In [2]:
```
import numpy as np
```

https://ru.wikipedia.org/wiki/TLE (https://ru.wikipedia.org/wiki/TLE)

http://celestrak.com/NORAD/elements/ (http://celestrak.com/NORAD/elements/)

https://github.com/pytroll/pyorbital/blob/master/pyorbital/tests/test_orbital.py (https://github.com/pytroll/pyorbital/blob/master/pyorbital/tests/test_orbital.py)

http://licensing.agi.com/stk/ (http://licensing.agi.com/stk/)

Part 1

1.1) Estimate maximum spatial error in position on Earth surface if UTC is used instead of UT1

In [3]:
```
# 15 degree / 60 min
# 1 deg / 4 min
# 0.25 deg / 1 min
# 0.00416666667 / 1 sec
angle = 0.00416666667
h = 35800
R = 6400
error = np.pi * R * angle / 180
print("error in km", "%.1f" % error)
```

```
error in km 0.5
```

1.2) Calculate period of a satellite X1 on near circular orbit with altitude 500км

In [4]:
```
earth_period = 1
G = 6.674*10**(-11)  # N m² kg⁻²
M = 5.972*10**24  # kg
earth_semi_major = 6400
# sat_radius = earth_semi_major + 500
sat_radius = earth_semi_major + 370
sat_radius
```

Out[4]: 6770

In [5]:
```
orbit_length = 2*np.pi*sat_radius
print("orbit_length, km", "{0:.1f}".format(orbit_length), "%.1f" % orbit_length)
```

```
orbit_length, km 42537.2 42537.2
```

$$v = \sqrt{\frac{GM}{r}}$$

In [6]:
```
sat_speed = np.sqrt(G*M/(sat_radius*1000))
print("sat_speed, m/sec", "%.1f" % sat_speed)
```

```
sat_speed, m/sec 7672.9
```

$$T = 2\pi \sqrt{\frac{r^3}{GM}}$$

In [7]:
```
orbit_length = 1000*orbit_length  # meters
period = ((orbit_length)/sat_speed)  # sec
sat_period = period / 60
print("orbit_period, min", "%.1f" % sat_period)
```

```
orbit_period, min 92.4
```

In [8]:
```
# sat_period = 2*np.pi*np.sqrt(sat_radius**3/(G*M))
# sat_period
```

1.3) Calculate mean velocity of the satellite X1 and mean velocity of its subsatellite point

In [9]:
```
sat_speed = np.sqrt(G*M/(sat_radius*1000))
print("sat_speed, m/sec", "%.1f" % sat_speed)
```

```
sat_speed, m/sec 7672.9
```

In [10]:
```
# time is equal
subsat_point_length = 2*np.pi*earth_semi_major
print("subsat_point_length, km", "%.1f" % subsat_point_length)
```

```
subsat_point_length, km 40212.4
```

In [11]:
```
subsat_point_speed = (1000*subsat_point_length) / (60*sat_period)  # km / min
print("subsat_point_speed, m/sec", "%.1f" % subsat_point_speed)
```

```
subsat_point_speed, m/sec 7253.5
```

1.4) Calculate longitudinal distance (km) between two successive ground tracks for X1. a) at equator b) at 55d. latitude

difference between tracks

prbly it's the distance earth rotated in that period

In [12]:
```
# 15 degree / 60 min
# 1 deg / 4 min
# 0.25 deg / 1 min
# 0.00416666667 / 1 sec
angle = 0.00416666667
h = 35800
R = 6400
dist_btw_two_ground_tracks = np.pi * R * (60*angle)*sat_period / 180
print("longitudinal distance at equator in km", "%.1f" % dist_btw_two_ground_tracks)
```

```
longitudinal distance at equator in km 2580.2
```

In [ ]:

1.5) How many imaging GEO satellites are required to cover Earth without gaps in equatorial belt ?

how many arcs

In [13]:
```
GEO_h = 42164
GEO = 35786
```

In [14]:
```
#tangent_line_x = np.sqrt(GEO_h**2 - R**2)
alpha = np.arccos(R/GEO_h)
print("alpha in rad", "%.1f" % alpha)
```

```
alpha in rad 1.4
```

In [15]:
```
arc_length = R*(2*alpha)
print("arc_length, km", "%.1f" % arc_length)
```

```
arc_length, km 18155.8
```

In [16]:
```
arcs_in_equator = 2*np.pi*R/arc_length
print("arcs_in_equator", "%.1f" % arcs_in_equator)
print("need three satellites on GEO orbit")
```

```
arcs_in_equator 2.2
need three satellites on GEO orbit
```

Part 2

For next problems select one satellite from a list: AQUA,TERRA, RADARSAT-2.

