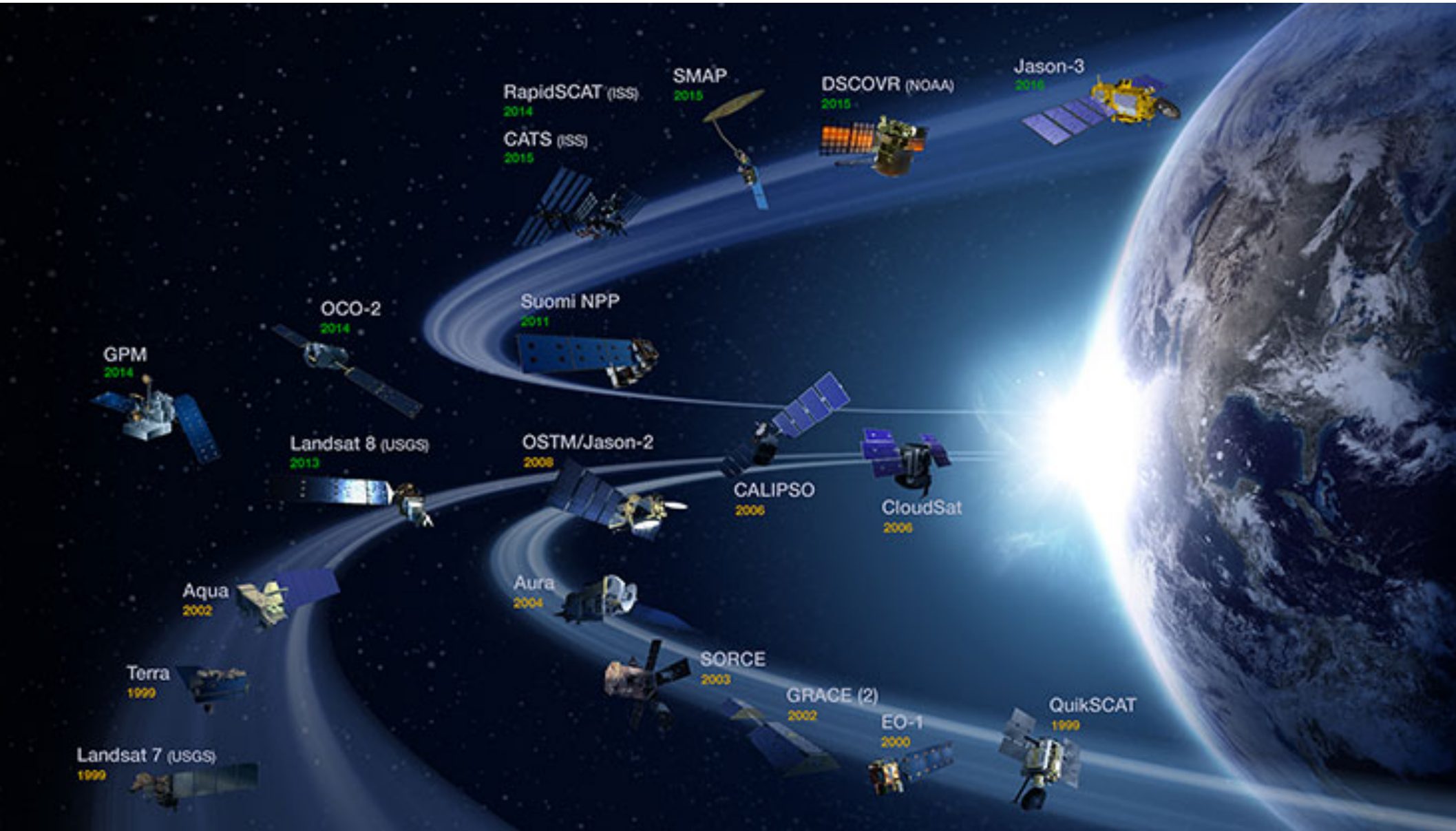
Fundamentals of Remote Sensing

L1 - Celestial mechanics

for remote sensing specialists

Main question: where is satellite with our sensor at given moment ?



Reference frame definitions

Time scales

- second, day
- atomic time TAI
- UTC
- solar time and UT1

Center of universe

- Earth mass center

Spatial coordinate system

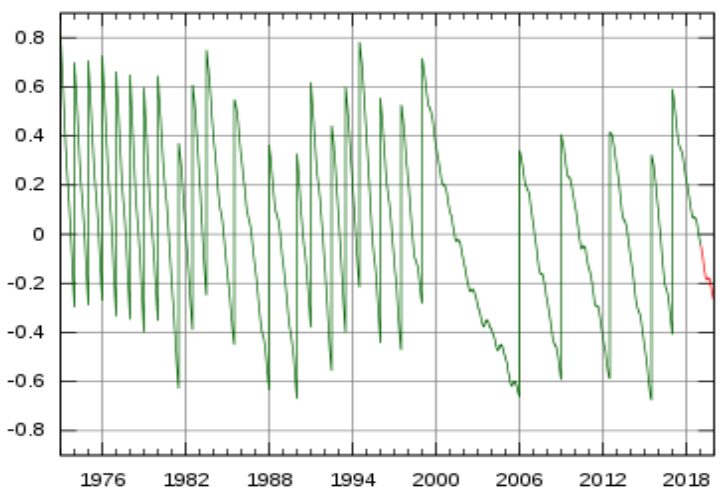
- (kilo) meter
- Cartesian or polar coordinates
- where to place reference points?

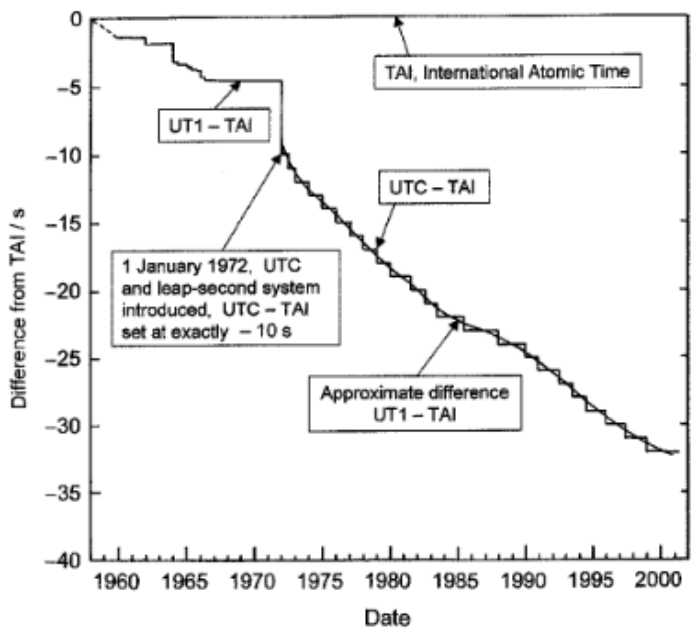
Time scales

TAI (clock rate) – International Atomic Time (French name: **T**emps **A**tomique **I**nternational). TAI is a weighted average of the time kept by over 400 atomic clocks in over 50 national laboratories worldwide

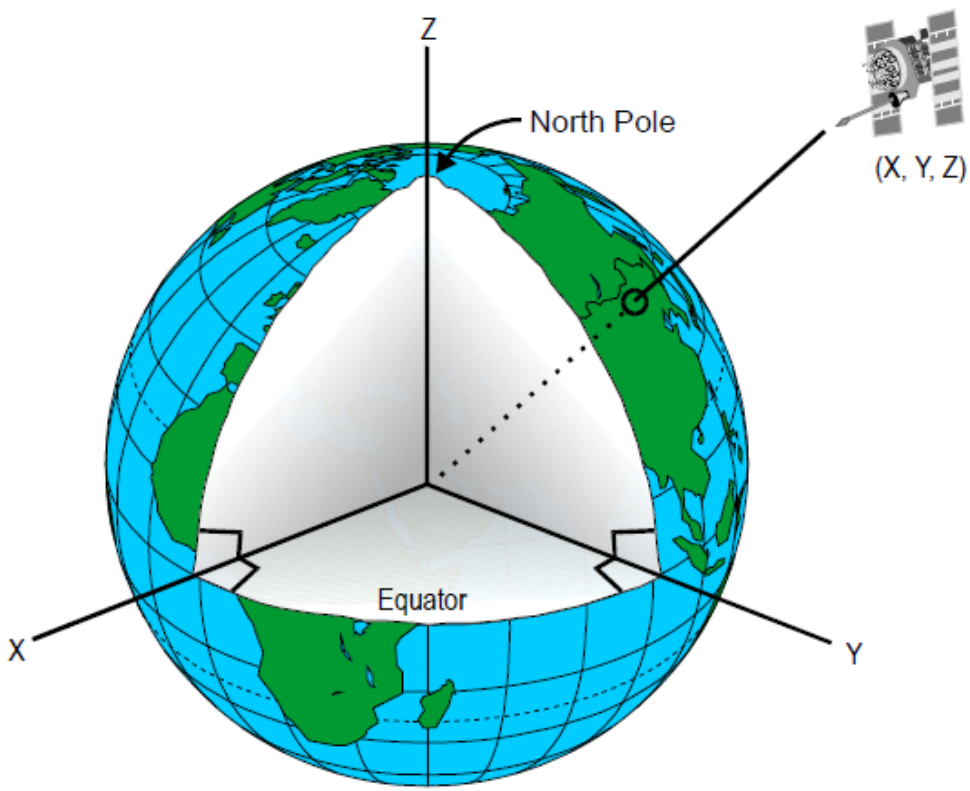
UT1 (solar time) – mean solar time at 0° longitude. It is proportional to the rotation angle of the Earth with respect to distant stars

UTC (wall time) - Coordinated Universal Time. It is the primary time standard by which the world regulates clocks and time. It is a discontinuous (i.e. regularly adjusted by leap seconds) time scale composed from segments that are linear transformations of TAI.
https://en.wikipedia.org/wiki/Leap_second





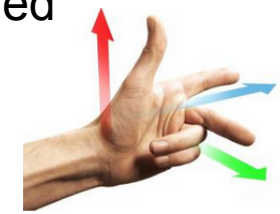
Earth Centered, Earth Fixed (ECEF) Coordinates



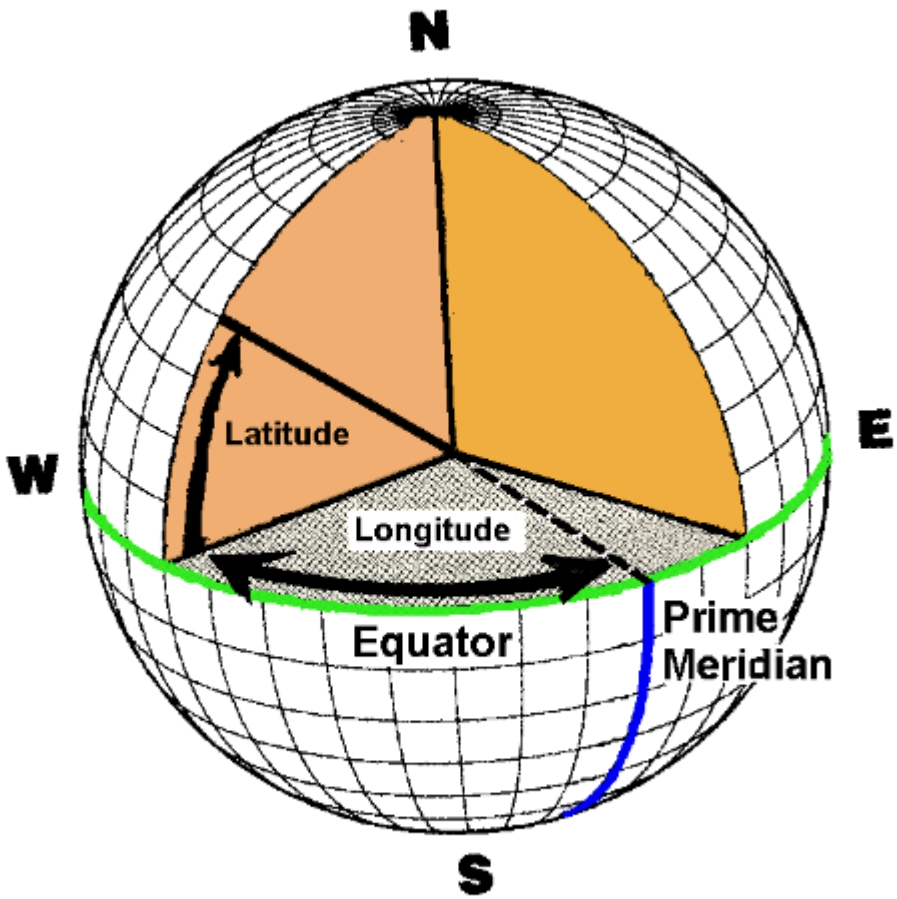
Z – The positive Z axis goes out the north pol.The X-Y plane will be the equatorial plane.

