TIDAL VARIABILITY IN IONOSPHERE

By:ASHISH KUMAR SETH(CSE/15011/066)

and

ABHAS KUMAR(CSE/15001/056)

Bachelor Thesis submitted to

Indian Institute of Information Technology Kalyani

for the partial fulfillment of the degree of

Bachelor of Technology

In

Computer Science and Engineering ,

May, 2018

Certificate

This is to certify that the thesis entitled “Tidal Variability in Ionosphere” being submitted by Ashish Kumar Seth and Abhas Kumar, undergraduate students, Reg.No.0000066, Roll No.15011,Reg.No.0000056, Roll No.15001 respectively, in the Department of Computer Science and Engineering , Indian Institute of Information Technology Kalyani, West Bengal 741235, India, for the award of Bachelors of Technology in Computer Science and Engineering is an original research work carried by them under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of Indian Institute of Information Technology Kalyani and in my opinion, has reached the standards needed for submission. The work, techniques and the results presented have not been submitted to any other University or Institute for the award of any other degree or diploma.

Dr.Uma Das

Position:Assistant Professor

Departmental address:IIIT Kalyani

Declaration

We hereby declare that the work being presented in this thesis entitled, "Tidal Variability in Ionosphere", submitted to Indian Institute of Information Technology Kalyani in partial fulfillment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering during the period from July, 2017 to May, 2018 under the supervision of **Dr.Uma Das**, Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India, does not contain any classified information.

Ashish Kumar Seth(CSE/15066/066)

Abhas Kumar(CSE/15001/056)

Computer Science and Engineering

Indian Institute of Information Technology

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

.....

Dr.Uma Das

Position:Assistant Professor

Departmental address:IIIT Kalyani

Place:Kalyani,West Bengal

Date:13 April 2018

Acknowledgments

First of all, I would like to take this opportunity to thank my supervisor Dr.Uma Das.

Ashish Kumar Seth(CSE/15066/066)

Abhas Kumar(CSE/15001/056)

Computer Science and Engineering

Indian Institute of Information Technology,kalyani

Place:Kalyani,West Bengal

Date:13 April 2018

List of Tables

List of Figures

Chapters

Page

ABSTRACT

The Ionosphere , ionized part of the earth's upper atmosphere, atoms of this area have been stripped of into electrons and ions because of the high energy of the sun and cosmic rays, so it is certain that the molecules those come in the way of sun's rays, would be ionized,more ionization will be occurred mainly in day time, and in the night time no. of moving electron would be less. Somehow one can clearly see that there is variation in electron distribution in ionosphere.and this distribution represented in terms of electron density,by seeing the falling and raising of electron density , the term defines this variation as tides(also called atmospheric tides). This project "Tidal Variability in Ionosphere" is all about investigating those variation which are actually happening in ionosphere.In order to investigate those thing we have worked on 1 year of Data which is measured by COSMIC satellite and encodes these data into nc(NetCdf) files. This satellites capture lots of data like list of longitude value , time , latitude,altitude ,electron density and so on. We extract those data from the nc files by applying some constraints and make it uniform by using some python modules and machine learning concept and try to plot the contour so that one can observe the variation of electron density value in ionosphere. Once we have recorded all the observation, can utilize these information in radio communication application.

CHAPTER 1 : Introduction

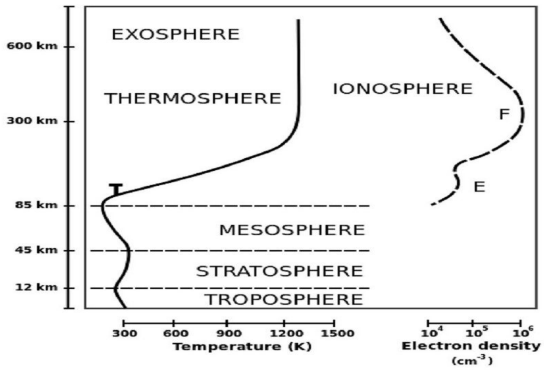
1.1 About Ionosphere,Longitude,Latitude,Electron density.