Fetch and use a fresh TLE from www.celestrak.com

Ground station and imaging target are located at
55N.latitude, 37 E.longitude unless specified otherwise

Reference or start of scenario time is
2019-01-30T00:00:00UTC

https://github.com/pytroll/pyorbital/blob/master/pyorbital/orbital.py (https://github.com/pytroll/pyorbital/blob/master/pyorbital/orbital.py)

https://github.com/pytroll/pyorbital/blob/master/pyorbital/tests/test_orbital.py
(https://github.com/pytroll/pyorbital/blob/master/pyorbital/tests/test_orbital.py)

http://celestrak.com/NORAD/elements/ (http://celestrak.com/NORAD/elements/)

https://docs.google.com/document/d/1_qHbpU1Rq9xvh7Y_159CCZthJZOem5NN2awYLQd5hko/edit
(https://docs.google.com/document/d/1_qHbpU1Rq9xvh7Y_159CCZthJZOem5NN2awYLQd5hko/edit)

2.1) Calculate satellite position at reference time (Cartesian, lat-lon-alt , topocentric(az,el,range)

In [106]:
```
from pyorbital.orbital import Orbital
from datetime import datetime, timedelta
import time

import matplotlib.pyplot as plt
import matplotlib


from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.graph_objs as go

init_notebook_mode(connected=True)
```

In [18]:
```
dt = datetime(2019, 1, 30, 0, 0, 0)
dt
```

Out[18]:
```
datetime.datetime(2019, 1, 30, 0, 0)
```

In [19]:
```
line1 = '1 32382U 07061A   19047.12869166  .00000214  00000-0  10000-3 0  9993'
line2 = '2 32382  98.5773  55.8244 0001171  82.6262 350.4330 14.29984888583323'
```

In [20]:
```
orb = Orbital("RADARSAT-2",
             line1 = line1, line2 = line2)
```