X – The X-Axis is along the prime meridian (zero point of longitude)

Y – The Y axis is then set to make the system right handed



Earth Centered, Earth Fixed (ECEF) Latitude and Longitude on Spherical Earth



Latitude for a Spherical Earth

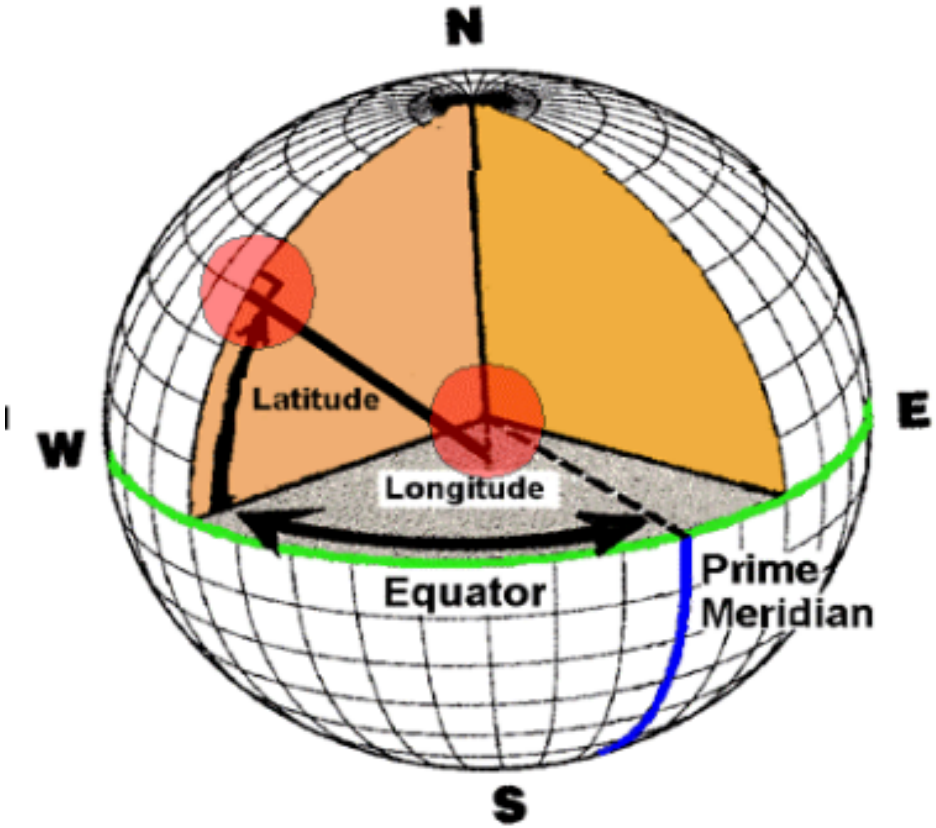
For a spherical earth polar coordinate angles seen from the center of the earth

The angle up from the equator is **latitude**

The angle in the equatorial plane is the **longitude**. The zero line, called the prime meridian, is taken, by convention, as the line through Greenwich England

Height above Earth surface

Earth Centered, Earth Fixed (ECEF) Latitude and Longitude on Ellipsoidal Earth

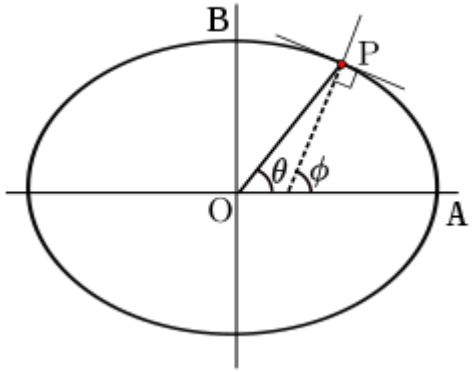


The longitude is the exactly the same as for a spherical earth

The extension to an ellipsoidal earth is to use the line perpendicular to the ellipsoid to define the vertical

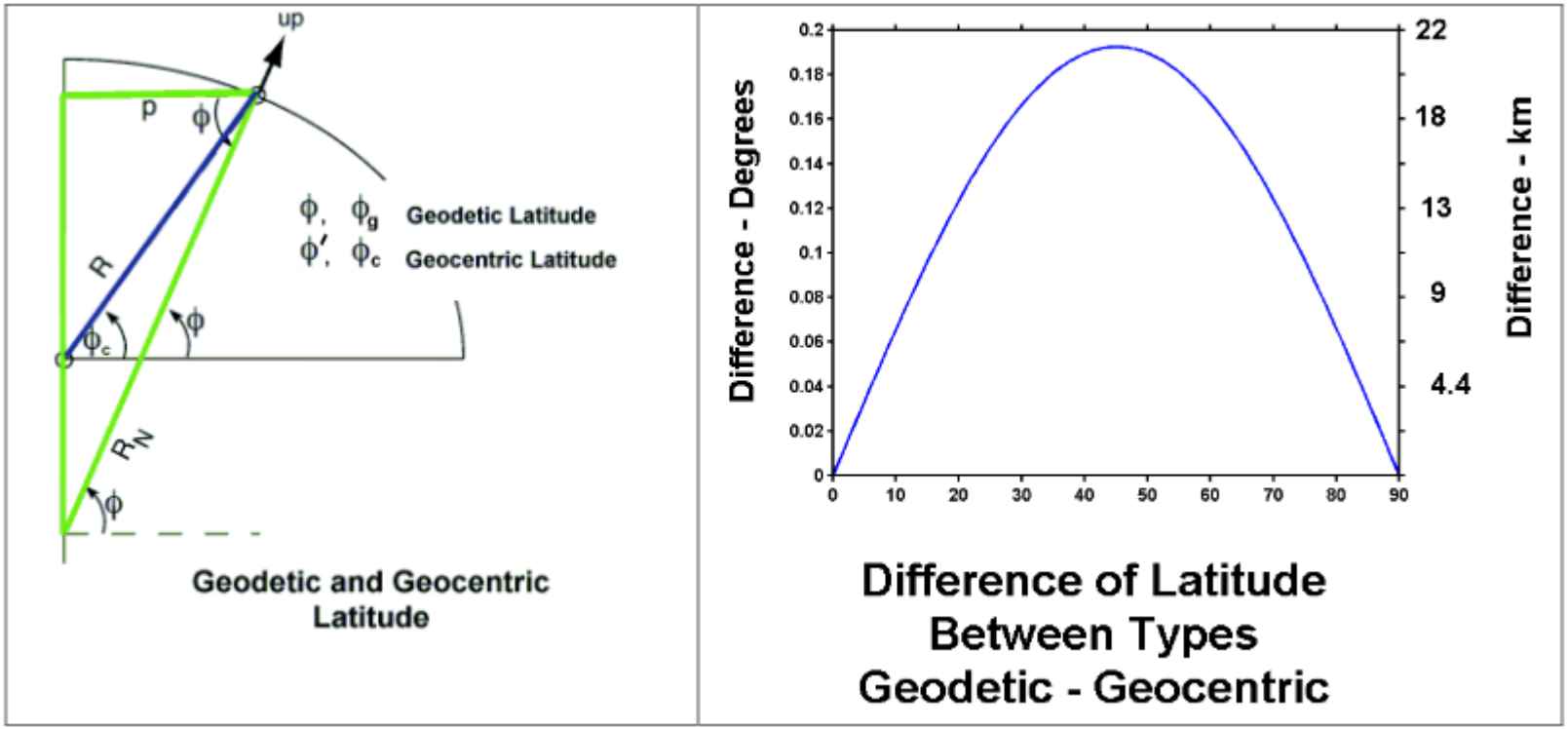
The angle from the center of the earth is called the **geocentric latitude**

The line perpendicular to the ellipsoid makes an angle with the equatorial plane that is called the **geodetic latitude**



+ **Height** above Ellipsoid surface (MSL)

Geocentric vs Geodetic Latitude



Geodetic coordinates (latitude ϕ, longitude λ, height h) can be converted into ECEF coordinates using the following equation:

$$X = (N(\phi) + h)\cos\phi\cos\lambda$$
$$Y = (N(\phi) + h)\cos\phi\sin\lambda$$
$$Z = \left(\frac{b^2}{a^2}N(\phi) + h\right)\sin\phi$$

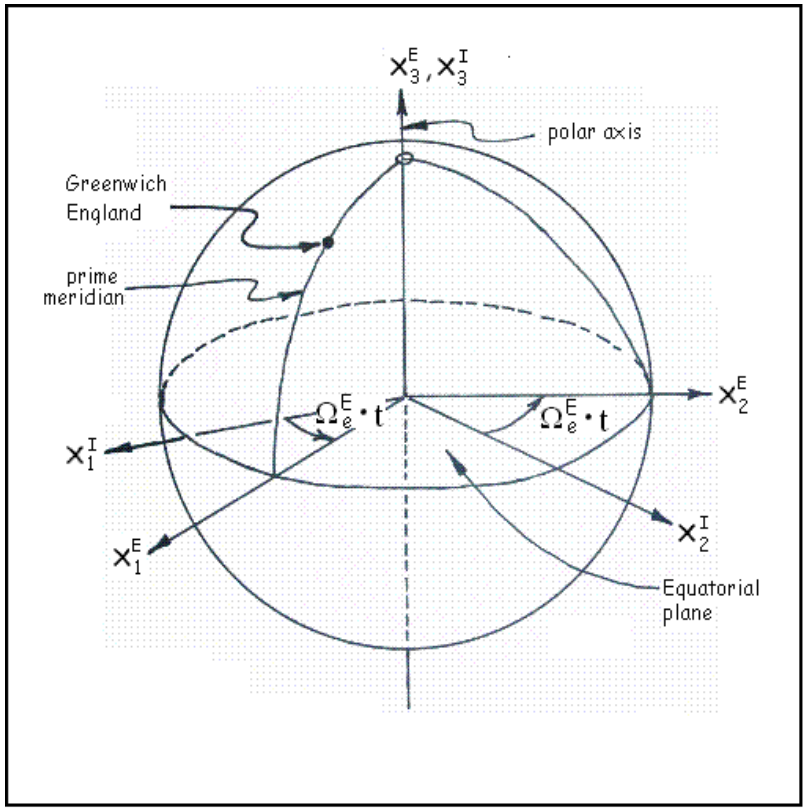
https://en.wikipedia.org/wiki/Geographic_coordinate_conversion

where

$$N(\phi) = \frac{a^2}{\sqrt{a^2\cos^2\phi + b^2\sin^2\phi}} = \frac{a}{\sqrt{1 - e^2\sin^2\phi}},$$

WGS84 ellipsoid:
a = 6378137
b = 6356752

Earth Centered Inertial (ECI) coordinates



International Celestial Reference System (**ICRS**)
~ equatorial coordinate system

The fundamental plane and the primary direction are aligned with Earth's equator and pole,
- does not rotate with the Earth
- remains fixed against the background stars

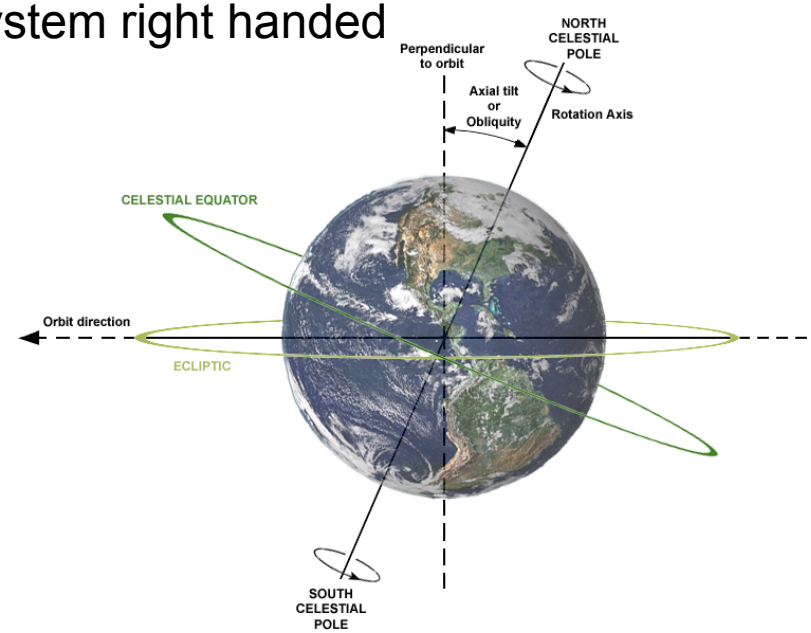
Z – The positive Z axis goes out the north pol

X – The primary direction toward the **vernal equinox**

Y – to make the system right handed

$$\theta(t_U) = 2\pi(0.779\,057\,273\,2640 + 1.002\,737\,811\,911\,354\,48 t_U)$$

where t_U is the Julian UT1 date − 2451545.0.



Newton's laws of mechanics + Newton's law of universal gravitation solve the problem

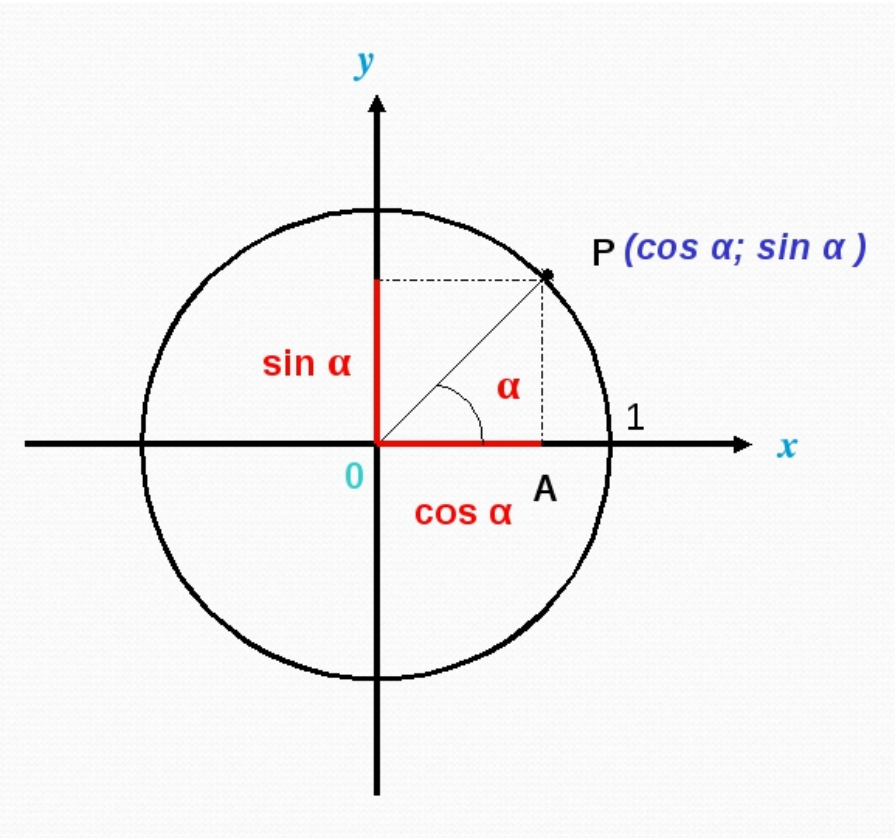
Model assumptions:
- Two body problem
- Only central gravitational force
- M >> m

$$m * \ddot{x} = F_x = \frac{G * M * m}{(x^2 + y^2 + z^2)} \frac{x}{\sqrt{x^2 + y^2 + z^2}}$$

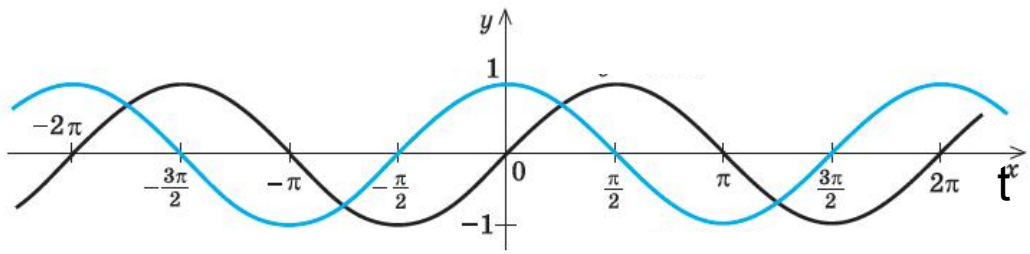
$$m * \ddot{y} = F_y = \frac{G * M * m}{(x^2 + y^2 + z^2)} \frac{y}{\sqrt{x^2 + y^2 + z^2}}$$

$$m * \ddot{z} = F_z = \frac{G * M * m}{(x^2 + y^2 + z^2)} \frac{z}{\sqrt{x^2 + y^2 + z^2}}$$

