1 is taken as the negative of Earth's rotation rate. This can be defined as -$\frac{2\pi}{T_E}$ rad/s, here T_E = 86164.1004s is the length of sidereal day[4]. ω_2 is now the mean angular motion (n) of the satellite in its orbit.

The mean angular motion of a spacecraft in a circular orbit around Earth is outlined with the aid of Equation 1[4][pg.51]. GPS satellite altitude is taken as 20,200 km[1] and R_{Earth} = 6378.136km[4].

$$n = \sqrt{\frac{\mu_{Earth}}{a^3}} \tag{1}$$
$$\mu_{Earth} = 398600.441 km^3/s^2, \text{ a } = R_{Earth} + \text{ GPS satellite altitude}$$

For the GPS problem this report looks into one satellite revolution. The period of one complete revolution can be estimated with the aid of Equation 2[4][pg. 51].

$$P = \frac{2\pi}{n}[s] \tag{2}$$

Initial values of ϕ is taken as ϕ_{10} = 90 deg and ϕ_{20} = 0 deg. To facilitate better visualisation of one orbit revolution. Results of the analysis are outlined in the next section. A copy of the script updated to handle the GPS application is outlined in Section 6.

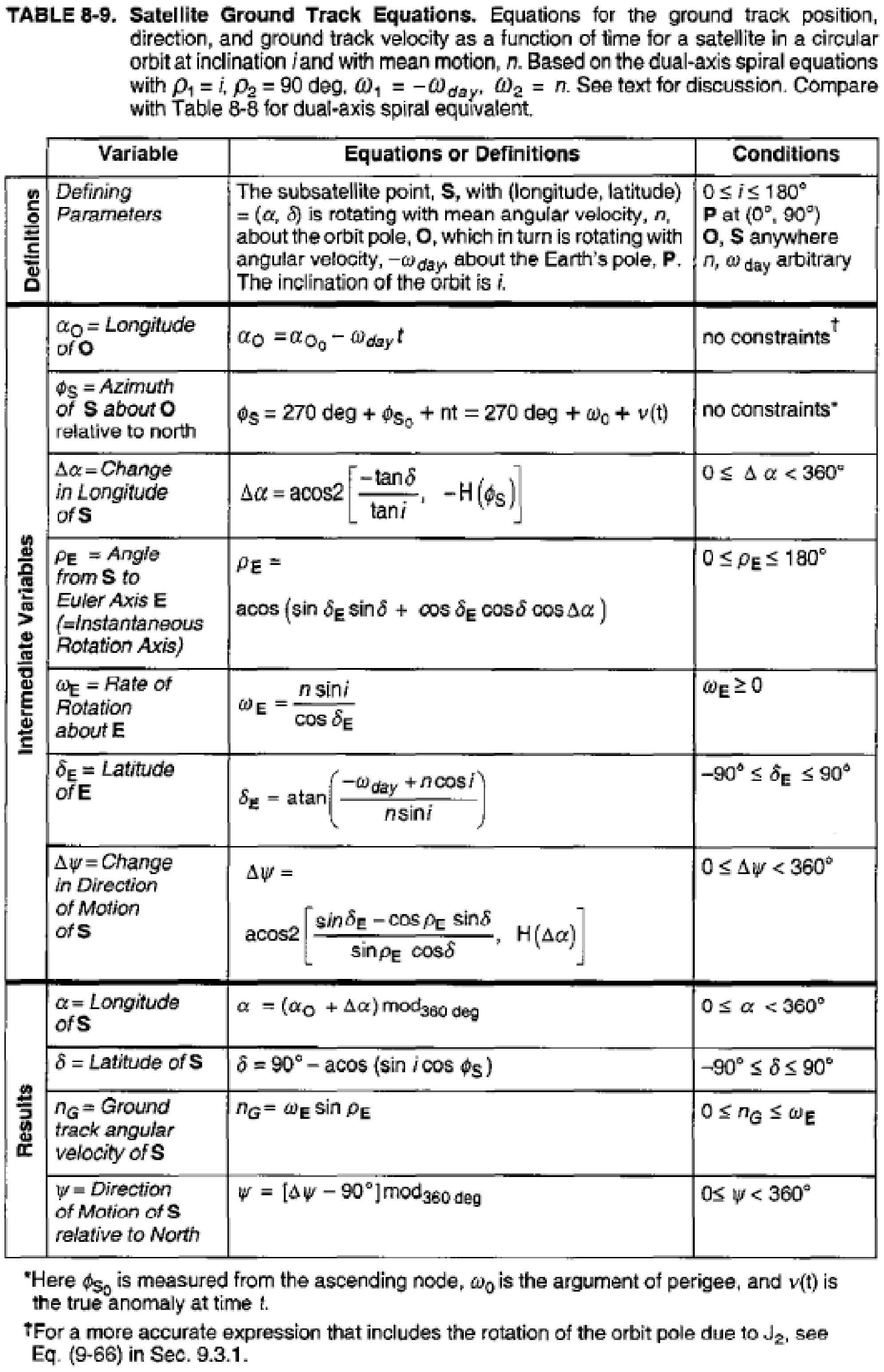


TABLE 8-9. Satellite Ground Track Equations. Equations for the ground track position, direction, and ground track velocity as a function of time for a satellite in a circular orbit at inclination i and with mean motion, n. Based on the dual-axis spiral equations with $\rho_1 = i$, $\rho_2 = 90$ deg, $\omega_1 = -\omega_{day}$, $\omega_2 = n$. See text for discussion. Compare with Table 8-8 for dual-axis spiral equivalent.