Ionosphere

Ionosphere is a region of Earth's upper atmosphere from about 60 km to 1000 km altitude that includes the thermosphere and parts of the mesosphere and exosphere,it is ionised by solar radiation that plays an important part in atmospheric electricity.

Atmospheric electricity is the study of electron charges in the Earth's atmosphere.

*The ionosphere is a shell of electrons and electrically charged atoms and molecules that surrounds the Earth,primarily due to ultraviolet radiation from the sun.



Longitude

Longitude, is a geographic coordinate that specifies the east-west position of a point on the Earth's surface. It is an angular measurement, usually expressed in degrees.Ranges from 0°to 360°

Latitude

In geography, latitude is a geographic coordinate that specifies the north–south position of a point on the Earth's surface. Latitude is an angle which ranges from 0° at the Equator to 90° at the poles.

Electron Density

Electron density is one of the key parameters in physics of a gas discharge.

Analysing Electron Density of Ionosphere can help in better understanding on atmospheric electricity , Radio communication and can be very useful in such as.Radio application(communications or GNSS augmentation) as well as ionospheric sciences.High Frequency radio wave propagation prediction

1.2 Atmospheric Tides

Atmospheric tides are global-scale periodic oscillations of the atmosphere.In many ways they are analogous to ocean tides.

Atmospheric tides can be excited by:

a) The regular day–night cycle in the Sun's heating of the atmosphere(insolation)

b) The gravitational field pull of the Moon

c) Non-linear interactions between tides and planetary waves.

d) Large-scale latent heat release due to deep convection in tropics.

Atmospheric tides are also produced through the gravitational effects of the Moon, Lunar (gravitational) tides are much weaker than solar (thermal) tides and are generated by the motion of the Earth's oceans (caused by the Moon) and to a lesser extent the effect of the Moon's gravitational attraction on the atmosphere.

1.3 nc(NetCdf) Profiles

It is a set of self-describing ,machine independent data format that supports creation ,access ,and sharing of array-oriented scientific data .

The project homepage is hosted by the unidata program at the University Corporation For Atmosphere Research(UCAR)

Extension of such file is .nc.

We have worked on netcdf files that have electron density at +/- 90 degrees latitude , 0 to 360 degrees longitude,200-300 km MSL_alt and other attribute and variables measured by COSMIC/FORMOSAT-3 satellite,n year 2012 and 2013 .

1.4 Why Nc(NetCdf) Profiles ?

1. NC files can be manipulated using Unidata's free netCDF tools, a set of libraries that provide interfaces in the Java, C, C++, Fortran, and Perl programming languages.

2. An NC file is a file encrypted with mcrypt, a Linux crypting utility that allows users to encrypt and decrypt files or data streams. It contains a file encrypted with an algorithm, such as md5, tiger, or whirlpool, specified by the user and secured with a passphrase.

1.5. Introduction to Data Mining.

Data mining is the process of discovering patterns in large data sets involving methods at the intersection of machine learning, statistics, and database systems.

The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.

Data mining involves six common classes of tasks.

- **Anomaly detection**
- **Association rule learning**
- **Clustering**
- **Classification**
- **Regression**
- **Summarization**

1.6.Different python Modules.

In this section we will discuss few of the important python Modules and inbuilt function we have used in our project.

Scipy

SciPy is a Python module that provides convenient and fast N-dimensional array manipulation. It provides many user-friendly and efficient numerical routines, such as routines for numerical integration and optimization. SciPy has modules for optimization, linear algebra, integration etc.

Scikit-learn

Scikit-Learn is a Python module for machine learning built on top of SciPy. It provides a set of common machine learning algorithms to users through a consistent interface. Scikit-Learn helps to quickly implement popular algorithms on datasets,such as clustering, classification, regression, etc.

Statsmodels

Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests.

Chapter 2 :Reading nc files with python

All the codes in this thesis are written in language python 2 .