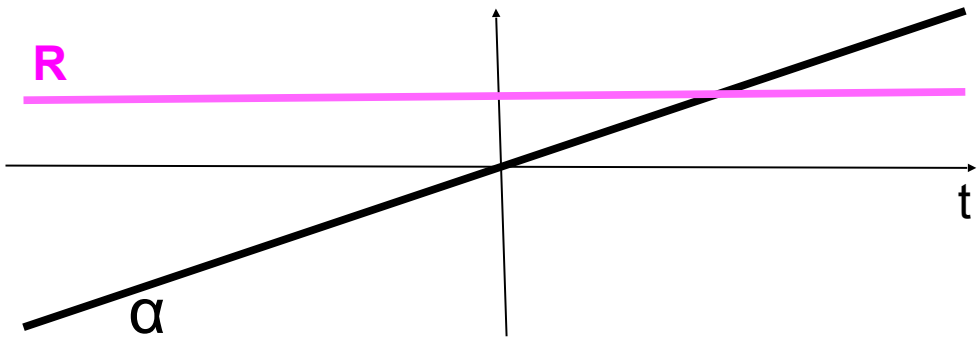
Variable change trick



X - Y

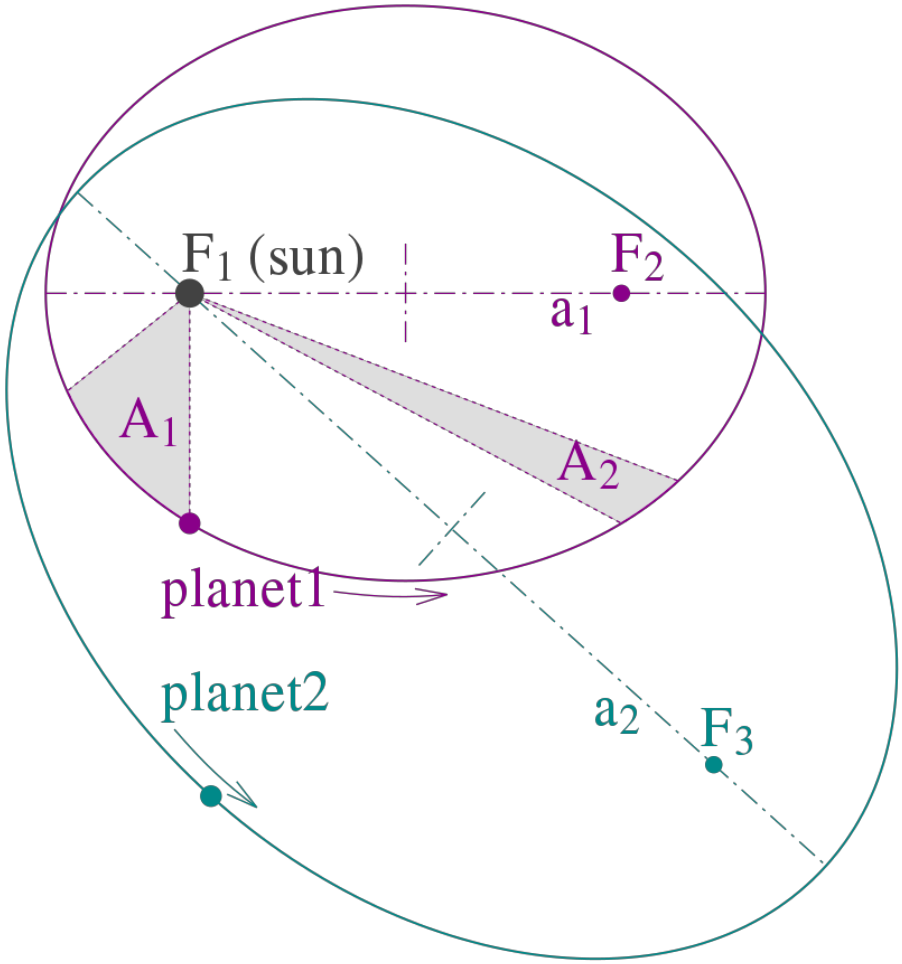


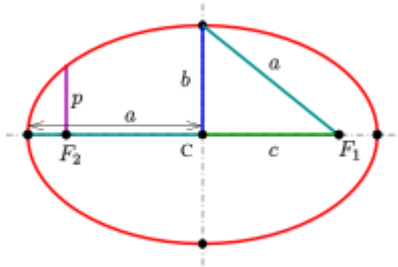
Angle - radius



1st Kepler's laws of planetary motion

The orbits are ellipses, with focal points F1 and F2 for the first planet and F1 and F3 for the second planet. The Sun is placed in focal point F1.





$$\varepsilon = \sqrt{1 - \frac{b^2}{a^2}}.$$

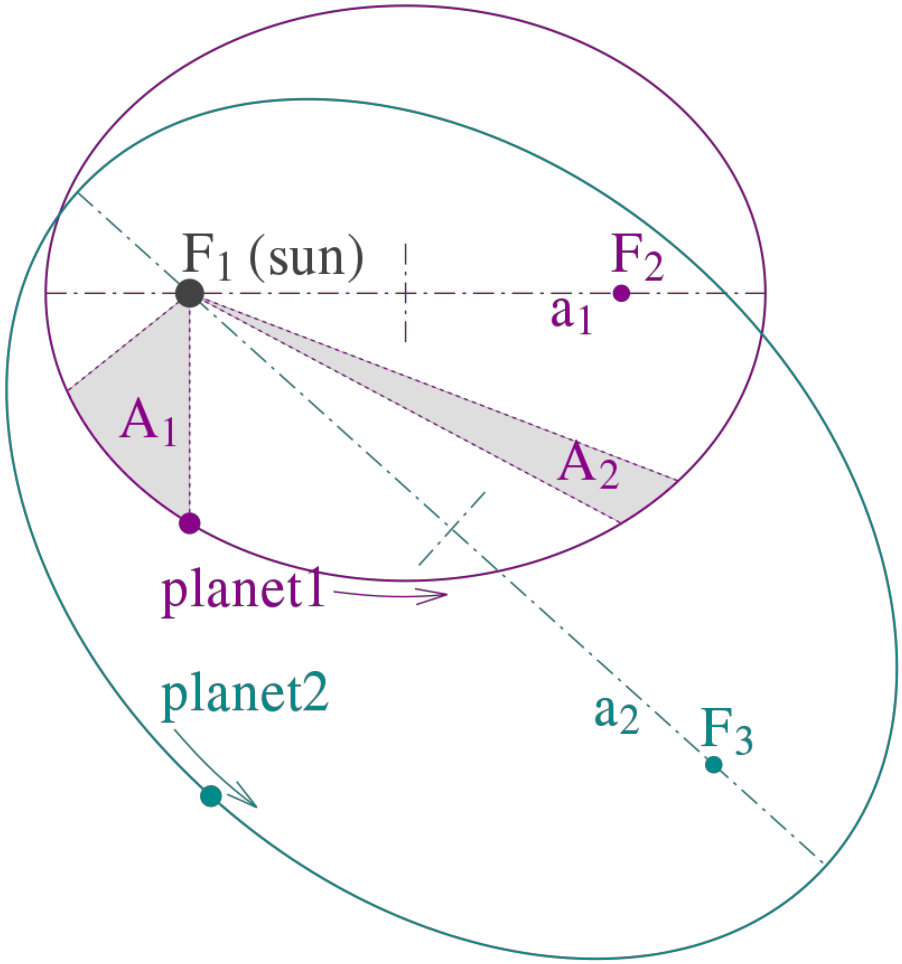
$$p = \frac{b^2}{a}.$$

$$r = \frac{p}{1 + \varepsilon \cos\theta},$$

2nd Kepler's laws of planetary motion

The two shaded sectors A1 and A2 have the same surface area and the time for planet 1 to cover segment A1 is equal to the time to cover segment A2

A line segment joining a planet and the Sun sweeps out equal areas during equal intervals of time



Areal velocity is constant:

$$\frac{dA}{dt} = \frac{1}{2} r^2 \frac{d\theta}{dt}.$$

For the full cycle (period T)- full ellipse area:

$$T \frac{1}{2} r^2 \frac{d\theta}{dt} = \pi ab$$

3d Kepler's laws of planetary motion

The total orbit times for planet 1 and planet 2 have a ratio

$$\frac{T_1^2}{T_2^2} = \frac{a_1^3}{a_2^3}$$

Using Newton's Law of gravitation, this relation can be found in the case of a circular orbit by setting the centripetal force equal to the gravitational force:

$$mr\left(\frac{2\pi}{T}\right)^2 = G\frac{mM}{r^2} \rightarrow T^2 = \left(\frac{4\pi^2}{GM}\right)r^3 \rightarrow T^2 \propto r^3$$

Kepler's equation. Propagation in time

Compute the position of a planet as a function of time T

1. Compute the **mean anomaly** $M = nt$ where n is the mean motion.

   $n \cdot P = 2\pi$ radians where P is the period.

2. Compute the **eccentric anomaly** E by solving Kepler's equation:

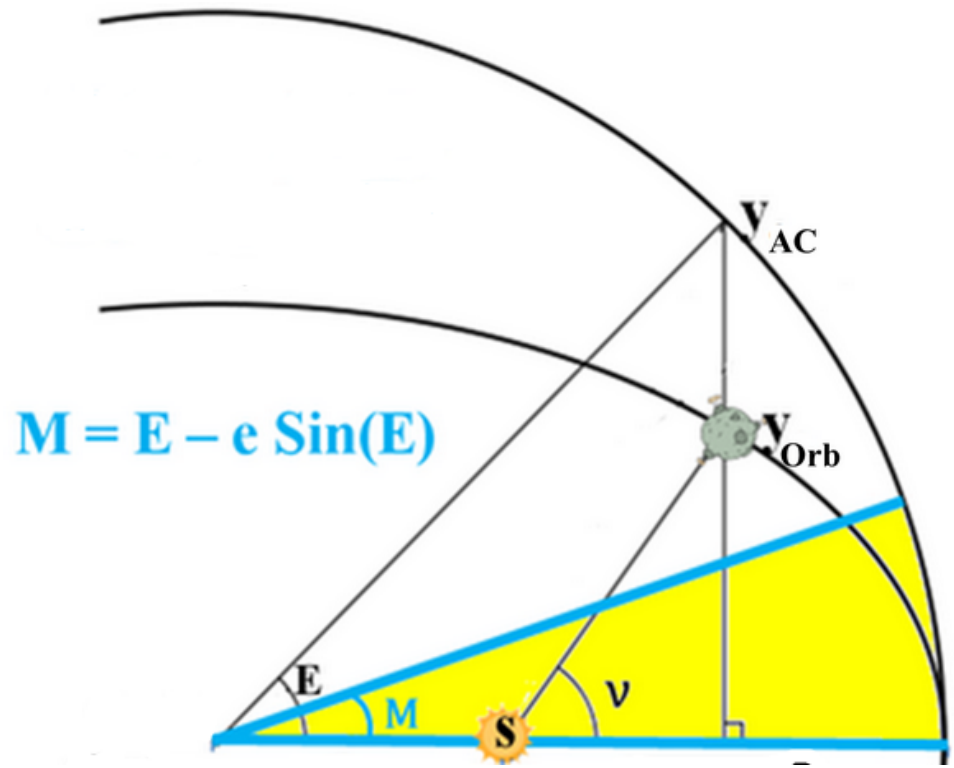
   $$M = E - \varepsilon \sin E$$

3. Compute the **true anomaly** θ by the equation:

   $$(1-\varepsilon)\tan^2\frac{\theta}{2} = (1+\varepsilon)\tan^2\frac{E}{2}$$

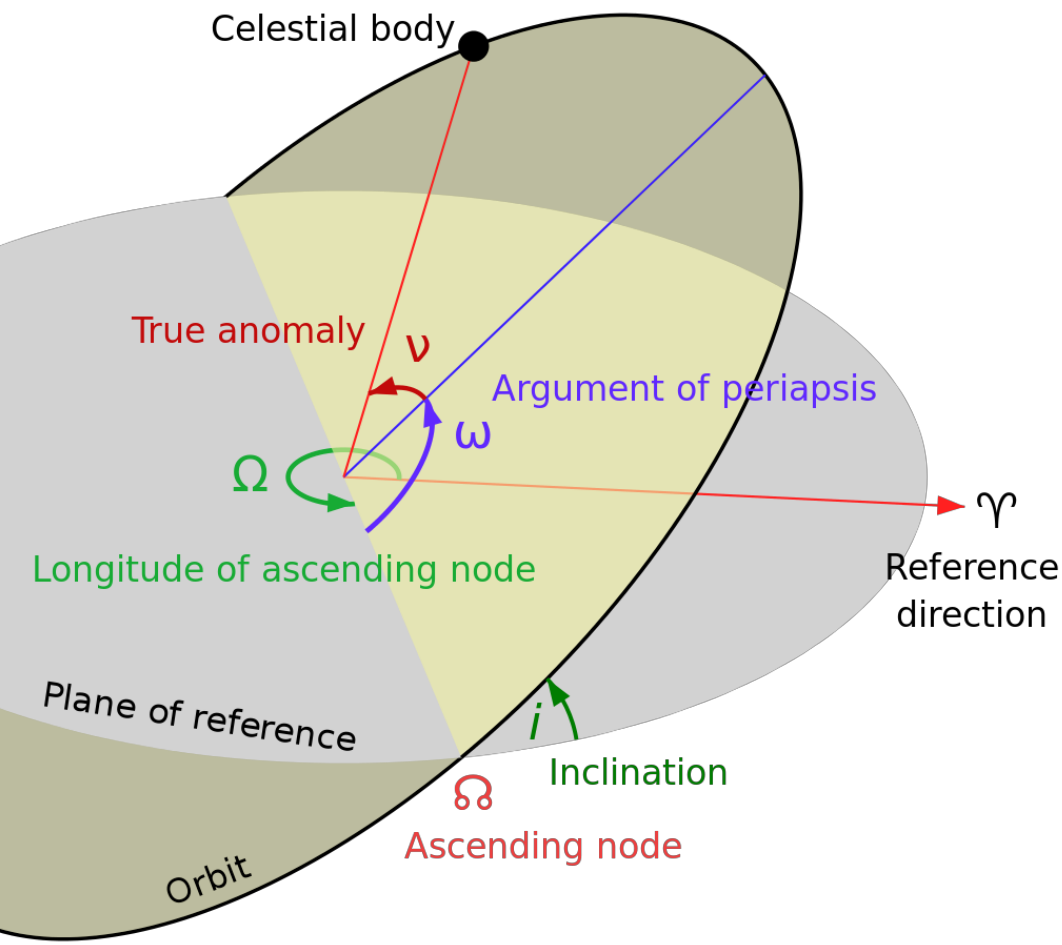
4. Compute the **heliocentric distance**

   $$r = a(1-\varepsilon\cos E).$$



Keplerian orbital elements

State vector (x,y,z, Vx,Vy, Vz) => (A,e,i, Ω, ω, M) orbital (Kepler) elements



A – Semimajor axis (period, MM)

e – Eccentricity, shape of the ellipse

i – Inclination, vertical tilt of the ellipse with respect to the reference plane

Ω - Longitude of the ascending node, horizontally orients the ascending node of the ellipse