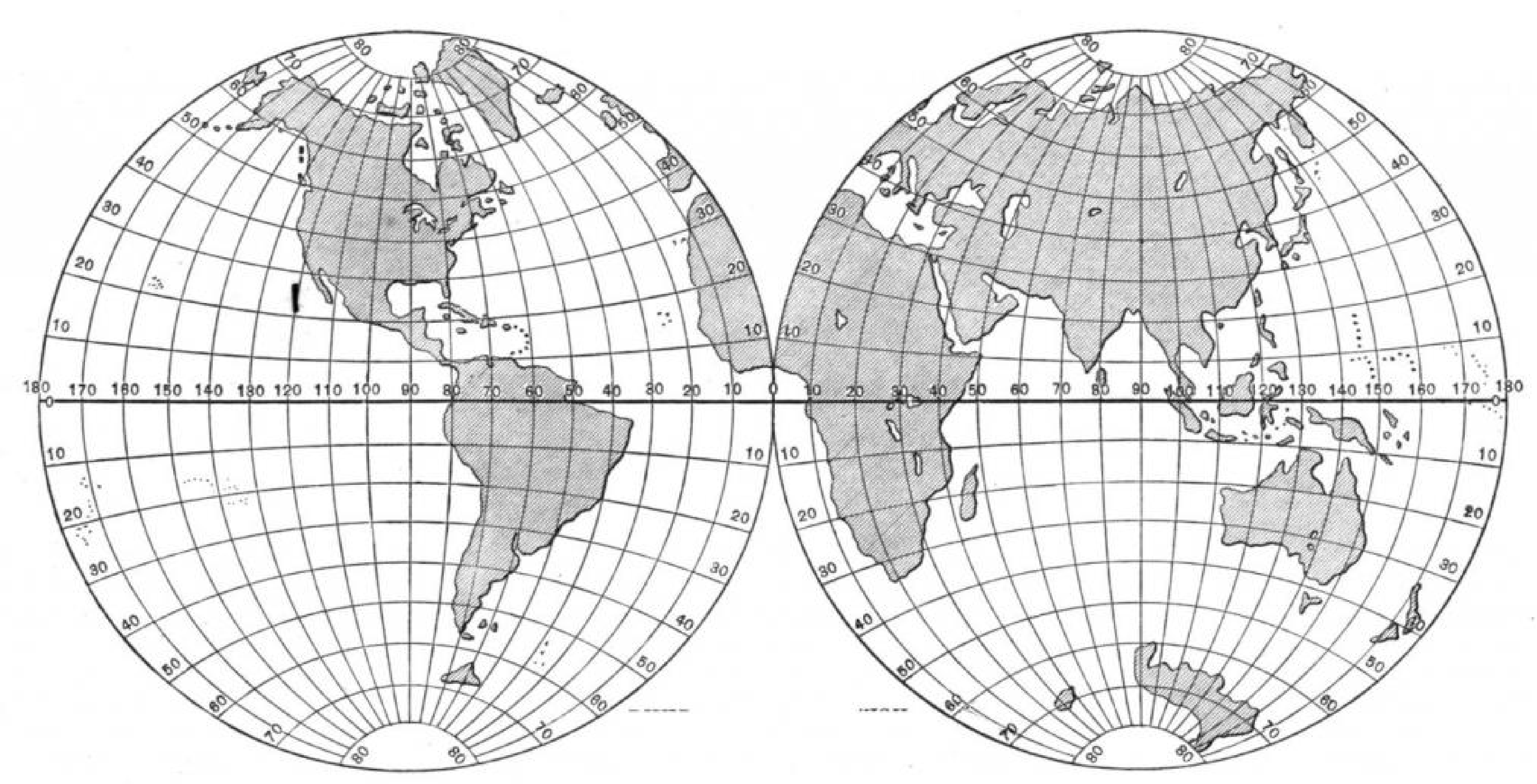
In [21]:
```
# orb = Orbital("AQUA",
#              line1='1 27424U 02022A   19039.92419981  .00000070  00000-0  25584-4 0  9995',
#              line2='2 27424  98.2216 342.6641 0002877  76.0819  15.6033 14.57108406891855')
```

In [22]:
```
# utc_time = '2019-01-30 00:00:00'
cart = orb.get_position(dt, normalize=False)
print('cartesian position and velocity are: ')
print("pos: ", cart[0])
print("vel: ", cart[1])
```

```
cartesian position and velocity are:
pos:  [-4257.61638408 -4206.73432558  3942.36847584]
vel:  [-3.81362025 -1.89573259 -6.12418654]
```

In [23]:
```
lon, lat, alt = orb.get_lonlatalt(dt)
print('lon, lat, alt are:', lon, lat, alt)
```

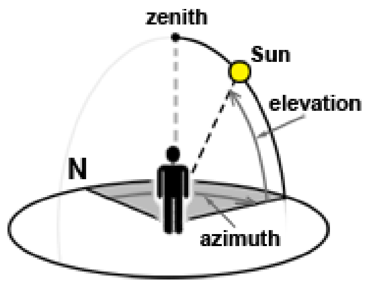
```
lon, lat, alt are: 95.71127549023225 33.529200848155064 795.3754769591919
```



In [46]:
```
# Moscow
lon_gs, lat_gs, alt_gs = 55., 37., 0  # ground station
topocentric = orb.get_observer_look(dt, lon_gs, lat_gs, alt_gs)
print('Azimuth is, degree: ', "%.2f" % topocentric[0])
print('Elevation is, degree: ', "%.2f" % topocentric[1])
```

```
Azimuth is, degree:  83.59
Elevation is, degree:  -5.45
```

In [25]:
```
# sat_range =
```



2.2) Calculate AOS/LOS events for first contact with the ground station (min. elevation level 10deg)

and

2.3) Calculate first pass culmination event time and culmination elevation.

AOS stands for Acquisition of Signal (or Satellite). AOS is the time that a satellite rises above the horizon of an observer.

TCA stands for Time of Closest Approach. This is the time when the satellite is closest to the observer and when Doppler shift is zero. This usually corresponds to the time that the satellite reaches maximum elevation above the horizon.

LOS stands for Loss of Signal (or Satellite). LOS is the time that a satellite passes below the observer's horizon.