We read the same netCDF file using python(because source code was meant to written in python language).To get all the require netCDF modules of python in order to plot graphs and manipulate the data accordingly ,we installed two important softwares:

1.Anaconda

2.NetCDF

Now one read the nc file after importing important modules.

Code to read and print various dimensions,variables,global attributes of a nc profile('ionPrf_C004.2012.001.00.40.G16_2010(2).2640_nc').

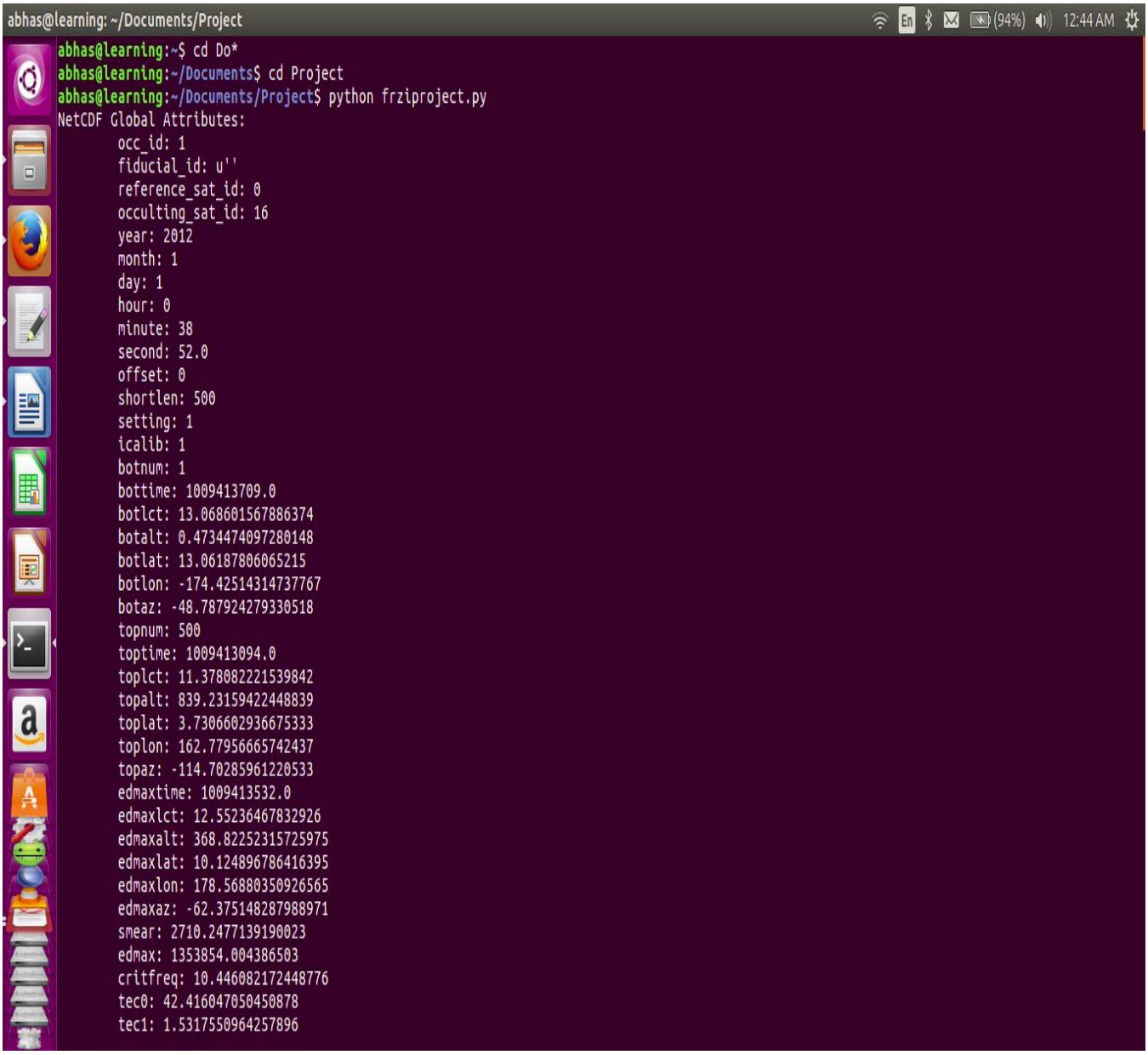
Code

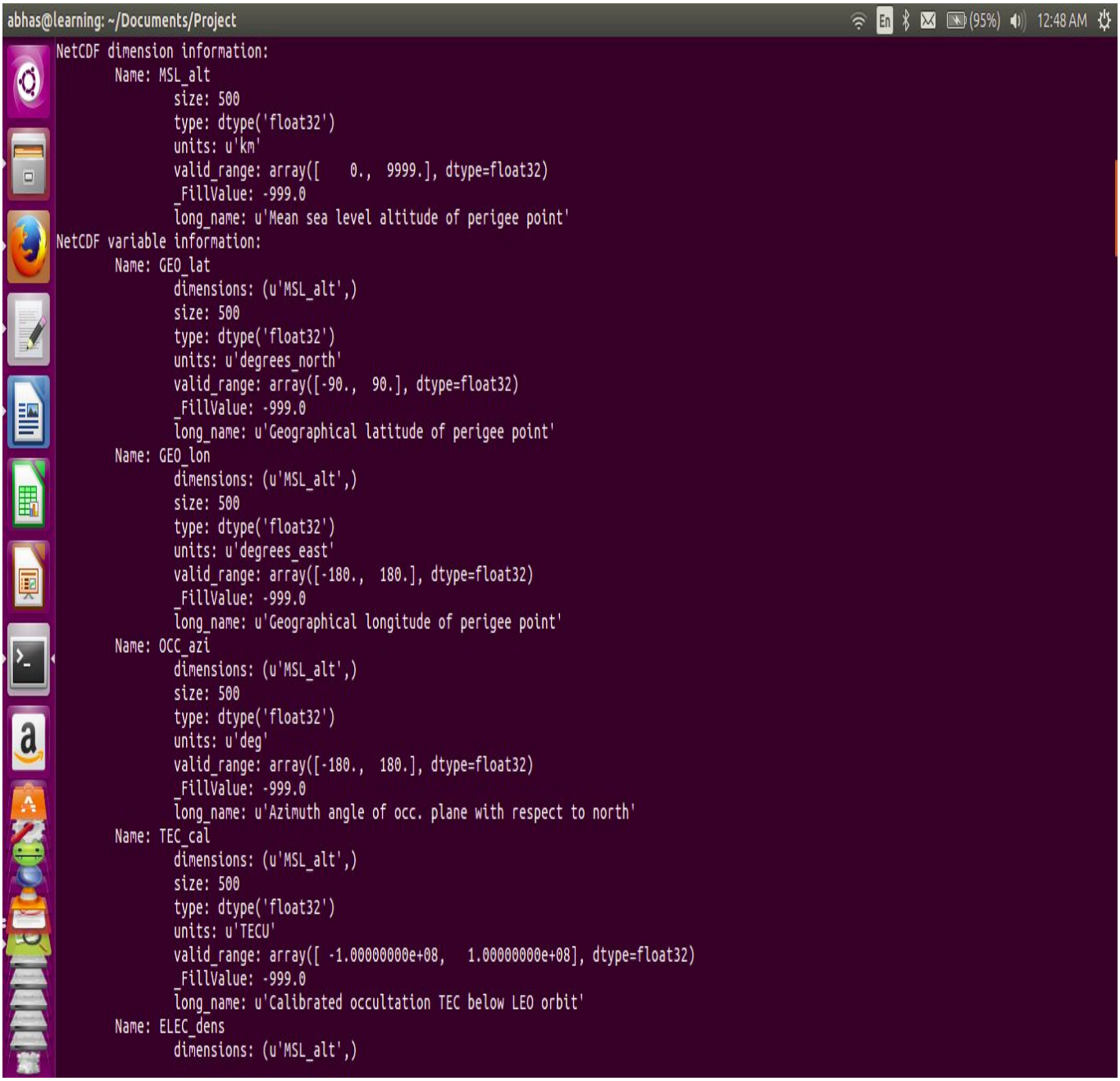
```
import numpy as np

from netCDF4 import Dataset


def ncdump(nc_fid, verb=True):
```

```
"""ncdump outputs dimensions, variables and their attribute
information.
"""
def print_ncattr(key):
"""
Prints the NetCDF file attributes for a given key
"""
Try:
print "\t\ttype:", repr(nc_fid.variables[key].dtype)
for ncattr in nc_fid.variables[key].ncattrs():
print '\t\t%s:' % ncattr,\
repr(nc_fid.variables[key].getncattr(ncattr))
except KeyError:
print "\t\tWARNING: %s does not contain variable attributes" % key
nc_attrs = nc_fid.ncattrs()
if verb:
print "NetCDF Global Attributes:"
for nc_attr in nc_attrs:
print '\t%s:' % nc_attr, repr(nc_fid.getncattr(nc_attr))
nc_dims = [dim for dim in nc_fid.dimensions]
if verb:
```

```
print "NetCDF dimension information:"

for dim in nc_dims:

print "\tName:", dim

print "\t\tsize:", len(nc_fid.dimensions[dim])

print_ncattr(dim)

nc_vars = [var for var in nc_fid.variables]

if verb:

print "NetCDF variable information:"

for var in nc_vars:

if var not in nc_dims:

print '\tName:', var

print "\t\tdimensions:", nc_fid.variables[var].dimensions

print "\t\tsize:", nc_fid.variables[var].size

print_ncattr(var)

return nc_attrs, nc_dims, nc_vars

nc_f = 'ionPrf_C004.2012.001.00.40.G16_2010(2).2640_nc'

nc_fid = Dataset(nc_f, 'r')

nc_attrs, nc_dims, nc_vars = ncdump(nc_fid)
```