ω - Argument of periapsis defines the orientation of the ellipse in the orbital plane

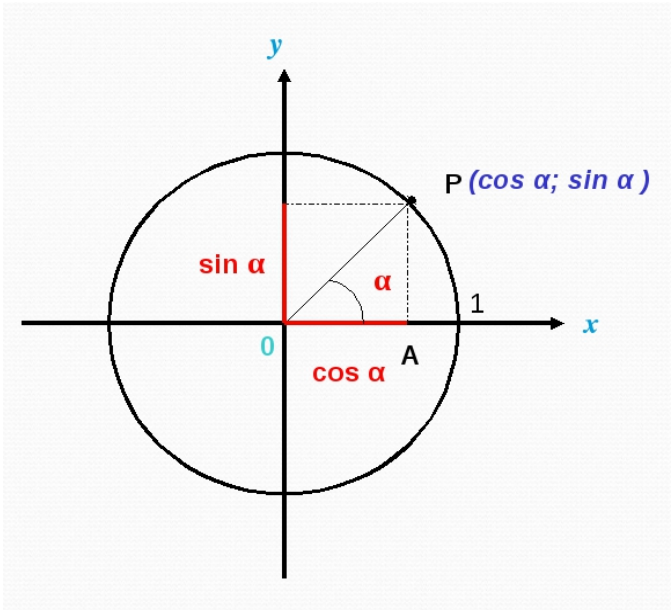
v_0 - True anomaly defines the position of the orbiting body along the ellipse at a specific time

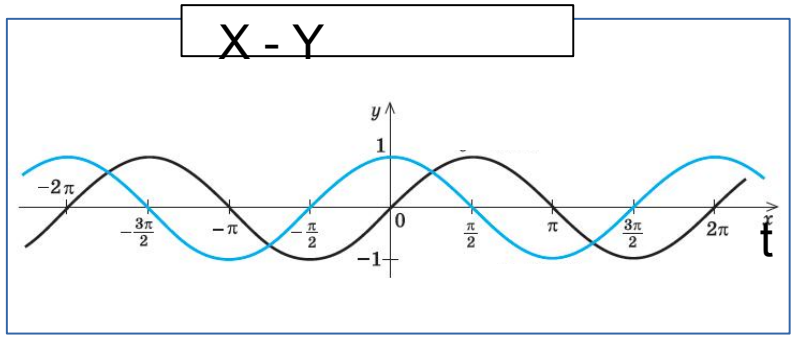
Keplerian orbital elements

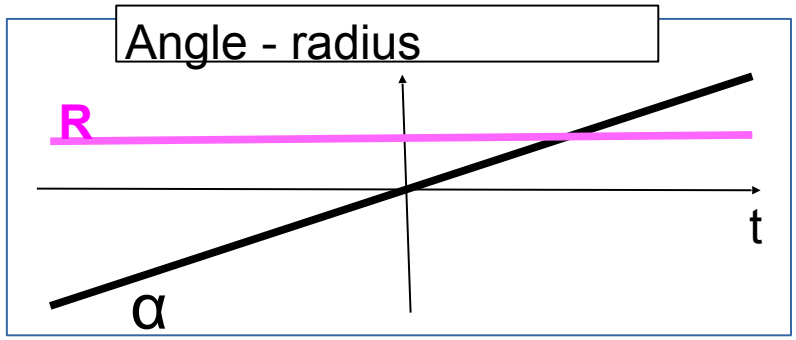
Under ideal conditions of a perfectly spherical central body and zero perturbations:

if at any instant t0 the orbital parameters are [e_0, a_0, i_0, Ω_0, ω_0, M_0], then the elements at time t0 + δt is given by [e_0, a_0, i_0, Ω_0, ω_0, $M_0 + n\,\delta t$]

Convertion between Cartesian and Keplerian state vectors
(http://orbitsimulator.com/formulas/OrbitalElements.html)







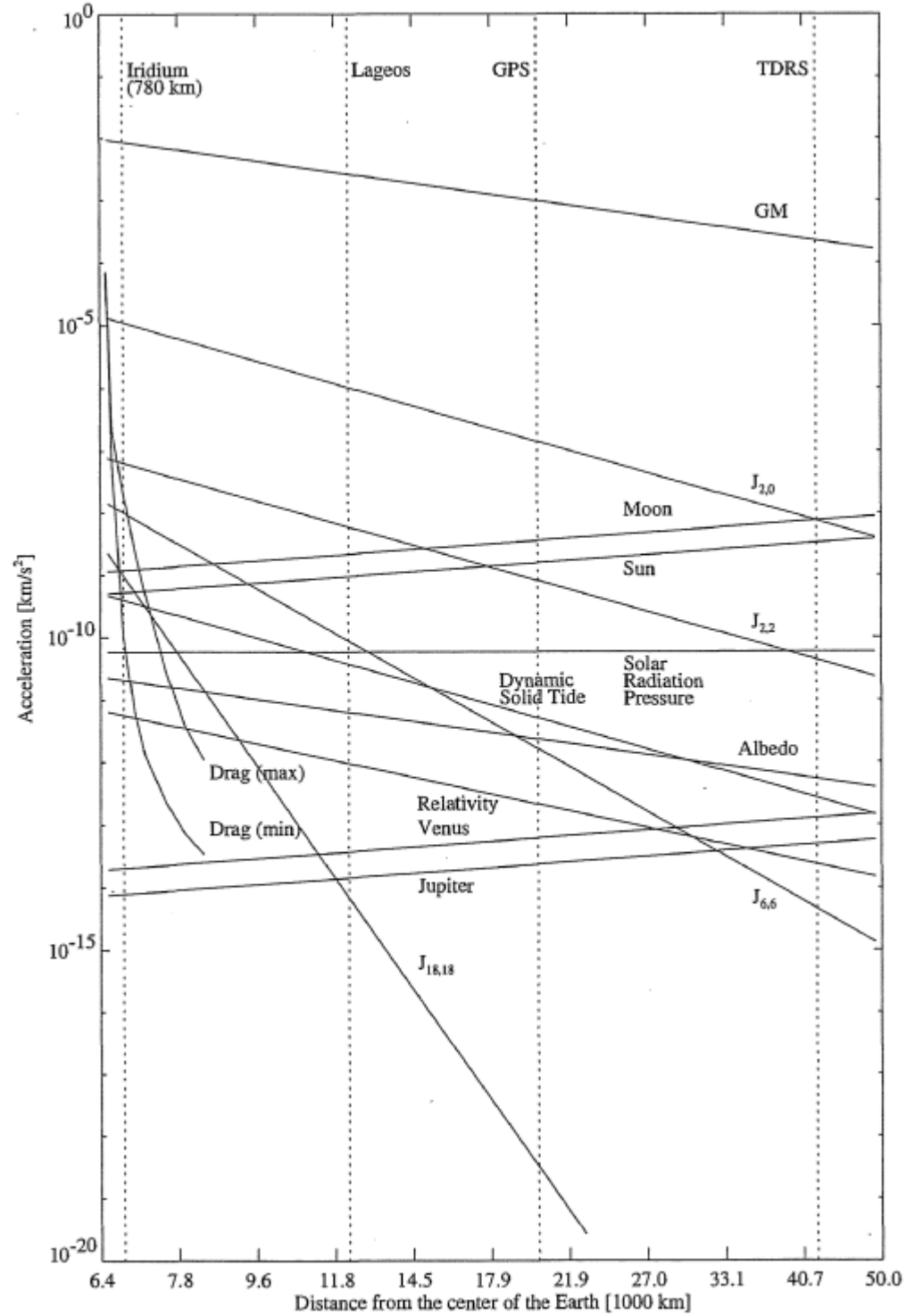
Perturbation forces

- The effect of the Earth flattening. First spherical harmonics J_2 causes orbit precession

$$\overset{1.}{\Delta\Omega} = -2\pi \frac{J_2}{\mu\, p^2} \frac{3}{2} \cos i$$

$$\overset{1.}{\Delta\omega} = \Delta\omega_1 + \Delta\omega_2 = -2\pi \frac{J_2}{\mu\, p^2} 3 \left(\frac{5}{4} \sin^2 i - 1 \right)$$

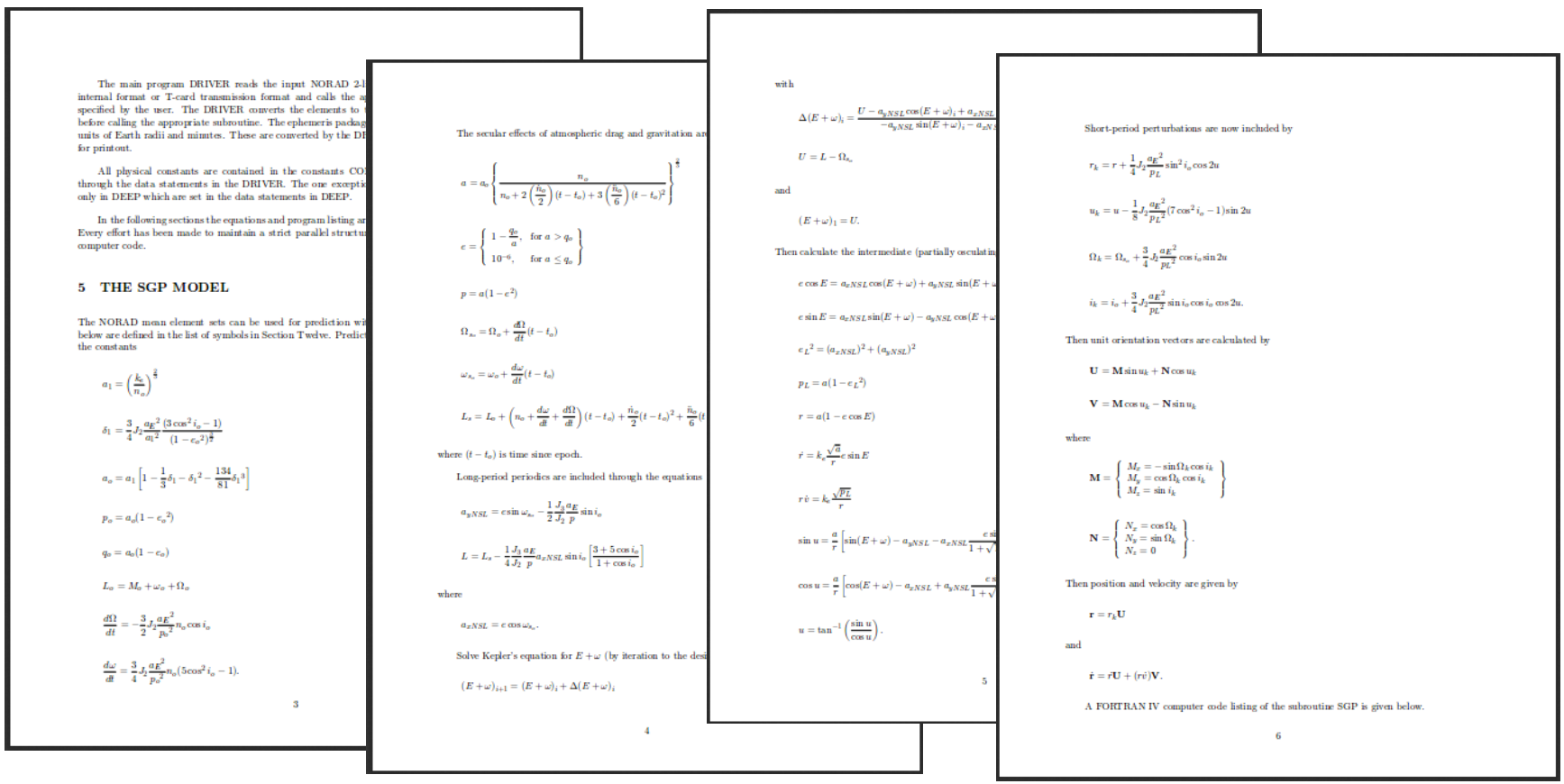
- atmospheric drag
- Moon, Sun, planets gravity
- Earth and ocean tides
- solar radiation pressure



SDP/SGP orbital models

Simplified perturbations models predict the effect of perturbations caused by the Earth's shape, drag, radiation, and gravitation effects from other bodies such as the sun and moon

- "*Spacetrack Report #3. Models for Propagation of NORAD Element Sets*" (http://www.celestrak.com/NORAD/documentation/spacetrk.pdf)
- "*Revisiting Spacetrack Report #3*" (http://celestrak.com/publications/AIAA/2006-6753/)



The main program DRIVER reads the input NORAD 2-l
internal format or T-card transmission format and calls the a
specified by the user. The DRIVER converts the elements to
before calling the appropriate subroutine. The ephemeris packag
units of Earth radii and minutes. These are converted by the DI
for printout.

All physical constants are contained in the constants CO
through the data statements in the DRIVER. The one exceptio
only in DEEP which are set in the data statements in DEEP.

In the following sections the equations and program listing ar
Every effort has been made to maintain a strict parallel structu
computer code.

5 THE SGP MODEL

The NORAD mean element sets can be used for prediction wi
below are defined in the list of symbols in Section Twelve. Predic
the constants

$$a_1 = \left(\frac{k_e}{n_o}\right)^{\frac{2}{3}}$$

$$\delta_1 = \frac{3}{4} J_2 \frac{a_E^2}{a_1^2} \frac{(3\cos^2 i_o - 1)}{(1-e_o^2)^{\frac{3}{2}}}$$

$$a_o = a_1\left[1 - \frac{1}{3}\delta_1 - \delta_1^2 - \frac{134}{81}\delta_1^3\right]$$

$$p_o = a_o(1-e_o^2)$$

$$q_o = a_o(1-e_o)$$

$$L_o = M_o + \omega_o + \Omega_o$$

$$\frac{d\Omega}{dt} = -\frac{3}{2} J_2 \frac{a_E^2}{p_o^2} n_o \cos i_o$$

$$\frac{d\omega}{dt} = \frac{3}{4} J_2 \frac{a_E^2}{p_o^2} n_o (5\cos^2 i_o - 1).$$

3

The secular effects of atmospheric drag and gravitation ar

$$a = a_o\left\{\frac{n_o}{n_o + 2\left(\frac{\dot{n}_o}{2}\right)(t-t_o) + 3\left(\frac{\ddot{n}_o}{6}\right)(t-t_o)^2}\right\}^{\frac{2}{3}}$$

$$e = \left\{\begin{array}{ll} 1 - \frac{q_o}{a}, & \text{for } a > q_o \\ 10^{-6}, & \text{for } a \le q_o \end{array}\right\}$$

$$p = a(1-e^2)$$

$$\Omega_{s_o} = \Omega_o + \frac{d\Omega}{dt}(t-t_o)$$

$$\omega_{s_o} = \omega_o + \frac{d\omega}{dt}(t-t_o)$$

$$L_s = L_o + \left(n_o + \frac{d\omega}{dt} + \frac{d\Omega}{dt}\right)(t-t_o) + \frac{\dot{n}_o}{2}(t-t_o)^2 + \frac{\ddot{n}_o}{6}(t$$

where $(t-t_o)$ is time since epoch.