Ground station and imaging target are located at 55N.latitude, 37 E.longitude unless specified otherwise

this is Moscow

In [35]:
```
dt_gs = datetime(2019, 1, 30, 0, 0, 0)  # ground station
print(dt, 10, lon_gs, lat_gs, alt_gs)
```

```
2019-01-30 00:00:00 10 55.0 37.0 0
```

In [47]:
```
"""Calculate passes for the next hours for a given start time and a
given observer.
Original by Martin.
utc_time: Observation time (datetime object)
length: Number of hours to find passes (int)
lon: Longitude of observer position on ground (float)
lat: Latitude of observer position on ground (float)
alt: Altitude above sea-level (geoid) of observer position on ground (float)
tol: precision of the result in seconds
horizon: the elevation of horizon to compute risetime and falltime.
Return: [(rise-time, fall-time, max-elevation-time), ...]
"""
hours = 2
horizon = 15
tol = 0.001

next_passes = orb.get_next_passes(dt, hours, lon_gs, lat_gs, alt_gs, tol, horizon)
print("rise-time (AOS): ", next_passes[0][0])
print("fall-time (LOS): ", next_passes[0][1])
print("max-elevation-time: ", next_passes[0][2])
```

```
rise-time (AOS):  2019-01-30 01:37:49.642676
fall-time (LOS):  2019-01-30 01:42:51.470050
max-elevation-time:  2019-01-30 01:40:20.896087
```

In [45]:
```
dt23 = datetime(2019, 1, 30, 1, 42, 51, 470050)
topocentric23 = orb.get_observer_look(dt23, lon_gs, lat_gs, alt_gs)
print('Culmination elevation is, degree: ', "%.2f" % topocentric[1])
```

```
Culmination elevation is, degree:  -5.45
```

roll alngle on sat is an accident angle

Seminar

Presentation https://github.com/Geoalert/presentation/tree/master/docs (https://github.com/Geoalert/presentation/tree/master/docs)

https://platform.digitalglobe.com/gbdx/ (https://platform.digitalglobe.com/gbdx/)

google earth engine

https://demo.aeronetlab.space/dashboard (https://demo.aeronetlab.space/dashboard)

https://search.kosmosnimki.ru (https://search.kosmosnimki.ru)

https://discover.digitalglobe.com (https://discover.digitalglobe.com)

https://www.planet.com/explorer/ (https://www.planet.com/explorer/)

https://scihub.copernicus.eu/dhus/#/home (https://scihub.copernicus.eu/dhus/#/home) https://scihub.copernicus.eu (https://scihub.copernicus.eu)

http://step.esa.int/main/ (http://step.esa.int/main/)

https://qgis.org/ru/site/forusers/download.html (https://qgis.org/ru/site/forusers/download.html)

Install virtual env and another version of python

https://stackoverflow.com/questions/41535881/how-do-i-upgrade-to-python-3-6-with-conda (https://stackoverflow.com/questions/41535881/how-do-i-upgrade-to-python-3-6-with-conda)

https://stackoverflow.com/questions/5506110/is-it-possible-to-install-another-version-of-python-to-virtualenv (https://stackoverflow.com/questions/5506110/is-it-possible-to-install-another-version-of-python-to-virtualenv)

https://stackoverflow.com/questions/23842713/using-python-3-in-virtualenv (https://stackoverflow.com/questions/23842713/using-python-3-in-virtualenv)

2.4) Calculate first 5 imaging events schedule for the target. Imaging incidence angle < 30deg.

In [54]:
```
# +- 30 degr from nadir
hours = 180
horizon = 60
tol = 0.001

next_passes = orb.get_next_passes(dt, hours, lon_gs, lat_gs, alt_gs, tol, horizon)
for i in next_passes:
    print(i[2])
```

```
2019-01-31 02:51:04.857641
2019-01-31 14:06:52.073310
2019-02-01 13:37:52.905574
2019-02-04 02:34:31.417543
2019-02-04 13:50:18.401248
```

2.6) * Select one of provided TLE files with one year coverage and investigate increase of propagation error vs epoch age. Present results in either tabular or (better) graphical form (error vs te age)

In [55]:
```
with open('terra-2018.tle') as f:
    lines = f.readlines()
```

In [63]:
```
lines[0]
```

Out[63]:
```
'TERRA\n'
```

In [60]:
```
lines[1]
```

Out[60]:
```
'1 25994U 99068A   18001.11047143  .00000139  00000-0  40989-4 0  9999\n'
```