	Variable	Equations or Definitions	Conditions
Definitions	*Defining Parameters*	The subsatellite point, **S**, with (longitude, latitude) = (α, δ) is rotating with mean angular velocity, n, about the orbit pole, **O**, which in turn is rotating with angular velocity, $-\omega_{day}$, about the Earth's pole, **P**. The inclination of the orbit is i.	$0 \le i \le 180°$ **P** at (0°, 90°) **O**, **S** anywhere n, ω_{day} arbitrary
Intermediate Variables	α_O = *Longitude of* **O**	$\alpha_O = \alpha_{O_0} - \omega_{day} t$	no constraints†
	ϕ_S = *Azimuth of* **S** *about* **O** relative to north	$\phi_S = 270 \text{ deg} + \phi_{S_0} + nt = 270 \text{ deg} + \omega_0 + v(t)$	no constraints*
	$\Delta\alpha$ = *Change in Longitude of* **S**	$\Delta\alpha = \text{acos2}\left[\dfrac{-\tan\delta}{\tan i},\ -H(\phi_S)\right]$	$0 \le \Delta\alpha < 360°$
	ρ_E = *Angle from* **S** *to Euler Axis* **E** *(=Instantaneous Rotation Axis)*	$\rho_E =$ $\text{acos}\left(\sin\delta_E \sin\delta + \cos\delta_E \cos\delta \cos\Delta\alpha\right)$	$0 \le \rho_E \le 180°$
	ω_E = *Rate of Rotation about* **E**	$\omega_E = \dfrac{n \sin i}{\cos\delta_E}$	$\omega_E \ge 0$
	δ_E = *Latitude of* **E**	$\delta_E = \text{atan}\left(\dfrac{-\omega_{day} + n\cos i}{n \sin i}\right)$	$-90° \le \delta_E \le 90°$
	$\Delta\psi$ = *Change in Direction of Motion of* **S**	$\Delta\psi =$ $\text{acos2}\left[\dfrac{\sin\delta_E - \cos\rho_E \sin\delta}{\sin\rho_E \cos\delta},\ H(\Delta\alpha)\right]$	$0 \le \Delta\psi < 360°$
Results	α = *Longitude of* **S**	$\alpha = (\alpha_O + \Delta\alpha)\,\text{mod}_{360\text{ deg}}$	$0 \le \alpha < 360°$
	δ = *Latitude of* **S**	$\delta = 90° - \text{acos}(\sin i \cos\phi_S)$	$-90° \le \delta \le 90°$
	n_G = *Ground track angular velocity of* **S**	$n_G = \omega_E \sin\rho_E$	$0 \le n_G \le \omega_E$
	ψ = *Direction of Motion of* **S** *relative to North*	$\psi = [\Delta\psi - 90°]\,\text{mod}_{360\text{ deg}}$	$0 \le \psi < 360°$

*Here ϕ_{S_0} is measured from the ascending node, ω_0 is the argument of perigee, and $v(t)$ is the true anomaly at time t.

†For a more accurate expression that includes the rotation of the orbit pole due to J_2, see Eq. (9-66) in Sec. 9.3.1.

Figure 17: Expressions underlying the dual spiral axis problem, suited for ground tracking of GPS satellites.

5 Result evaluation and discussion

This section presents and discusses the results obtained for the dual axis spiral problem of the GPS satellite. The ground track of the GPS satellite for one revolution is presented with the aid of Figures 18 and 19.

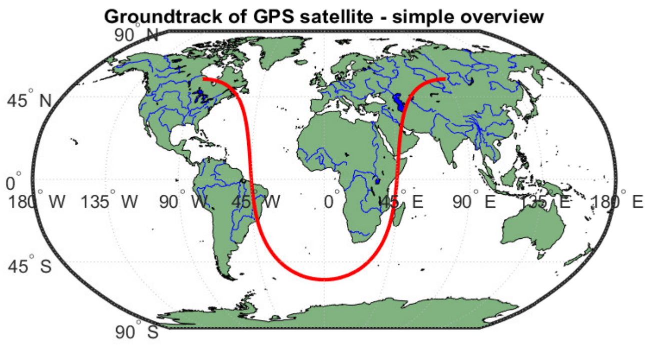


Figure 18: Simple Eckert IV projection of GPS satellite groundtrack on Earth.