Long-period periodics are included through the equations

$$a_{yNSL} = e\sin\omega_{s_o} - \frac{1}{2}\frac{J_3}{J_2}\frac{a_E}{p}\sin i_o$$

$$L = L_s - \frac{1}{4}\frac{J_3}{J_2}\frac{a_E}{p} a_{xNSL}\sin i_o\left[\frac{3+5\cos i_o}{1+\cos i_o}\right]$$

where

$$a_{xNSL} = e\cos\omega_{s_o}.$$

Solve Kepler's equation for $E+\omega$ (by iteration to the desi

$$(E+\omega)_{i+1} = (E+\omega)_i + \Delta(E+\omega)_i$$

4

with

$$\Delta(E+\omega)_i = \frac{U - a_{yNSL}\cos(E+\omega)_i + a_{xNSL}}{-a_{yNSL}\sin(E+\omega)_i - a_{xN}}$$

$$U = L - \Omega_{s_o}$$

and

$$(E+\omega)_1 = U.$$

Then calculate the intermediate (partially osculatin

$$e\cos E = a_{xNSL}\cos(E+\omega) + a_{yNSL}\sin(E+\omega$$

$$e\sin E = a_{xNSL}\sin(E+\omega) - a_{yNSL}\cos(E+\omega$$

$$e_L^2 = (a_{xNSL})^2 + (a_{yNSL})^2$$

$$p_L = a(1-e_L^2)$$

$$r = a(1 - e\cos E)$$

$$\dot{r} = k_e\frac{\sqrt{a}}{r}e\sin E$$

$$r\dot{v} = k_e\frac{\sqrt{pL}}{r}$$

$$\sin u = \frac{a}{r}\left[\sin(E+\omega) - a_{yNSL} - a_{xNSL}\frac{e\,s}{1+\sqrt{}}\right.$$

$$\cos u = \frac{a}{r}\left[\cos(E+\omega) - a_{xNSL} + a_{yNSL}\frac{e\,s}{1+\sqrt{}}\right.$$

$$u = \tan^{-1}\left(\frac{\sin u}{\cos u}\right).$$

5

Short-period perturbations are now included by

$$r_k = r + \frac{1}{4}J_2\frac{a_E^2}{p_L}\sin^2 i_o\cos 2u$$

$$u_k = u - \frac{1}{8}J_2\frac{a_E^2}{p_L^2}(7\cos^2 i_o - 1)\sin 2u$$

$$\Omega_k = \Omega_{s_o} + \frac{3}{4}J_2\frac{a_E^2}{pL^2}\cos i_o\sin 2u$$

$$i_k = i_o + \frac{3}{4}J_2\frac{a_E^2}{pL^2}\sin i_o\cos i_o\cos 2u.$$

Then unit orientation vectors are calculated by

$$\mathbf{U} = \mathbf{M}\sin u_k + \mathbf{N}\cos u_k$$

$$\mathbf{V} = \mathbf{M}\cos u_k - \mathbf{N}\sin u_k$$

where

$$\mathbf{M} = \left\{\begin{array}{l} M_x = -\sin\Omega_k\cos i_k \\ M_y = \cos\Omega_k\cos i_k \\ M_z = \sin i_k \end{array}\right\}$$

$$\mathbf{N} = \left\{\begin{array}{l} N_x = \cos\Omega_k \\ N_y = \sin\Omega_k \\ N_z = 0 \end{array}\right\}.$$

Then position and velocity are given by

$$\mathbf{r} = r_k\mathbf{U}$$

and

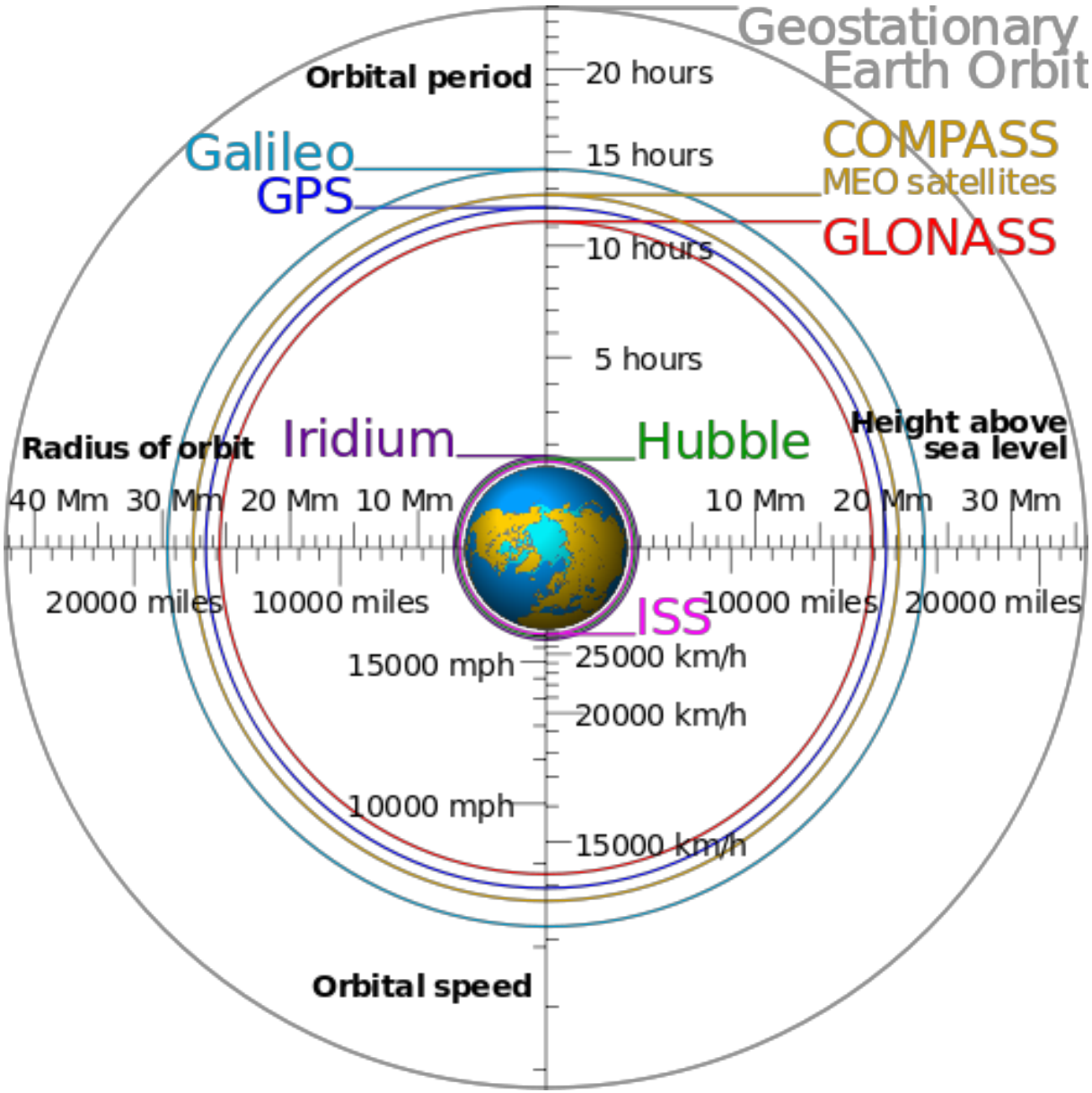
$$\dot{\mathbf{r}} = \dot{r}\mathbf{U} + (r\dot{v})\mathbf{V}.$$

A FORTRAN IV computer code listing of the subroutine SGP is given below.

6

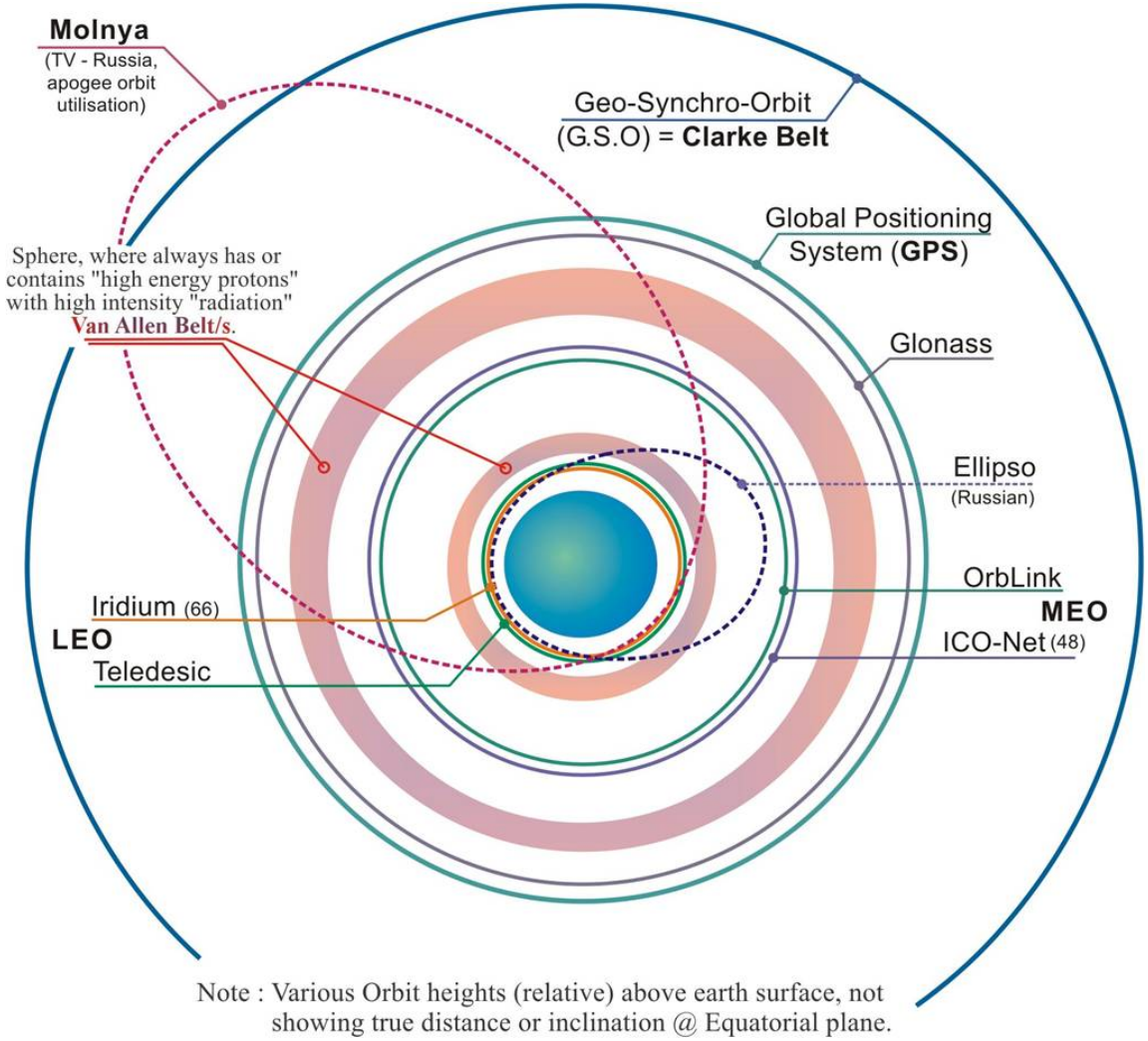
Orbit types. Radius

LEO – 160 – 2000км
MEO – 2000 – 35000km
GEO - 35,786 km (42,164 R)



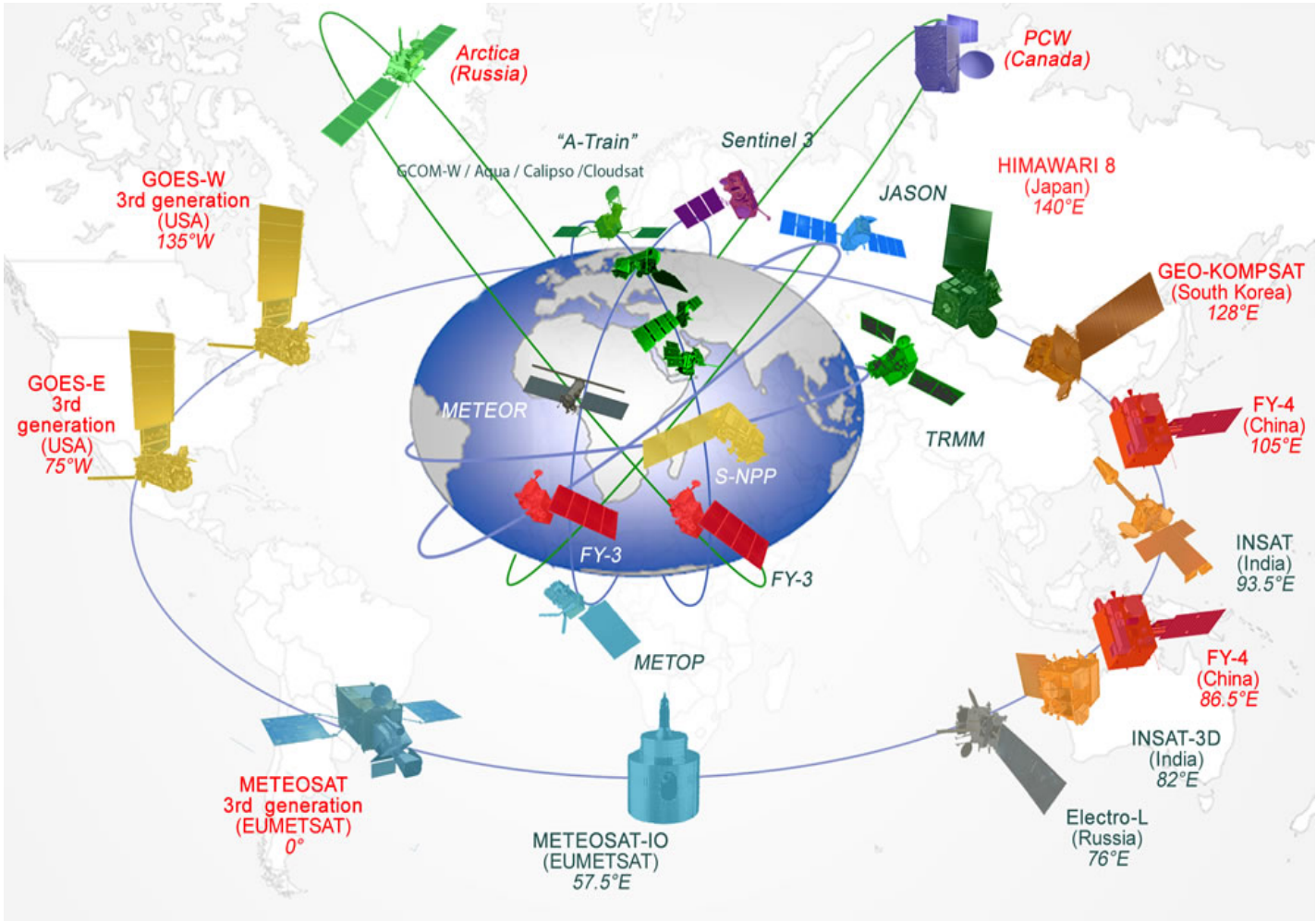
Van Allen radiation belts

A Van Allen radiation belt is a zone of energetic charged particles, most of which originate from the solar wind, that are captured by and held around a planet by that planet's magnetic field