We are giving 2 figures to show the output where one can see all the global attributes,different variables their dimensions and other informations.


abhas@learning: ~/Documents/Project
12:44 AM
abhas@learning:~$ cd Do*
abhas@learning:~/Documents$ cd Project
abhas@learning:~/Documents/Project$ python frziproject.py
NetCDF Global Attributes:
occ_id: 1
fiducial_id: u''
reference_sat_id: 0
occulting_sat_id: 16
year: 2012
month: 1
day: 1
hour: 0
minute: 38
second: 52.0
offset: 0
shortlen: 500
setting: 1
icalib: 1
botnum: 1
bottime: 1009413709.0
botlct: 13.068601567886374
botalt: 0.4734474097280148
botlat: 13.06187806065215
botlon: -174.42514314737767
botaz: -48.787924279330518
topnum: 500
toptime: 1009413094.0
toplct: 11.378082221539842
topalt: 839.23159422448839
toplat: 3.7306602936675333
toplon: 162.77956665742437
topaz: -114.70285961220533
edmaxtime: 1009413532.0
edmaxlct: 12.55236467832926
edmaxalt: 368.82252315725975
edmaxlat: 10.124896786416395
edmaxlon: 178.56880350926565
edmaxaz: -62.375148287988971
smear: 2710.2477139190023
edmax: 1353854.004386503
critfreq: 10.446082172448776
tec0: 42.416047050450878
tec1: 1.5317550964257896