In [61]:
```
lines[2]
```

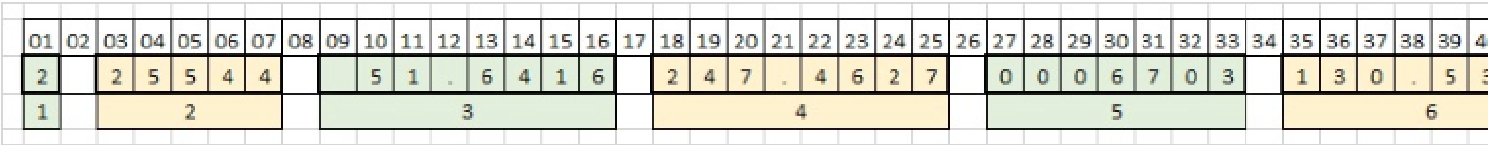
Out[61]:
```
'2 25994  98.2120  78.3294 0000706  73.9346 286.1946 14.57110151959499\n'
```

Строка 1 (обязательная)

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1		2	5	5	4	4	U		9	8	0	6	7	A				0	8	2	6	4	.	5	1	7	8	2	5	2	8		-	.	0	0	0	0	2
1		2					3		4		5			6				7		8													9						

Номер	Положение	Содержание	Пример
1	01-01	Номер строки	1
2	03-07	Номер спутника в базе данных NORAD	25544
3	08-08	Классификация (U=**U**nclassified — не секретный)	U
4	10-11	Международное обозначение (последние две цифры года запуска)	98
5	12-14	Международное обозначение (номер запуска в этом году)	067
6	15-17	Международное обозначение (часть запуска)	A
7	19-20	Год эпохи (последние две цифры)	08
8	21-32	Время эпохи (целая часть — номер дня в году, дробная — часть дня)	264.51782528
9	34-43	Первая производная от среднего движения (ускорение), делённая на два [виток/день^2]	-.00002182
10	45-52	Вторая производная от среднего движения, делённая на шесть (подразумевается, что число начинается с десятичного разделителя) [виток/день^3]	00000-0
11	54-61	Коэффициент торможения B* (подразумевается, что число начинается с десятичного разделителя)	-11606-4
12	63-63	Изначально — типы эфемерид, сейчас — всегда число 0	0
13	65-68	Номер (версия) элемента	292
14	69-69	Контрольная сумма по модулю 10	7

Строка 2 (обязательная)



Номер	Положение	Содержание	Пример
1	01-01	Номер строки	2
2	03-07	Номер спутника в базе данных NORAD	25544
3	09-16	Наклонение в градусах	51.6416
4	18-25	Долгота восходящего узла в градусах	247.4627
5	27-33	Эксцентриситет (подразумевается, что число начинается с десятичного разделителя)	0006703
6	35-42	Аргумент перицентра в градусах	130.5360
7	44-51	Средняя аномалия в градусах	325.0288
8	53-63	Частота обращения (оборотов в день) (среднее движение) [виток/день]	15.72125391
9	64-68	Номер витка на момент эпохи	56353
10	69-69	Контрольная сумма по модулю 10	7