I – 2-4 th. km
II - 12-20 th. rm

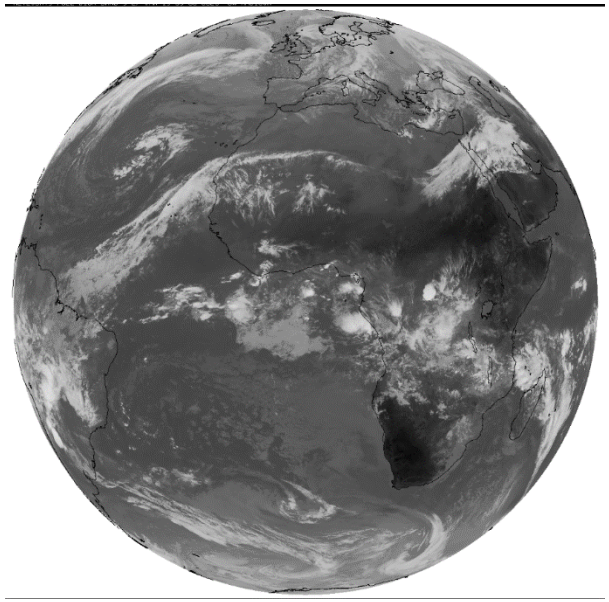
Geostationary satellite for Remote sensing

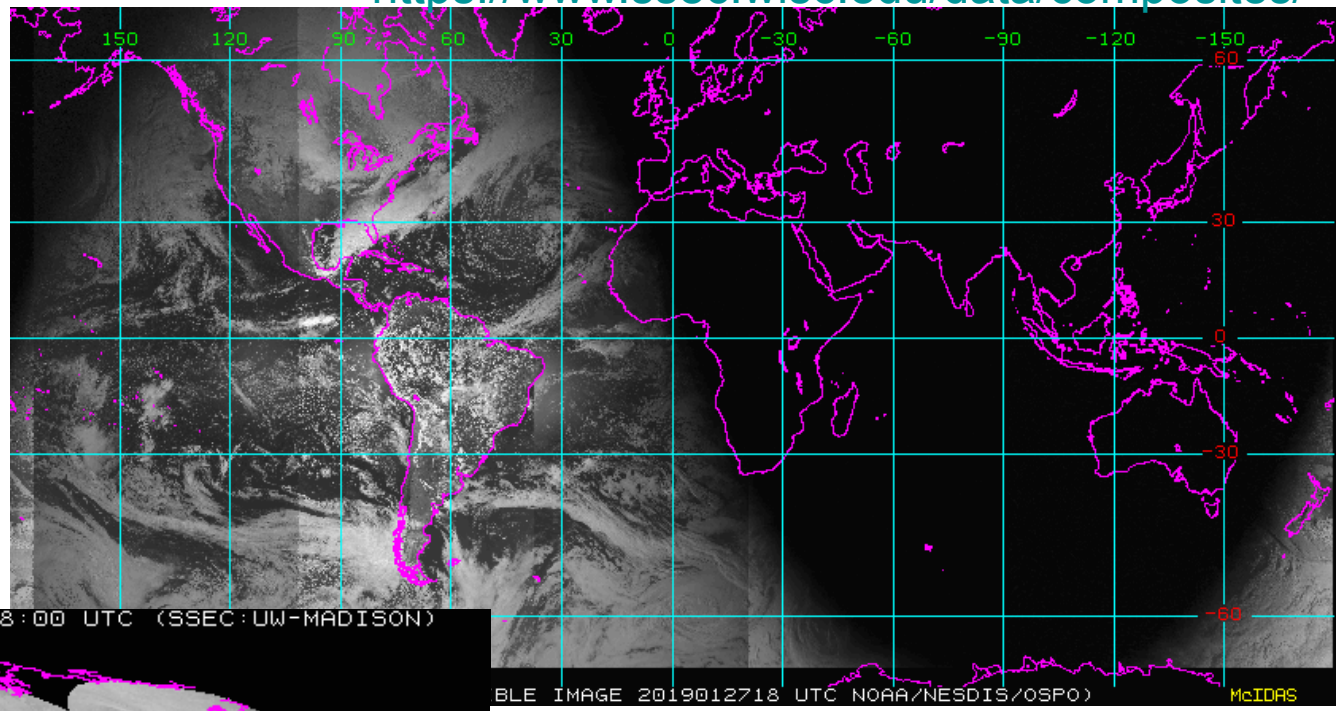


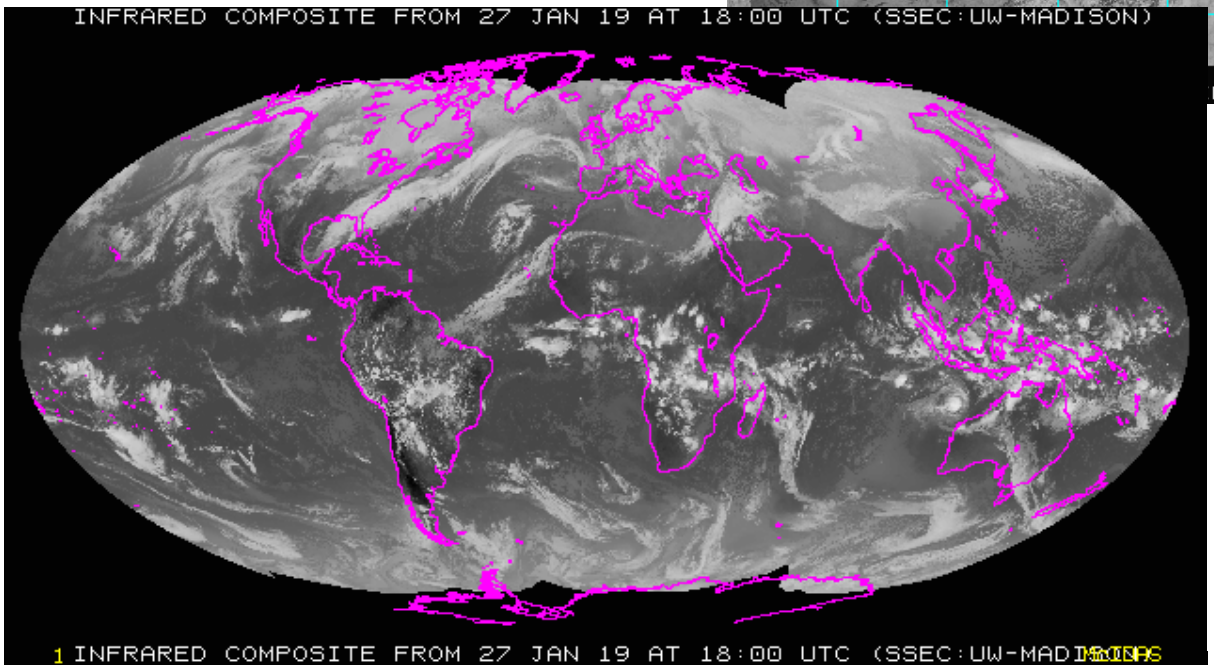
How it looks from 35800km above

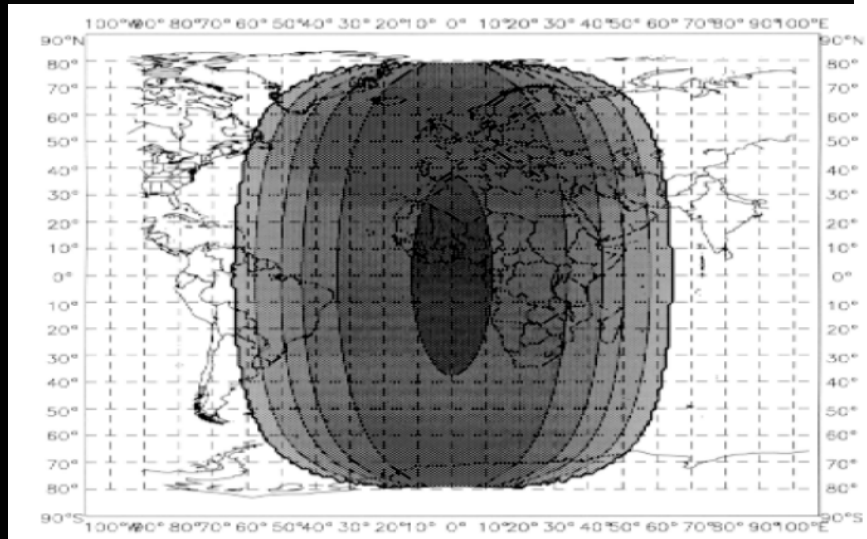
https://www.ssec.wisc.edu/data/geo/#/animation?satellite=met-prime