abhas@learning: ~/Documents/Project
(95%) 12:48 AM
NetCDF dimension information:
Name: MSL_alt
size: 500
type: dtype('float32')
units: u'km'
valid_range: array([0., 9999.], dtype=float32)
_FillValue: -999.0
long_name: u'Mean sea level altitude of perigee point'
NetCDF variable information:
Name: GEO_lat
dimensions: (u'MSL_alt',)
size: 500
type: dtype('float32')
units: u'degrees_north'
valid_range: array([-90., 90.], dtype=float32)
_FillValue: -999.0
long_name: u'Geographical latitude of perigee point'
Name: GEO_lon
dimensions: (u'MSL_alt',)
size: 500
type: dtype('float32')
units: u'degrees_east'
valid_range: array([-180., 180.], dtype=float32)
_FillValue: -999.0
long_name: u'Geographical longitude of perigee point'
Name: OCC_azi
dimensions: (u'MSL_alt',)
size: 500
type: dtype('float32')
units: u'deg'
valid_range: array([-180., 180.], dtype=float32)
_FillValue: -999.0
long_name: u'Azimuth angle of occ. plane with respect to north'
Name: TEC_cal
dimensions: (u'MSL_alt',)
size: 500
type: dtype('float32')
units: u'TECU'
valid_range: array([-1.00000000e+08, 1.00000000e+08], dtype=float32)
_FillValue: -999.0
long_name: u'Calibrated occultation TEC below LEO orbit'
Name: ELEC_dens
dimensions: (u'MSL_alt',)

One can fetch values of these variables and attributes.

```
file = 'ionPrf_C004.2012.001.00.40.G16_2010(2).2640_nc'
ncfile =Dataset(file,'r')
alt= np.array(ncfile.variables['MSL_alt'][:],dtype=np.float32)
temp= np.array(ncfile.variables['TEC_cal'][:],dtype=np.float32)
ed= np.array(ncfile.variables['ELEC_dens'][:],dtype=np.float32)
oa = np.array(ncfile.variables['OCC_azi'][:],dtype=np.float32)
g_lat = np.array(ncfile.variables['GEO_lat'][:],dtype=np.float32)
g_lon = np.array(ncfile.variables['GEO_lon'][:],dtype=np.float32)
hour = nc_fid.getncattr('hour')
min = nc_fid.getncattr('minute')
sec = nc_fid.getncattr('second')
fs=nc_fid.getncattr('fileStamp')
```

After this one can print these values and visualise.All manipulation can be done similar as all list ,tuples and variables in python.

```
time=((hour[i])+(float(minute[i])/60)+(second[i]/3600))
print “Electron Density values”,ed
print  “Latitute”,g_lat
print “time of measurement”,time
```

Chapter 3 :Data Extraction

3.1 Extraction from single nc profile

To extract value from the all the variables or lists of **single** nc profile(electron density(ed),longitude(g_lon),latitude(g_lat) etc) with given constraints,one can do this by applying same list manipulation with conditional statements as we do with python lists.

Extraction example in our project.

We extracted the values of Electron Density and stored them in a list for each day ,following these constraints.

i. Where electron density is positive.

ii.Within +/-10 latitude.

iii.Within 0 to 360 degrees longitude.

iv.At 200km.

```
for i in range(len(msl_alt)):
        if msl_alt[i])==200 and -10<=g_lat[i]<=10 and ed[i]>=0:
                ed_new.append(ed[i])
                alt_new.append(msl_alt[i])
                lat_new.append(g_lat[i])
```

One can manipulate similarly as we do with python lists and variables

See this example.

```
count=0
n=0
ed_sum=0
h=0
l=0
inner=outer=0
while outer<8:
    inner=0
    l=0
    while inner<9:
        for i in range(len(lon_new)):
            if (-180+l)<=lon_new[i]<=(-140+l) and
(0+h)<=time[i]<=(3+h):
                count=count+1
                if count>1:
                    lon_final.append(-180+l)
                    time_final.append(0+h)
                    break
        if count<=1:
            sub_lon.append(-180+l)
```

```
                sub_time.append(0+h)
        l=l+40
        ed_sum=0
        inner=inner+1
        count=0
h=h+3
outer=outer+1
```

TABLE :

3.2.Extraction from multiple nc profile

Similarly one can work with multiple nc profiles ,we need to have name of all nc profiles with which we want to work (e.g

Extraction(multiple nc profile) example in our project.