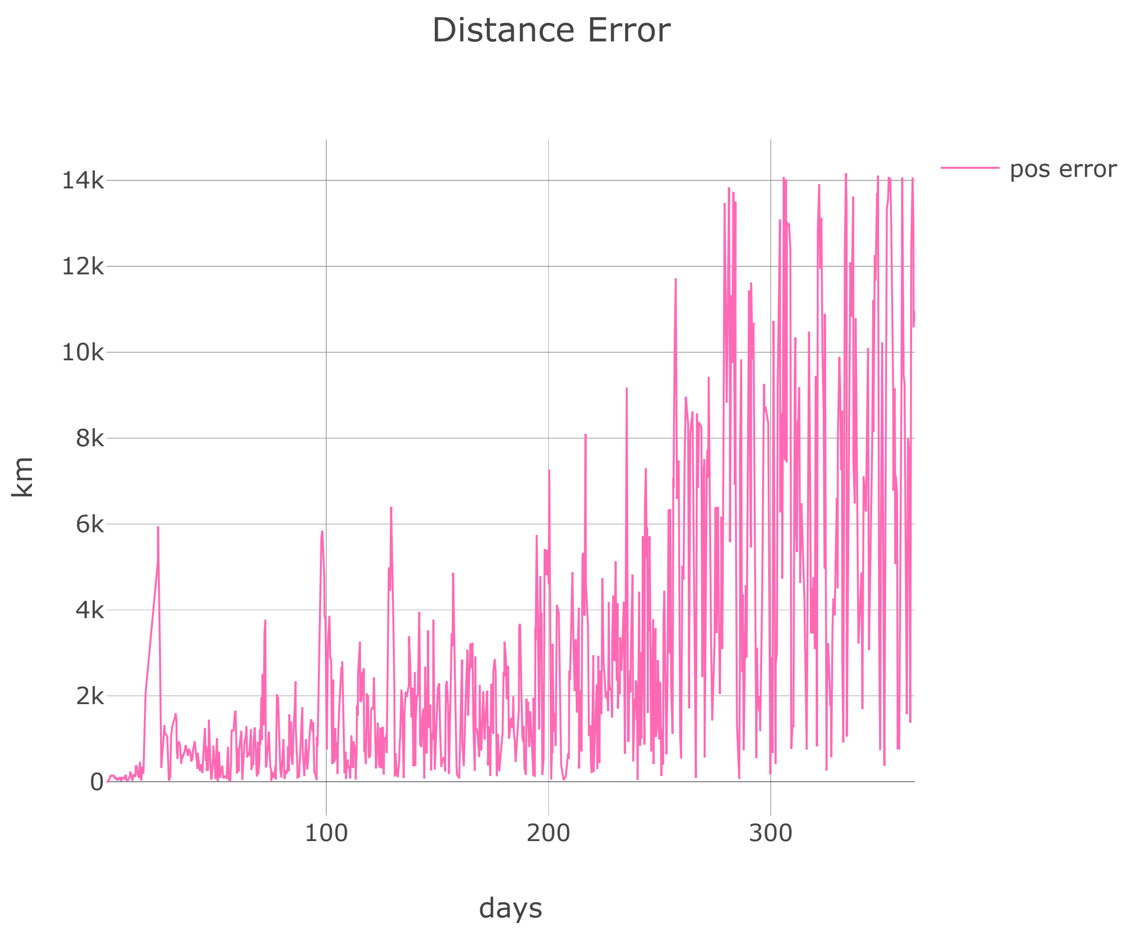
In [116]:
```
# lines
checkmarks = int(len(lines)/3)
print(checkmarks)
```

```
995
```

In [115]:
```
dt6 = datetime(2019, 1, 30, 0, 0, 0)
# utc_time = '2019-01-30 00:00:00'
```

In [114]:
```
sat_x = np.zeros(checkmarks)
sat_y = np.zeros(checkmarks)
sat_z = np.zeros(checkmarks)

days = np.zeros(checkmarks)

for i in range(checkmarks):

    line1 = lines[3*i + 1]
    line2 = lines[3*i + 2]

    sat = Orbital("TERRA",
             line1 = line1, line2 = line2)

    day = line1.split()[3]
    day = float(day[2:])  # number of day and part of day
    days[i] = day

    cart_pos, cart_angle = sat.get_position(dt6, normalize = False)
    sat_x[i] = cart_pos[0]
    sat_y[i] = cart_pos[1]
    sat_z[i] = cart_pos[2]

"""
using one reference time calculate prediction for sat pos
substract prediction from true data to get error

"""

distance = np.sqrt((sat_x - sat_x[-1])**2 +
                   (sat_y - sat_y[-1])**2 +
                   (sat_z - sat_z[-1])**2)
days_inv = days[::-1]
```

In [117]:
```
data = [
    go.Scatter(
        x=days_inv,
        y=distance,
        mode = 'lines',
        line = dict(
            color = ('rgb(255,105,180)'),
            width = 1,
        ),
        name='pos error'
    ),
]

layout= go.Layout(
    title= 'Distance Error',
    xaxis= dict(
        title= 'days',
    ),
    yaxis=dict(
        title= 'km',
        scaleratio=1,
    ),
    showlegend= True
)
fig= go.Figure(data=data, layout=layout)
iplot(fig)
```



Export to plot.ly »