https://www.ssec.wisc.edu/data/composites/

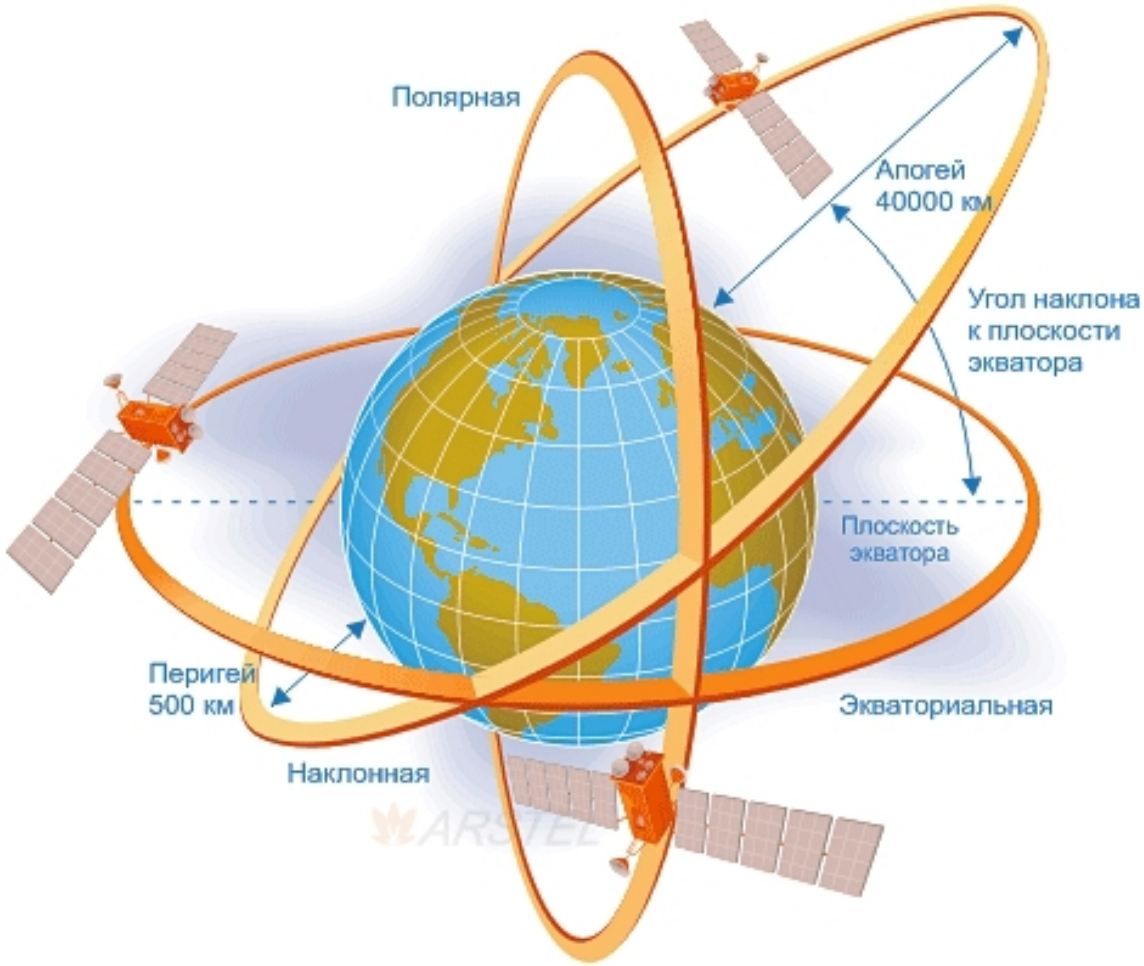




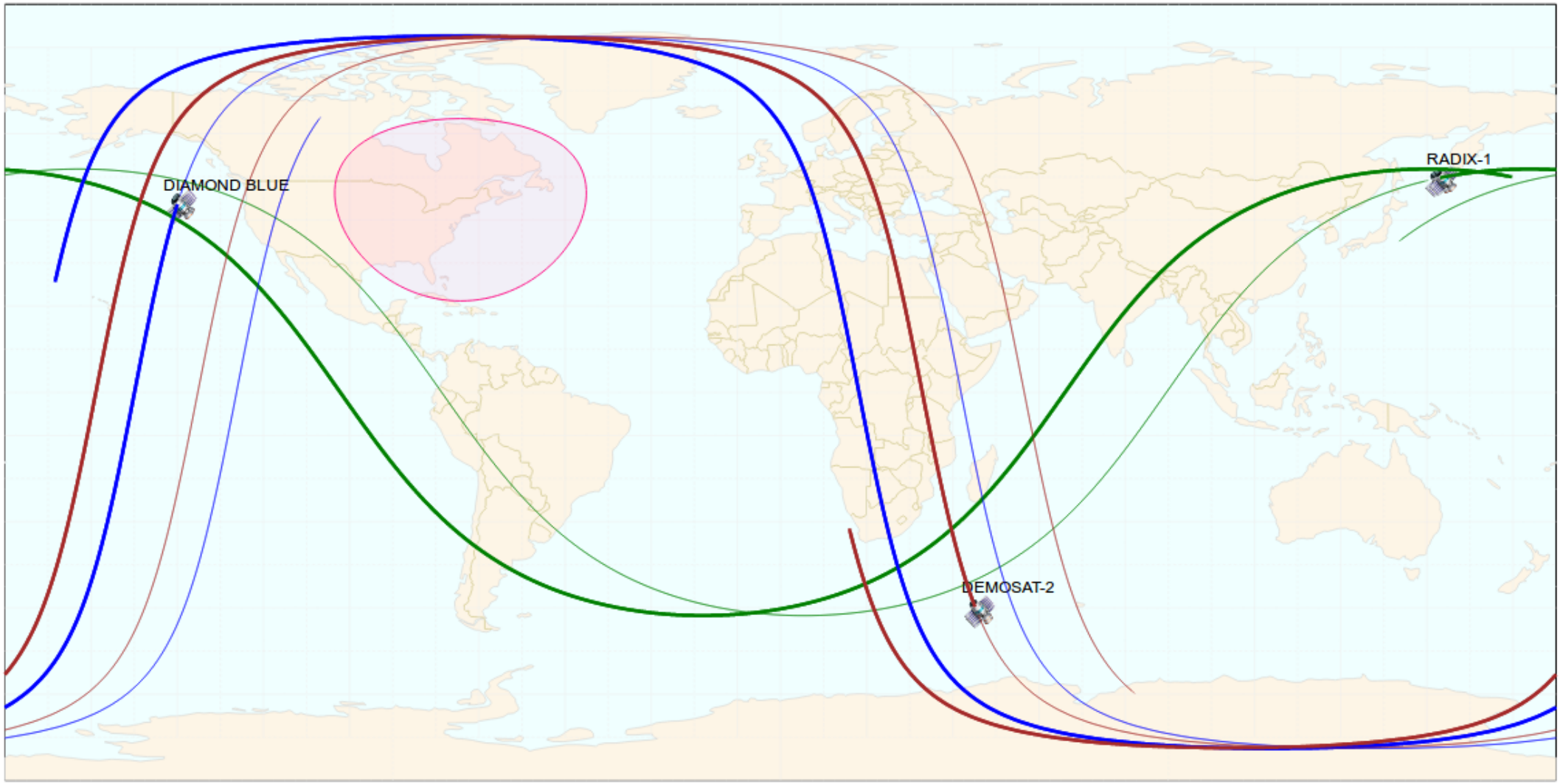




Orbit types. Inclination

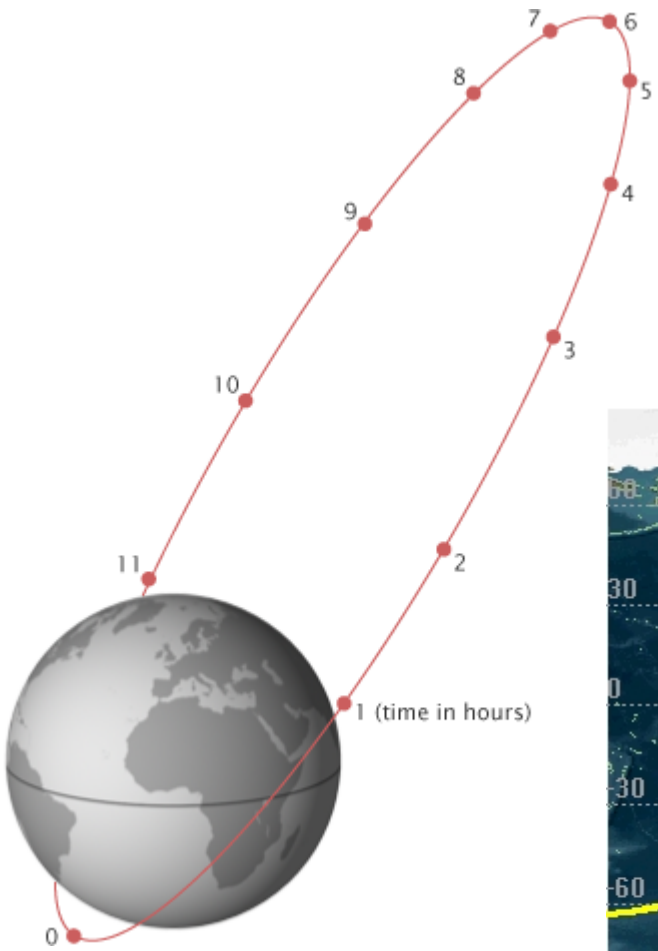


Orbit types. Inclination. Ground tracks



Molniya orbit

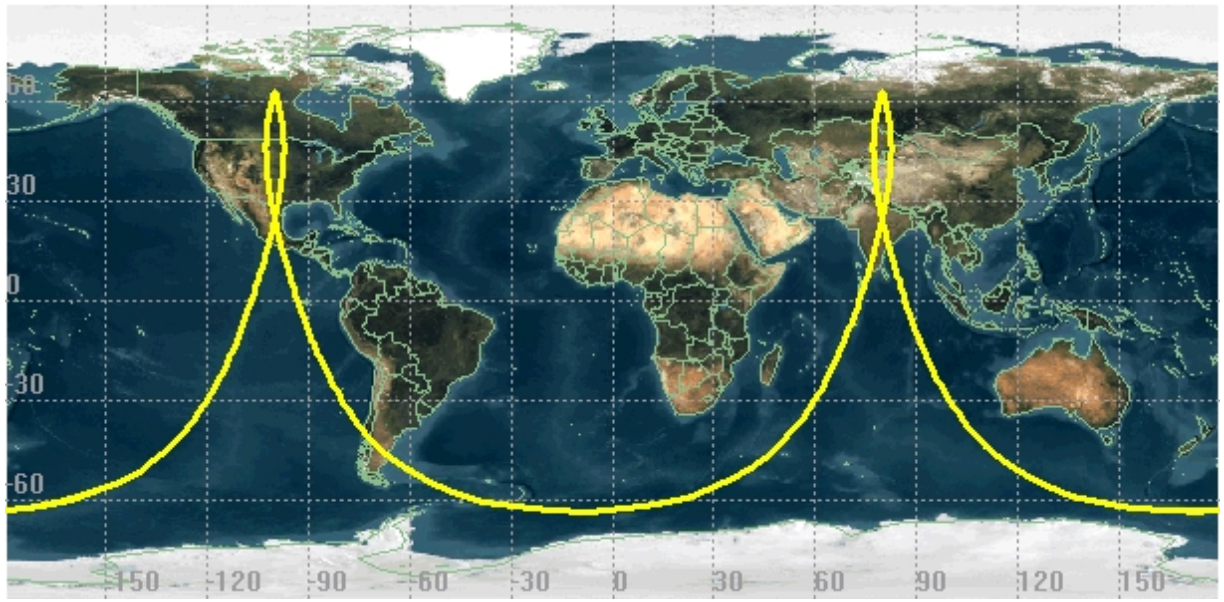
It is a highly elliptical orbit with an inclination of 63.4 degrees, an argument of perigee of 270 degrees and an orbital period of approximately half a sidereal day



1.

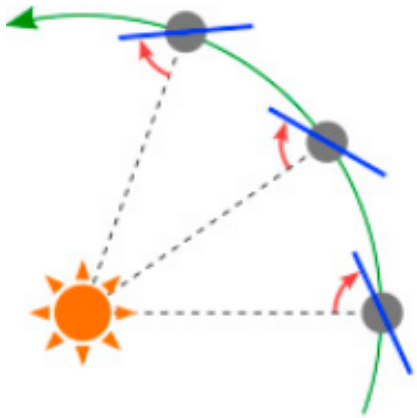
$$\Delta\omega = \Delta\omega_1 + \Delta\omega_2 = -2\pi \frac{J_2}{\mu\, p^2}\, 3\left(\frac{5}{4}\sin^2 i - 1\right)$$

Argument of perigee precession is close to 0 when i=63.4



Sun-synchronous orbit

It is an orbit arranged so that it precesses through one complete revolution each year, so it always maintains the same relationship with the Sun.



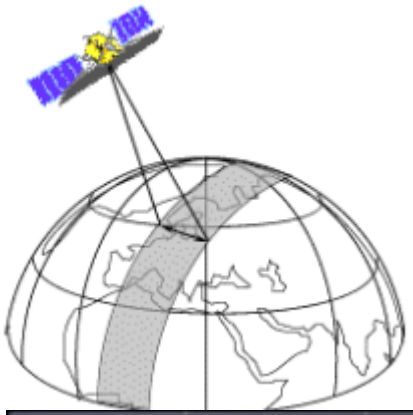
1.
$$\Delta\Omega = -2\pi \frac{J_2}{\mu\, p^2} \frac{3}{2} \cos i$$

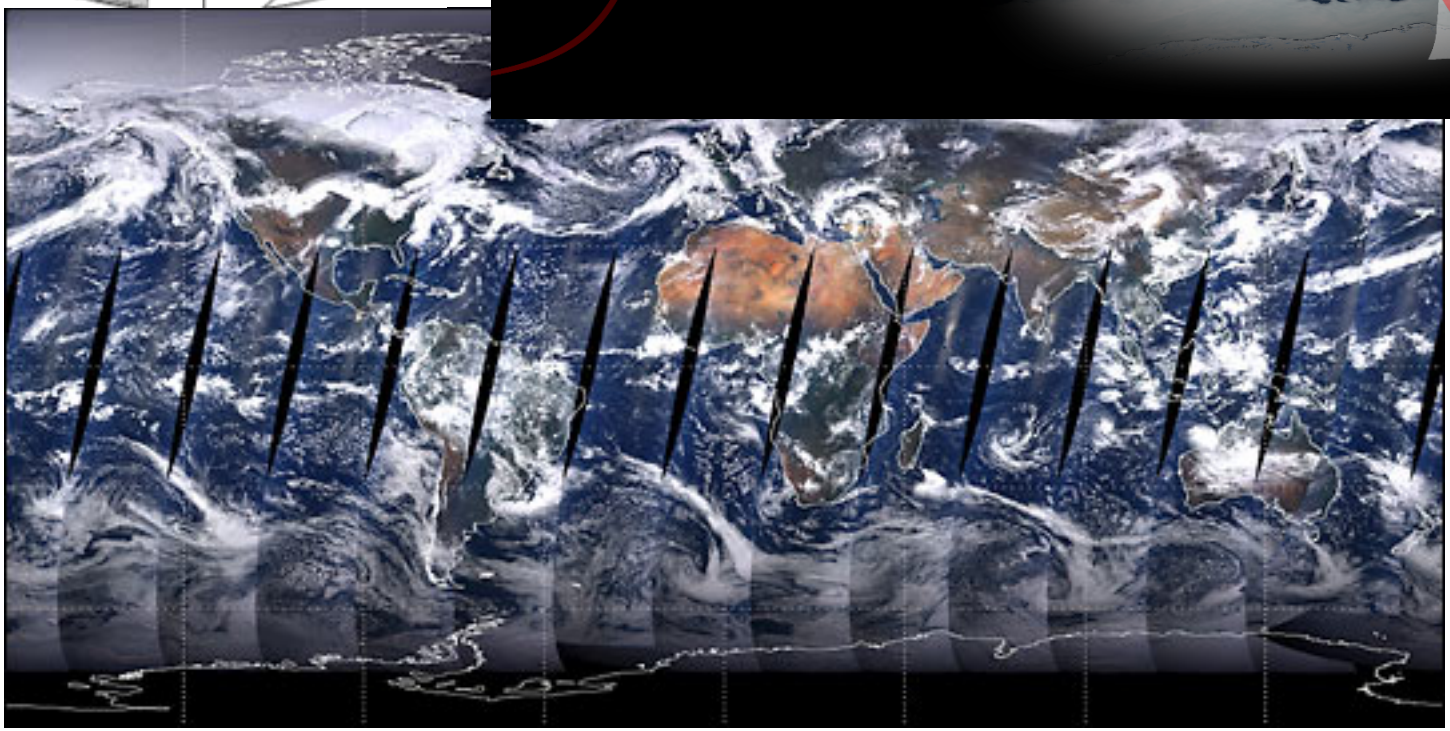
or when ρ is 360° per year,

$$\cos i \approx -\frac{\rho\sqrt{\mu}}{\frac{3}{2}J_2} a^{\frac{7}{2}} = -\left(\frac{a}{12\,352\ \mathrm{km}}\right)^{\frac{7}{2}} = -\left(\frac{P}{3.795\ \mathrm{h}}\right)^{\frac{7}{3}}.$$

Orbits per day	Period (h)	Height above Earth's surface (km)	Maximal latitude	Inclination
16	$1\frac{1}{2}$ = 1 h 30 min	282	83.4°	96.6°
15	$1\frac{3}{5}$ = 1 h 36 min	574	82.3°	97.7°
14	$1\frac{5}{7} \approx$ 1 h 43 min	901	81.0°	99.0°
13	$1\frac{11}{13} \approx$ 1 h 51 min	1269	79.3°	100.7°
12	2	1688	77.0°	103.0°