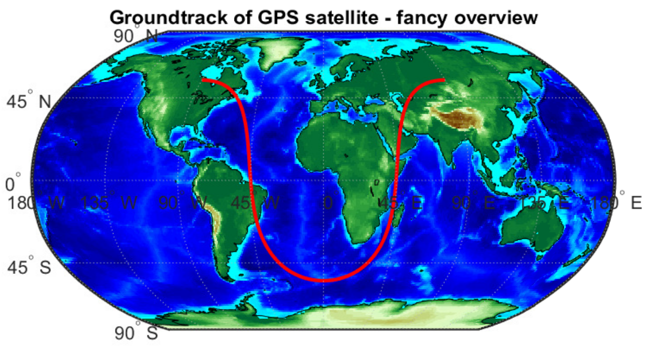


Figure 19: Fancy Eckert IV projection of GPS satellite groundtrack on Earth.

Figure 20 presents the behaviour of α, δ and Ψ parameters over time. A time step of 1 second is chosen for the plots. It can be observed that the secondary axis S (satellite orbit pole) has an oscillatory behaviour, that oscillates with the same time of revolution as that of satellite. The oscillation has an amplitude of 84.8995 deg with a minimum value of 185.1005 deg and a maximum value of 354.8995 deg. The rate of change of δ parameter is somewhat linked to the Ψ parameter. Figure 21 outlines the behaviour of α and δ parameters for one GPS orbit revolution.

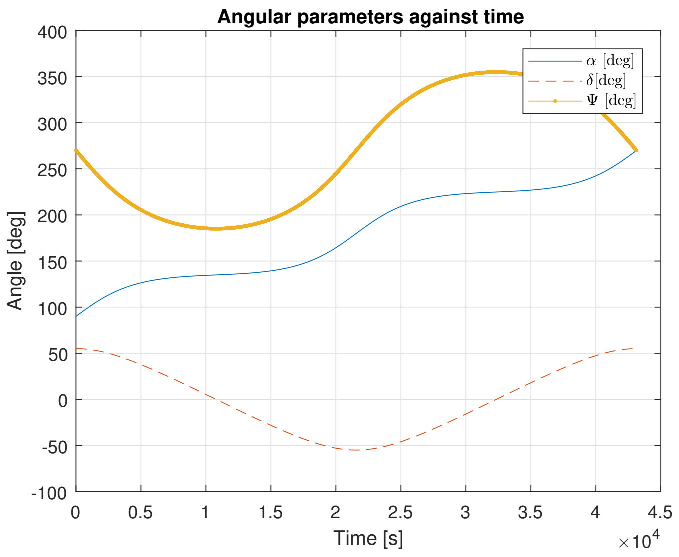


Figure 20: Behaviour of α, δ and Ψ parameters over time for one revolution of satellite.

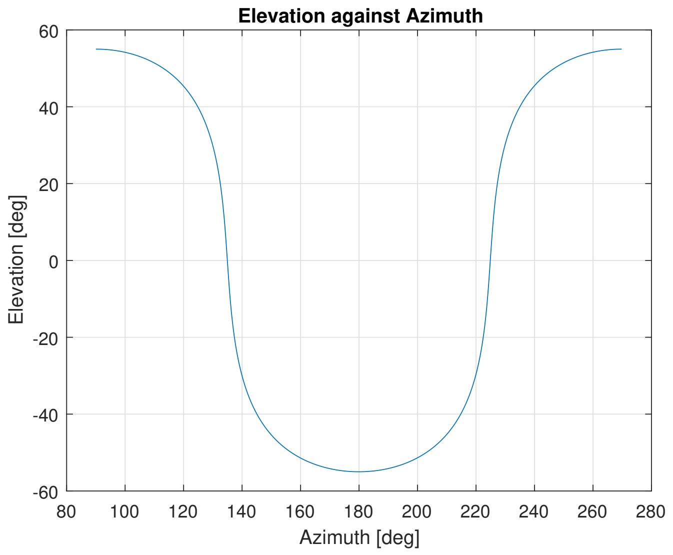


Figure 21: Behaviour of α and δ for one revolution.

Figure 22 shows the behaviour of angular velocity [rad/s] over time for one revolution. An interesting comparison would be to transform the given angular velocity to tangential velocity and compare it against the Kepler solution of velocity obtained with the vis-viva equation outlined in Equation 3[4][pg.51]. The tangential velocity can be obtained by multiplying the angular velocity with the semi-major axis which is the arm length of the satellite (since the orbit is circular). The results are compared in Figure 23. The differences are significant, esp. since the gravitational parameter μ_{Earth} is absent from the dual axis spiral analysis. Furthermore the differences between the solutions are also due to the rotation of Earth which is not taken into account for the vis-viva solution. This is observed with the oscillatory behaviour not present in the vis-viva solution. As seen earlier, the angular velocity of S is also generated around the Euler axis in dual axis spiral problem rather than the orbital axis which is used for vis-viva solution.