Example 1:

```
j=.00123
while j<.047:
        c=2013.000+j
        path = "./projfiles/"+str(c)[:8]+"/"
        day = os.listdir( path )
        for m in range(len(day1)):
                nc_f = day[m]
        for i in range(len(msl_alt)):
                if 199<(msl_alt[i])<201  and -10<=g_lat[i]<=10 and
        ed[i]>=0:
                ed_new.append(ed[i])
                alt_new.append(msl_alt[i])
                lat_new.append(g_lat[i])
        j=j+.00100
```

3.3. Extraction and Data Storage, Data structure

AS we have already Extracted the data from given nc files. It is Essential to store the data so that can be used for further work.

In our project the data on which are working , is one -one mapping ,In other word if we talk about longitude , time and electron density so for each longitude value have corresponding electron density value.

We are using list Data structure to store the data, List is also kind of array but in python it is defined by list , here we are giving some list name which are used to store the key value in order to plot the graph.

Example :

```
ed_final=[]
fs_new=[]
amp=[]
hour=[]
time=[]
minute=[]
second=[]
ed_new=[]
lon_new=[]
alt_new=[]
file_new=[]
lon_final=[]
time_final=[]
sub_lon=[]
sub_time=[]
```

These all the empty lists are used for different purpose and storage, throughout our program. For example , ed_final[] is stand for electron density , it has all the extracted electron densities value corresponding their longitude and some other constraints(+/- 5 degree latitude and 200 km altitude).

Chapter 4 :Graph Plotting

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms.

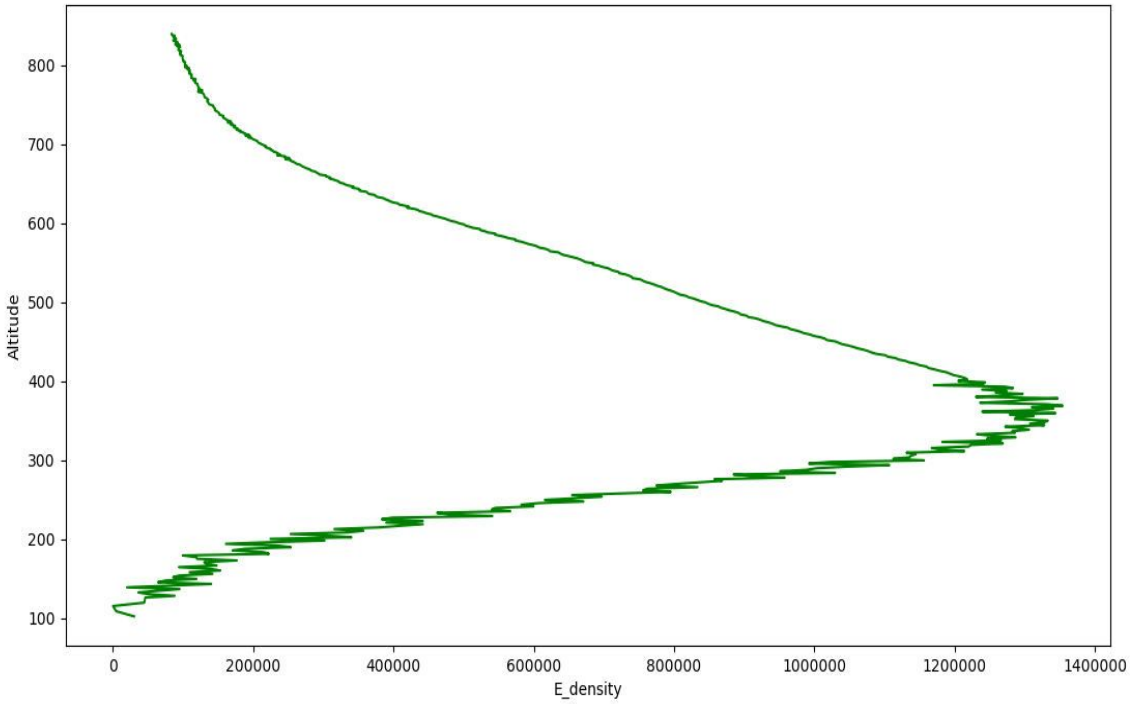
With the extraction of electron density and mean sea level altitude now can plot graph between them.

Note : To plot any type of graph/plot in matplotlib size of dependent and independent variables must be equal.