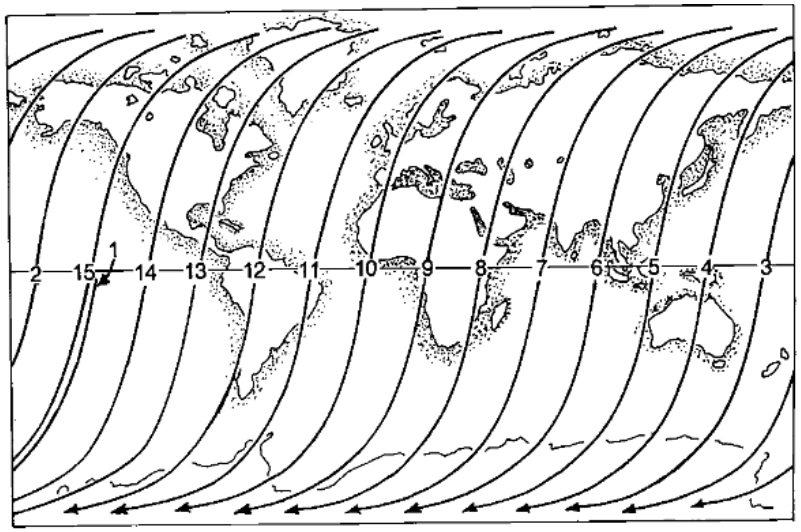
Image swath from sun-synchronous orbits



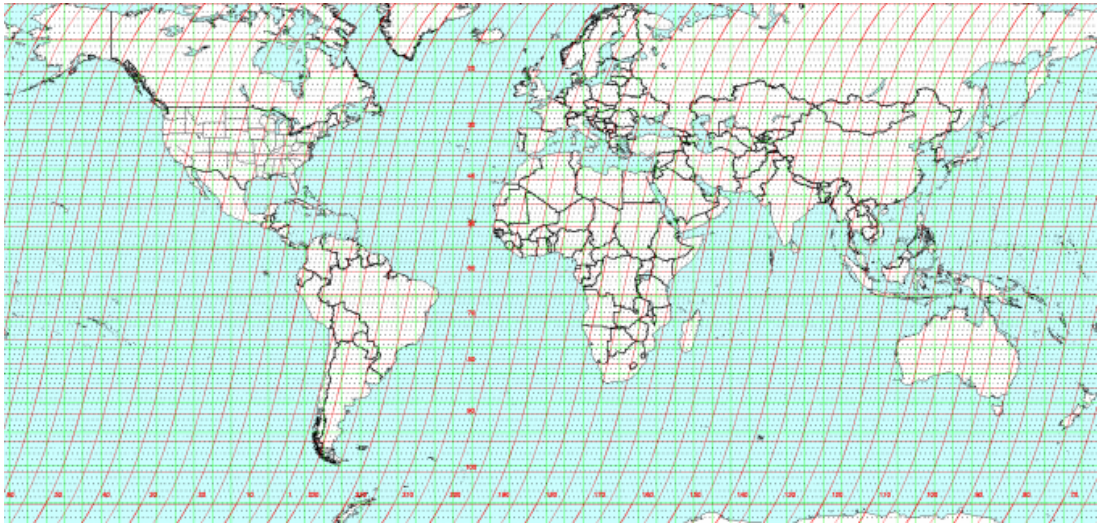


Ground tracks and path/row grid

1 day = 14.571 orbits



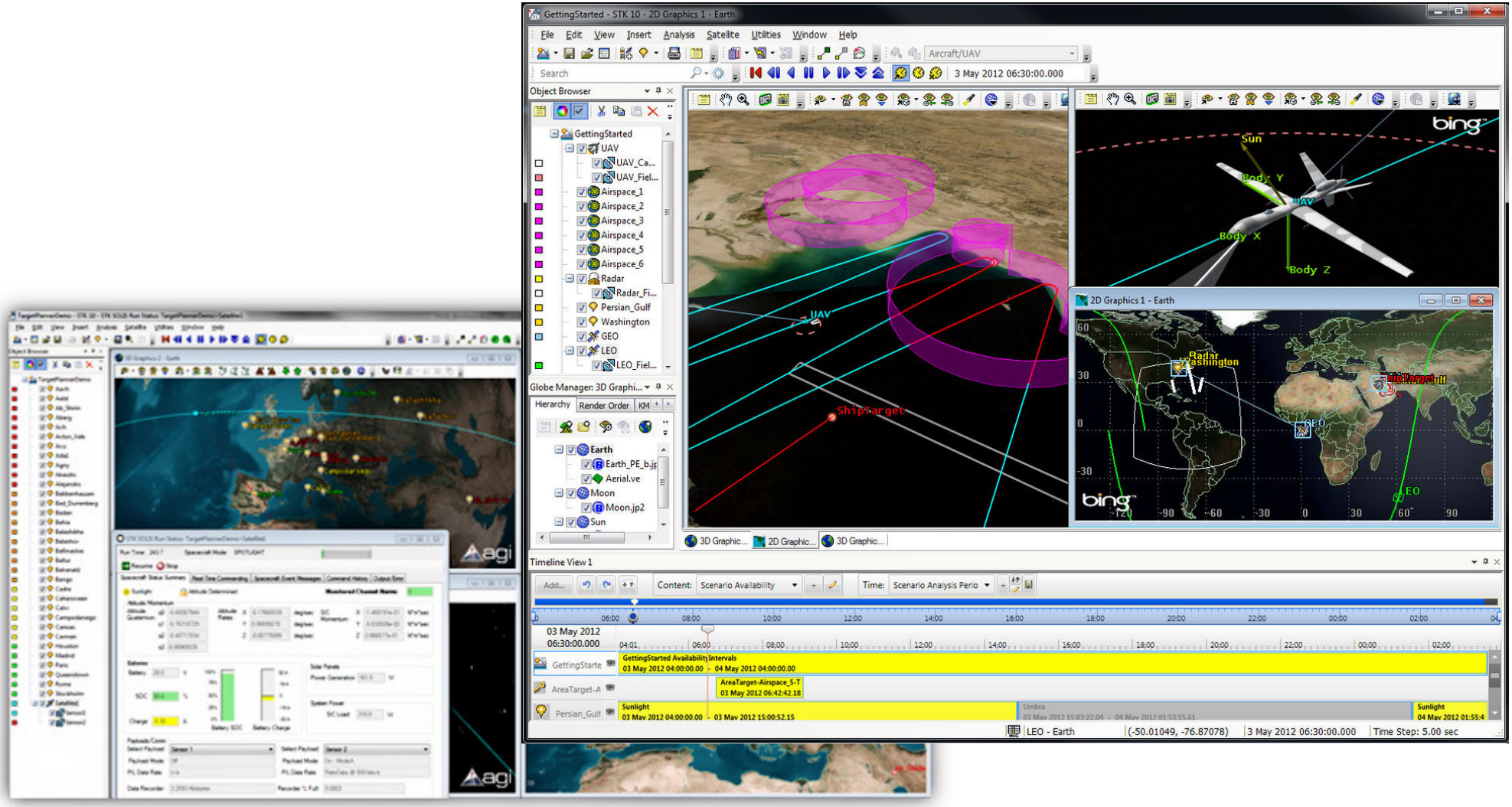
16 days = 233 orbits/pathes



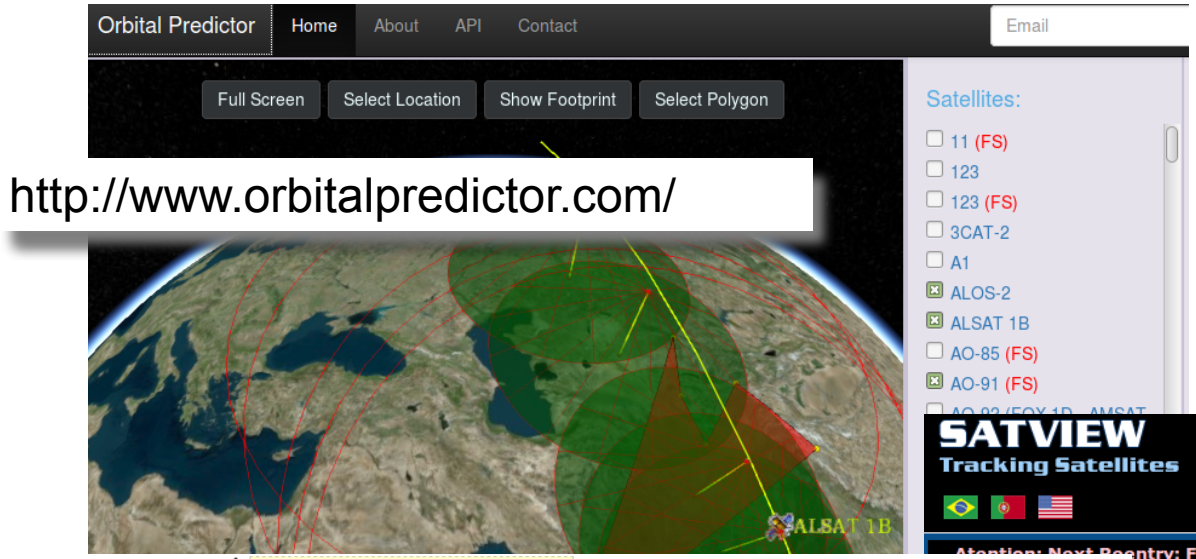
Main question: where is platform with sensor carrier at given moment ?

Use a proper program

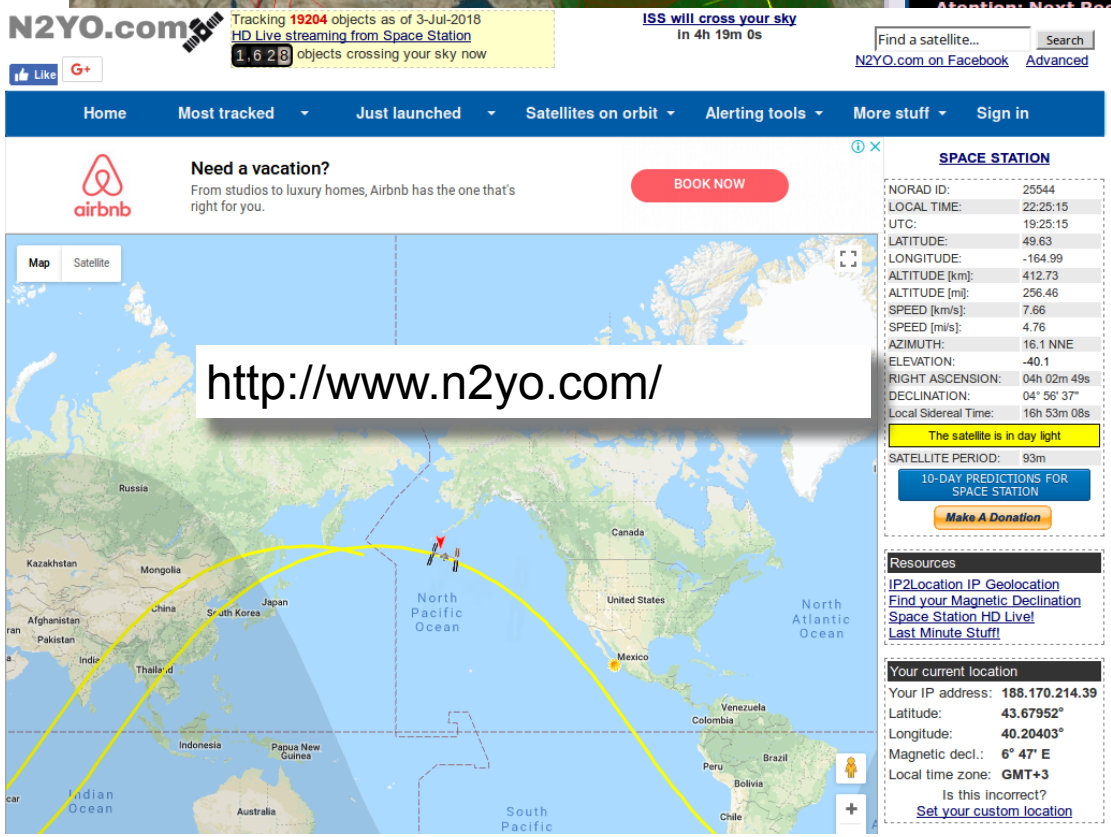
STK (Systems Tool Kit ,formerly Satellite Tool Kit) from Analytical Graphics, Inc.(AGI) http://www.agi.com/stk



Online tools



http://www.orbitalpredictor.com/



http://www.n2yo.com/

http://www.satview.org/

Orbital Libraries

- http://celestrak.com/publications/AIAA/2006-6753/ (C++, C#, FORTRAN, Java, MATLAB, Pascal)
- MATLAB, Orbital toolkit

- Python SGP4: https://pypi.org/project/sgp4/
- Python Pytroll/Pyorbital : https://github.com/pytroll/pyorbital
- Python PyEphem : http://rhodesmill.org/pyephem/

https://www.orekit.org

Most Visited | Linux Mint | YaMail | YANDEX | Slack | VISA | GenLib | Horizontal coordinat...

Orekit | Rugged | Stavor

About | Features | News | Developers | Community | Support

Orekit

An accurate and efficient core layer for space flight dynamics applications

Orekit 9.2 released, get it!

What is it?

Orekit, a low level space dynamics library written in Java, has gained widespread recognition since it was released under an open source license in 2008.

Low level space dynamics library

Orekit aims at providing accurate and efficient low level components for the development of flight dynamics applications. It is designed to be easily

Who is using it?

- Orekit has already been successfully used during the real time monitoring of the rendez-vous phase between the Automated Transfer Vehicle (ATV) and the International Space Station (ISS) by the Centre National d'Études Spatiales (CNES, the French space agency) and the European Space Agency (ESA).
- Orekit has been selected in early 2011 by CNES to be the basis of its next generation space flight dynamics systems, including operational

Who is behind it?

Development

Orekit has been in development since 2002 inside CS Systèmes d'Information and is still used and maintained by its space dynamics experts.

Several major actors of space research and industry are now involved into this project.

Open governance

Input Initial data for calculation

The United States Air Force tracks all detectable objects in Earth orbit, creating a corresponding TLE for each object, and makes available TLEs for non-classified objects on the website Space Track.[2] [3] The TLE format is a de facto standard for distribution of an Earth-orbiting object's orbital elements.

Specially designed to be used with SGP/SDP model

Public access via Celestrak site: https://www.celestrak.com/NORAD/elements/

Format description: https://ru.wikipedia.org/wiki/TLE

```
ISS (ZARYA)
1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11606-4 0  2927
2 25544  51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537
```

Two-lines elements. Line 1

```
ISS (ZARYA)
1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11606-4 0  2927
2 25544  51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537
```

Field	Columns	Content	Example
1	01–01	Line number	1
2	03–07	Satellite number	25544
3	08–08	Classification (U=Unclassified)	U
4	10–11	International Designator (Last two digits of launch year)	98
5	12–14	International Designator (Launch number of the year)	067
6	15–17	International Designator (piece of the launch)	A
7	19–20	Epoch Year (last two digits of year)	08
8	21–32	Epoch (day of the year and fractional portion of the day)	264.51782528
9	34–43	First Time Derivative of the Mean Motion divided by two [11]	−.00002182
10	45–52	Second Time Derivative of Mean Motion divided by six (decimal point assumed)	00000-0
11	54–61	BSTAR drag term (decimal point assumed) [11]	-11606-4
12	63–63	The number 0 (originally this should have been "Ephemeris type")	0
13	65–68	Element set number. Incremented when a new TLE is generated for this object.[11]	292
14	69–69	Checksum (modulo 10)	7

Two-lines elements. Line 2

```
ISS (ZARYA)
1 25544U 98067A   08264.51782528 -.00002182  00000-0 -11606-4 0  2927
2 25544  51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537
```

Field	Columns	Content	Example
1	01–01	Line number	2
2	03–07	Satellite number	25544
3	09–16	Inclination (degrees)	51.6416
4	18–25	Right ascension of the ascending node (degrees)	247.4627
5	27–33	Eccentricity (decimal point assumed)	0006703
6	35–42	Argument of perigee (degrees)	130.5360
7	44–51	Mean Anomaly (degrees)	325.0288
8	53–63	Mean Motion (revolutions per day)	15.72125391
9	64–68	Revolution number at epoch (revolutions)	56353
10	69–69	Checksum (modulo 10)	7

```python
from sgp4.earth_gravity import wgs72
from sgp4.io import twoline2rv

line1 = ('1 00005U 58002B   00179.78495062  .00000023  00000-0  28098-4 0  4753')
line2 = ('2 00005  34.2682 348.7242 1859667 331.7664  19.3264 10.82419157413667')

satellite = twoline2rv(line1, line2, wgs72)
position, velocity = satellite.propagate( 2000, 6, 29, 12, 50, 19)
print(position)
print(velocity)
```

```python
from datetime import datetime, timedelta
from pyorbital import orbital

sat = orbital.Orbital("ISS (ZARYA)",
    line1="1 43798U 18099AS  18346.44873607 -.00000205  00000-0 -13222-4 0  9992",
    line2="2 43798  97.7673  55.5275 0011668 217.0742 142.9674 14.95653258  1288")
d = datetime(2019, 1, 23, 0, 3, 22)
lon, lat, alt = sat.get_lonlatalt(d)
position,velocity = sat.get_position(d,normalize=False)
print(position)
print(velocity)
```

Terminal

File Edit View Search Terminal Help

```
[  -33.01180935  5394.27440504  4380.9923147 ]
[ 1.34803019 -4.70791839  5.78301925]
```

Going to practice. Do it yourself….