$$V^2 = \frac{\mu_{Earth}}{a} \tag{3}$$

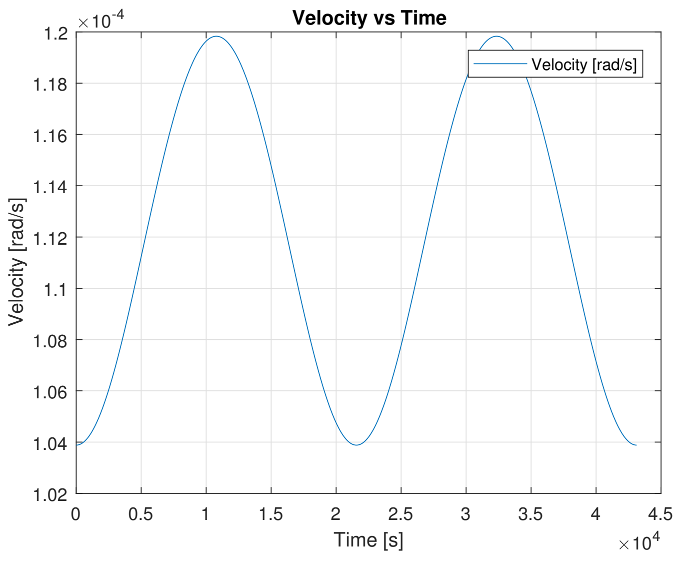


Figure 22: Behaviour of velocity[rad/s] over time for revolution of satellite.

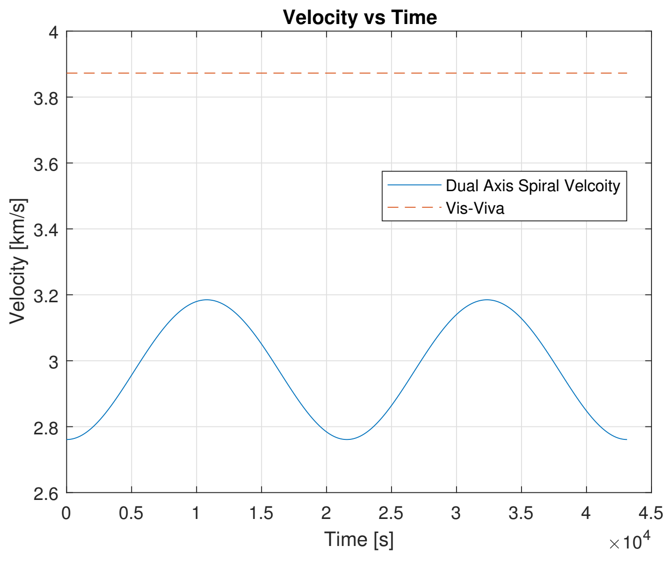


Figure 23: Comparison of velocity obtained via dual axis spiral approach against the vis-viva approach.

6 Matlab Scipt

All files can be found at the following Git source: `https://github.com/Alixir/Mission-geometry-and-orbit-design`
Following function is used to obtain the *hemisphere()* function.

```
function res = hemisphere(theta)
    % Hemisphere function. Takes in values in degrees, outputs 1 or -1
    % In accordance with the definition in A-42, A-43, A-44 OCDM
    % Orbit Constellation Design and Management [OCDM] by James R Wertz 2nd
    % Printing.

    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    if 0 ≤ wrapTo360(theta) && wrapTo360(theta)<180
    res = 1;
    elseif 180 ≤ wrapTo360(theta) && wrapTo360(theta)< 360
    res = -1;
    end
end
```

Following function is used to estimate *acos2()*.

```
function res = acos2( cos_phi, H_phi)
    % Takes in degrees, outputs in degrees. Enter cos(theta) and
    % hemisphere function H(theta)
    % In accordance with the definition in A-42, A-43, A-44 OCDM
    % Orbit Constellation Design and Management [OCDM] by James R Wertz 2nd
    % Printing.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     res = wrapTo360( rad2deg(H_phi*acos(cos_phi)) );
    res = wrapTo360( H_phi*acosd(cos_phi) );
end
```

Following function is used to estimate *atan2()*, named as *myatan2d()*.

Following function is used to estimate *MAL()*.

```
function resC = MAL(A,B,a,b)
    % Middle Angle Law
    % Takes in degrees, outputs degrees
    % In accordance with the definition on pg.792 of OCDM
    % Orbit Constellation Design and Management [OCDM] by James R Wertz 2nd
    % Printing.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    sindC = ( sind(A)*cosd(B)*cosd(b) + sind(B)*cosd(A)*cosd(a) )/( 1 - ...
        sind(a)*sind(b)*sind(A)*sind(B));
    cosdC = ( -cosd(A)*cosd(B) + sind(A)*sind(B)*cosd(a)*cosd(b) )/( 1 - ...
        sind(a)*sind(b)*sind(A)*sind(B) );
%     resC = atan2d(sindC, cosdC);
    resC = myatan2d(sindC, cosdC);
end
```

Following function is used to estimate *MSL()*.

```
function resc = MSL(a, b, A, B)
    % Mean Side Law
    % Takes in degrees, outputs degrees
    % In accordance with the definition on pg.792 of OCDM
    % Orbit Constellation Design and Management [OCDM] by James R Wertz 2nd
    % Printing.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    sindc = ( sind(a)*cosd(b)*cosd(B) + sind(b)*cosd(a)*cosd(A) )/( 1 - ...
        sind(a)*sind(b)*sind(A)*sind(B) );
    cosdc = ( cosd(a)*cosd(b) - sind(a)*sind(b)*cosd(A)*cosd(B))/ (1 - ...
        sind(a)*sind(b)*sind(A)*sind(B));
%     resc = atan2d(sindc, cosdc);
    resc = myatan2d(sindc, cosdc);
end
```

Following is the run script for full sky geometry problem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% AE4878 Mission Geometry and Orbit Design %%%%%%%%%%%
%%%% Assignment 8 - Week 4.4 - FULL SKY 3 - Version 3 %%%%
%%%%% Author Info: Ali Nawaz; Student ID - 4276477 %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Part 1 Angle-Side-Angle

% Available data for spherical triangle
A = 45; % Angle [deg]
B = 60; % Angle [deg]
c = 22; % Side angle [deg]
% % Verification data for spherical triangle
% A = 300.64; % Angle [deg]
% B = 323.95; % Angle [deg]
% c = 35.53; % side [deg]

%  Solution 1
% Define Hc
if 0 ≤ wrapTo360(c) && wrapTo360(c)<180
    Hc = 1;
elseif 180 ≤ wrapTo360(c) && wrapTo360(c)< 360
    Hc = -1;
end
% Define HA
if 0 ≤ wrapTo360(A) && wrapTo360(A)<180
    HA = 1;
elseif 180 ≤ wrapTo360(A) && wrapTo360(A)< 360
    HA = -1;
end
% Define HB
```

```
if 0 ≤ wrapTo360(B) && wrapTo360(B)<180
    HB = 1;
elseif 180 ≤ wrapTo360(B) && wrapTo360(B)< 360
    HB = -1;
end

C1 = acos2(-cosd(A)*cosd(B) + sind(A)*sind(B)*cosd(c), Hc);
a1 = acos2( ( cosd(A) + cosd(B)*cosd(C1) )/( sind(B)*sind(C1) ), HA);
b1 = acos2( ( cosd(B) + cosd(A)*cosd(C1))/(sind(A)*sind(C1)), HB);

% Solution 2
C2 = 360-C1; % [deg]
a2 = wrapTo360(a1 + 180); % [deg]
b2 = wrapTo360(b1 + 180); % [deg]

% Accumulation the results [deg]
res1 = { 'C1', 'C2', 'a1', 'a2', 'b1', 'b2' ; C1, C2, a1, a2, b1, b2 };

%% Part 2 Angle-Angle-Side
% Test data points
A = 45; % Angle [deg]
B = 20; % Angle [deg]
a = 22; % side [deg]

% % Verification data points [deg]
% A = 300.64;
% B = 323.95;
% a = 30;

% Solution 1
b1 = wrapTo360( asind(  ( sind(B)*sind(a) )/( sind(A) ) ) ); % [deg]
c1 = MSL(a,b1,A,B); % [deg]
C1 = MAL(A,B,a,b1); % [deg]

% Solution 2
b2 = wrapTo360( 180 - b1); % [deg]
c2 = MSL(a,b2,A,B); % [deg]
C2 = MAL(A,B,a,b2); % [deg]

% accumulation of results [deg]
res2 = { 'C1', 'C2', 'b1', 'b2', 'c1', 'c2'; C1, C2, b1, b2, c1, c2};
```

Following is the general function to estimate all the parameters of a dual axis spiral problem.

```
function [Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, omegaE, v,Δ_Psi, Psi] = dual_axis_spiral_full(rho1, ...
    rho2,omega1,omega2, phi1_0, phi2_0, phi1, phi2, t)
    % Inputs [deg]:[ rho1, rho2,omega1,omega2, phi1_0, phi2_0, t ]
    % Point P with coordinates (alpha, Δ) is rotating with angular
    % velocity omega2 [deg/s] about the secondary axis S, which in turn is rotating
    % with angular velocity omega1 [deg/s] about the primary or central axis C
    % located at (0 deg, 90 deg)
    % rho1: angular distance of S from C
    % rho2: angular distance of P from S
    % phi1_0 = initial azimuth of S about C relative to alpha = 0
    % phi2_0 = initial azimuth of P about S relative to C
    % t = time vector [s]
    % Outputs: [azimuth, elevation, velocity, direction_of_motion]
    % alpha = Azimuth of P about C [deg]
    % Δ = Elevation of P relative to C [deg]
    % v = Velocity of P [m/s]
    % Psi = Direction of motion of P [deg]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script constructed with the aid of Table 8-8 OCDM
    % Orbit Constellation Design and Management [OCDM] by James R Wertz 2nd
    % Printing.
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if rho1<0 || rho1>180 || rho2<0 || rho2>180
        msg = ('Please enter 0≤ rho1, rho2 ≤180 degrees');
    end
    if isempty(phi1_0) == 1
        phi1 = phi1 ;% [deg] Azimuth of S about C relative to alpha = 0
    else
```

```
        phi1 = phi1_0 + omega1.*t; % [deg] Azimuth of S about C relative to alpha = 0
    end

    if isempty(phi2_0) == 1
        phi2 = phi2 ;% [deg] Azimuth of P about S relative to C
    else
        phi2 = phi2_0 + omega2.*t; % [deg] Azimuth of P about S relative to C
    end
    % Equations of motion for dual axis spiral
    Δ = 90 - acosd( cosd(rho1)*cosd(rho2) + sind(rho1)*sind(rho2)*cosd(phi2) ); % [deg] ...
        Elevation of P relative to C, -90≤ Δ < 90 deg
    Δ_alpha = acos2( ( cosd(rho2) - cosd(rho1)*sind(Δ) )/( sind(rho1)*cosd(Δ) ), ...
        -hemisphere(phi2) );% change in alpha [deg] 0 deg ≤ Δ_alpha <360 deg
    alpha = wrapTo360( phi1 + Δ_alpha); % [deg] alpha parameter, Azimuth of P about C. 0 deg ≤...
         alpha < 360 deg

    % Determining the rotation and orientation of Euler axis, E
    Δ_Ebar = myatan2d(deg2rad(omega2)*sind(rho1),( deg2rad(omega1) + ...
        deg2rad(omega2)*cosd(rho1) )) ; % [deg] co-elevation Δ_E' of Euler axis 0 deg ≤Δ...
        _E'≤180 deg
    omegaE = omega2* ( sind(rho1)/sind(Δ_Ebar) ); % Rotation about Euler axis, E [deg/s]
%     omegaE = sqrt( omega1^2 +omega2^2 + 2*omega1*omega2*cos(rho1)); % [deg/s] Alternate ...
    approach to above solution
    rhoE = acosd( cosd(Δ_Ebar)*sind(Δ) + sind(Δ_Ebar)*cosd(Δ)*cosd(Δ_alpha));% [deg] Angle ...
        from P to Euler axis E (Instantaneous Rotation Axis)
    Δ_Psi = acos2( ( cosd(Δ_Ebar) - cosd(rhoE)*sind(Δ))/( sind(rhoE)*cosd(Δ)), ...
        hemisphere(Δ_alpha));% [deg] Change in direction of Motion of P
    v = deg2rad(omegaE)*sind(rhoE); % [rad/s] Velocity of P 0≤ v ≤ omegaE in radians
    Psi = wrapTo360(Δ_Psi -90); % [deg] 0 deg ≤ Psi < 360 deg
    % Results
    azimuth = alpha;
    elevation = Δ;
    velocity = v;
    direction_of_motion = Psi;
end
```

Following is the run script for verifying the dual axis spiral problem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% AE4878 Mission Geometry and Orbit Design %%%%%%%%%%%
%%%% Assignment 8 - Week 4.4 - FULL SKY 8 - Version 1 %%%%
%%%%% Author Info: Ali Nawaz; Student ID - 4276477 %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% GENERAL DUAL AXIS SPIRAL PROGRAM RUN
% Available data for verification
% All input angles are in degrees, ouputs too apart from omegaE = rad/s and
% v = rad/s.

% Set 1
rho1 = 40;
rho2 = 20;
omega1 = 0; % [deg/s]
omega2 = rad2deg(3); % [deg/s]
phi1_0 = [];
phi2_0 = [];
phi1 = 5;
phi2 = 90;
t = 1;
[Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, omegaE, v,Δ_Psi, Psi] = dual_axis_spiral_full(rho1, ...
    rho2,omega1,omega2, phi1_0, phi2_0, phi1, phi2, t);

results1  = [Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, deg2rad(omegaE), v,Δ_Psi, Psi]';

% Set 2
rho1 = 40;
rho2 = 20;
omega1 = rad2deg(1); % [deg/s]
omega2 = rad2deg(3); % [deg/s]
phi1_0 = [];
phi2_0 = [];
phi1 = 5;
phi2 = 90;
t = 1;
[Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, omegaE, v,Δ_Psi, Psi] = dual_axis_spiral_full(rho1, ...
    rho2,omega1,omega2, phi1_0, phi2_0, phi1, phi2, t);

results2  = [Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, deg2rad(omegaE), v,Δ_Psi, Psi]';


% Set 3
rho1 = 40;
rho2 = 20;
omega1 = rad2deg(1); % [deg/s]
omega2 = rad2deg(3); % [deg/s]
phi1_0 = [];
phi2_0 = [];
phi1 = 5;
phi2 = 100;
t = 1;
[Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, omegaE, v,Δ_Psi, Psi] = dual_axis_spiral_full(rho1, ...
    rho2,omega1,omega2, phi1_0, phi2_0, phi1, phi2, t);

results3  = [Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, deg2rad(omegaE), v,Δ_Psi, Psi]';

% Set 4
rho1 = 40;
rho2 = 20;
omega1 = rad2deg(1); % [deg/s]
omega2 = rad2deg(3); % [deg/s]
phi1_0 = [];
phi2_0 = [];
phi1 = 5;
phi2 = 300;
t = 1;
[Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, omegaE, v,Δ_Psi, Psi] = dual_axis_spiral_full(rho1, ...
    rho2,omega1,omega2, phi1_0, phi2_0, phi1, phi2, t);

results4  = [Δ, Δ_alpha,alpha,Δ_Ebar,rhoE, deg2rad(omegaE), v,Δ_Psi, Psi]';

% All benchmark results
results = [ results1, results2, results3,  results4];
```

Following script outlines the function written to extend the dual axis spiral problem to GPS applications.

```
function [azimuth, elevation, velocity, direction_of_motion] = dual_axis_spiral(rho1, ...
    rho2,omega1,omega2, phi1_0, phi2_0, phi1, phi2, t)
    % Inputs [deg]:[ rho1, rho2,omega1,omega2, phi1_0, phi2_0, t ]
    % Point P with coordinates (alpha, Δ) is rotating with angular
    % velocity omega2 [deg/s] about the secondary axis S, which in turn is rotating
    % with angular velocity omega1 [deg/s] about the primary or central axis C
    % located at (0 deg, 90 deg)
    % rho1: angular distance of S from C
    % rho2: angular distance of P from S
    % phi1_0 = initial azimuth of S about C relative to alpha = 0
    % phi2_0 = initial azimuth of P about S relative to C
    % t = time vector [s]
    % Outputs: [azimuth, elevation, velocity, direction_of_motion]
    % alpha = Azimuth of P about C [deg]
    % Δ = Elevation of P relative to C [deg]
    % v = Velocity of P [m/s]
    % Psi = Direction of motion of P [deg]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script constructed with the aid of Table 8-8 OCDM
    % Orbit Constellation Design and Management [OCDM] by James R Wertz 2nd
    % Printing.
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if rho1<0 || rho1>180 || rho2<0 || rho2>180
        msg = ('Please enter 0≤ rho1, rho2 ≤180 degrees');
    end
    if isempty(phi1_0) == 1
        phi1 = phi1 ;% [deg] Azimuth of S about C relative to alpha = 0
    else
        phi1 = phi1_0 + omega1.*t; % [deg] Azimuth of S about C relative to alpha = 0
    end

    if isempty(phi2_0) == 1
```

```
        phi2 = phi2 ;% [deg] Azimuth of P about S relative to C
    else
        phi2 = phi2_0 + omega2.*t; % [deg] Azimuth of P about S relative to C
    end
    % Equations of motion for dual axis spiral
    ∆ = 90 - acosd( cosd(rho1)*cosd(rho2) + sind(rho1)*sind(rho2)*cosd(phi2) ); % [deg] ...
        Elevation of P relative to C, -90≤ ∆ < 90 deg
    ∆_alpha = acos2( ( cosd(rho2) - cosd(rho1)*sind(∆) )/( sind(rho1)*cosd(∆) ), ...
        -hemisphere(phi2) );% change in alpha [deg] 0 deg ≤ ∆_alpha <360 deg
    alpha = wrapTo360( phi1 + ∆_alpha); % [deg] alpha parameter, Azimuth of P about C. 0 deg ≤...
         alpha < 360 deg

    % Determining the rotation and orientation of Euler axis, E
    ∆_Ebar = myatan2d(deg2rad(omega2)*sind(rho1),( deg2rad(omega1) + ...
        deg2rad(omega2)*cosd(rho1) )) ; % [deg] co-elevation ∆_E' of Euler axis 0 deg ≤∆...
        _E'≤180 deg
    omegaE = omega2* ( sind(rho1)/sind(∆_Ebar) ); % Rotation about Euler axis, E [deg/s]
%     omegaE = sqrt( omega1^2 +omega2^2 + 2*omega1*omega2*cos(rho1)); % [deg/s] Alternate ...
    approach to above solution
    rhoE = acosd( cosd(∆_Ebar)*sind(∆) + sind(∆_Ebar)*cosd(∆)*cosd(∆_alpha));% [deg] Angle ...
        from P to Euler axis E (Instantaneous Rotation Axis)
    ∆_Psi = acos2( ( cosd(∆_Ebar) - cosd(rhoE)*sind(∆))/( sind(rhoE)*cosd(∆)), ...
        hemisphere(∆_alpha));% [deg] Change in direction of Motion of P
    v = deg2rad(omegaE)*sind(rhoE); % [m/s] Velocity of P 0≤ v ≤ omegaE in radians
    Psi = wrapTo360(∆_Psi -90); % [deg] 0 deg ≤ Psi < 360 deg
    % Results
    azimuth = alpha;
    elevation = ∆;
    velocity = v;
    direction_of_motion = Psi;
end
```

Following is the Matlab function for generating ground tracks on an Eckert IV Earth projection.

```
function gt = groundtrack(lat, lon,type,titleinfo)
    % Ground track generator. Input lattitude and longitude values in
    % degrees. Currently two types are supported. 'simple' type, plots
    % groundmap over solid coloured land, lakes and rivers. 'fancy' type,
    % plots groundmap over a terrain textured world map. Add the title
    % label to add to the figures.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script Author: Ali Nawaz, Delft University of Technology.
    % Aerospace Engineering Faculty [LR]
    % Mechanical, Maritime and Materials Engineering Faculty [3ME]
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if strcmp(type,'simple')== 1
        landareas = shaperead('landareas.shp','UseGeoCoords',true); % import landareas
        figure;
        load geoid; % load geoid shape
        axesm eckert4; % load Eckert4 projection coordinates
        framem; gridm; % show frame and grid
        axis on
        hold on
        ax = worldmap('World'); % plot world map
        land = shaperead('landareas', 'UseGeoCoords', true);
        geoshow(ax, land, 'FaceColor', [0.5 0.7 0.5]); % show lands
        lakes = shaperead('worldlakes', 'UseGeoCoords', true);
        geoshow(lakes, 'FaceColor', 'blue'); % show lakes
        rivers = shaperead('worldrivers', 'UseGeoCoords', true);
        geoshow(rivers, 'Color', 'blue'); % show rivers
        geoshow(lat,lon,'LineWidth', 2, 'color', 'r'); % show ground track
        title(titleinfo);
        hold off
    elseif strcmp(type,'fancy')== 1
        figure;
        hold on
        ∆ = lat; % store latitude of ground track
        alpha = lon; % store longitude of ground track
        load coast; % load coastlines
        load topo; % load topography information
        axesm eckert4; % load Eckert4 projection coordinates
        framem; gridm; % turn on grids and frames
        axis on; % show axis
        ax = worldmap('World'); % Show world map
        land = shaperead('landareas', 'UseGeoCoords', true);
        geoshow(ax, land, 'FaceAlpha', [0]);% Show coordinates
        geoshow(topo,topolegend,'Displaytype','texturemap') % show rough topography
        demcmap(topo) % add colormap appropriate to terrain elevation data, note it doesn't ...
            show the ice sheets!
        geoshow(lat,long,'Color','k'); % add borders to coastline
        geoshow(Δ,alpha,'LineWidth', 2, 'color', 'r'); % show the ground track
        title(titleinfo);
        hold off
    end
end
```

Following script is the run file used to generate results for GPS application problem.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% AE4878 Mission Geometry and Orbit Design %%%%%%%%%%%
%%%% Assignment 8 - Week 4.4 - FULL SKY 8 - Version 2 %%%%
%%%%% Author Info: Ali Nawaz; Student ID - 4276477 %%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% GPS satellite orbit parameters
GPS_alt = 20200; % Orbit altitude [km]
RE= 6378.136; % Radius of Earth[km]
mu_E = 398600.441; % Gravitational parameter of Earth[ km^3/s^2]
agps = (GPS_alt + RE); % Semi major axis of GPS satellite [km]
rho1 = 55; % Angular distance of S from C [deg]
rho2 = 90; % Angular distance of P from S [deg]
omega1 = rad2deg(-(2*pi)/(86164.1004));
omega2 = rad2deg(sqrt( mu_E/( (GPS_alt + RE)^3)));
phi1_0 = 90; % intial azimuth S about C [deg]
phi2_0 = 0; % initial azimuth P about S [deg]
phi1 = [];
phi2 = [];
t = [0:1:sqrt( ((agps^3)/(mu_E))*(4*pi^2)) ]; % Time of one S/C revolution
alpha = []; % stores propagated azimuth values
Δ = []; % stores propagated elevation values
v = []; % stores propagated velocity
Psi = []; % Store propagated direction of motion of P
for k = 1:length(t)
    % run the dual_axis_spiral function for given time stamp
    [azimuth, elevation, velocity, direction_of_motion] = dual_axis_spiral(rho1, ...
        rho2,omega1,omega2, phi1_0, phi2_0,phi1,phi2, t(k));
    alpha = [alpha,azimuth];
    Δ = [Δ, elevation];
    v = [v, velocity];
    Psi = [Psi, direction_of_motion];
end
% Generate ground track of GPS satellite on Earth map
groundtrack(Δ, alpha,'fancy','Groundtrack of GPS satellite');
% Plot elevation against azimuth
figure(2)
plot(wrapTo360(alpha-180),Δ)
title('Elevation against Azimuth');
xlabel('Azimuth [deg]');
ylabel('Elevation [deg]');
grid on
% Plot different angles against time
figure(3)
plot(t, wrapTo360(alpha-180),'-', t, Δ, '--', t, Psi,'.-')
legend({'$\alpha$ [deg]', '$\Δ$[deg]', '$\Psi$ [deg]'},'Interpreter','latex')
title('Angular parameters against time');
ylabel('Angle [deg]');
xlabel('Time [s]');
grid on
% Plot angular velocity [rad/s] against time
figure(4)
plot(t,v)
legend('Velocity [rad/s]')
title( 'Velocity vs Time')
xlabel('Time [s]')
ylabel('Velocity [rad/s]')
grid on

% Plot velocity [km/s] against time of vis viva and above solution.
figure(5)
plot(t,v.*agps,'-', t , sqrt(mu_E/agps).*ones(size(t)),'--');
legend('Dual Axis Spiral Velcoity','Vis-Viva','Location','Best');
```

```
title( 'Velocity vs Time')
xlabel('Time [s]')
ylabel('Velocity [km/s]')
grid on
```

References

[1] GPS satellite altitude. https://www.gps.gov/systems/gps/space/.

[2] Picture of Spherical Triangle. https://en.wikipedia.org/wiki/Spherical_trigonometry#/media/File:Spherical_trigonometry_basic_triangle.svg.

[3] R. Noomen. *AE4878 Mission Geometry and Orbit Design, Full Sky Geometry v4-13*. TU Delft, 2017.

[4] James R. Wertz. *Orbit Constellation Design Management*. Springer, 2009.