Example :

```
alt_new=[]
ed_new=[]
for i in range(len(msl_alt)):
        if ed[i]>0:
                alt_new.append(msl_alt[i])
                ed_new.append(ed[i])
```

```
fig,ax=plt.subplots()
ax.xaxis.set_ticks(np.linspace(min(ed_new),max(ed_new)+1,10))
ax.yaxis.set_ticks(np.linspace(min(alt_new),max(alt_new)+1,10))
plt.plot(ed_new,alt_new,'g');
plt.title("ED VS ALT");
plt.xlabel("E_density/cm3");
plt.ylabel("Altitude");
plt.show()
```

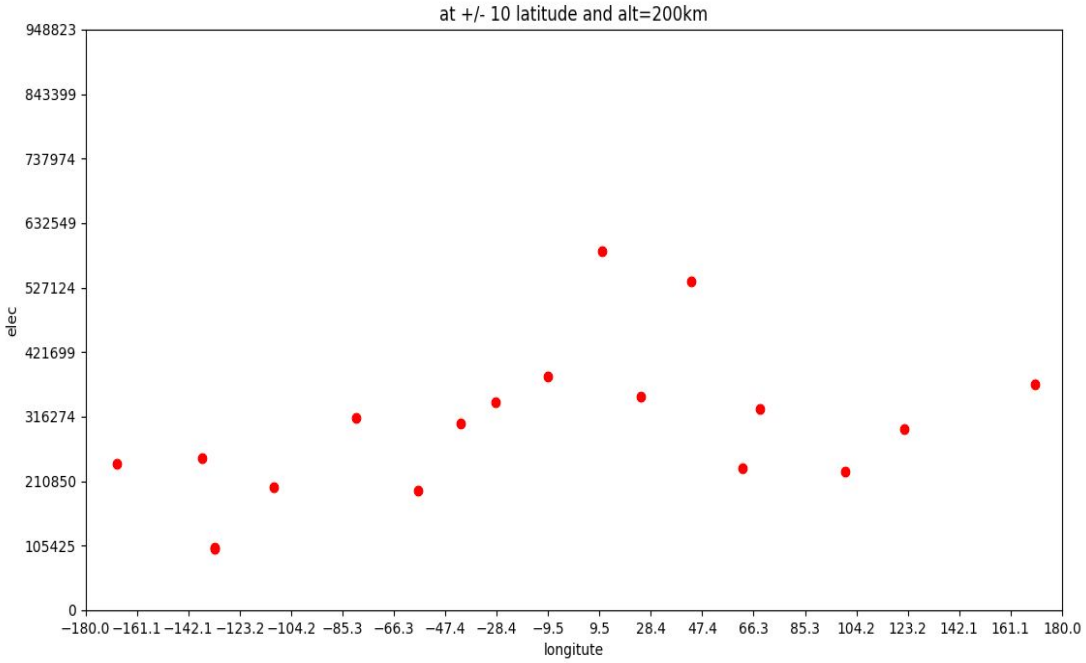


The coordinates of the points or line nodes are given by x, y.

The optional parameters are convenient way for defining basic formatting like color, marker and linestyle. It's a shortcut string notation described in the *Notes* section below.

plot(x, y) -----> plot x and y using default line style and color
plot(x, y, 'bo') -----> plot x and y using blue circle markers
plot(y) ------>plot y using x as index array 0..N-1
plot(y, 'r+') ------>ditto, but with red plusses
Example 2:

1. for i in range(len(msl_alt)):
    a. if (msl_alt[i])==200 and -10<=g_lat[i]<=10 and ed[i]>=0:
        i. lon_new.append(g_lon[i])
        ii. ed_new.append(ed[i])
        iii. alt_new.append(msl_alt[i])
2. pylab.ylim(0,max(ed))
3. pylab.xlim(-180,180)plt.yticks(np.linspace(0,max(ed),10,endpoint =True))
4. plt.xticks(np.linspace(-180,180,20,endpoint=True))
5. pylab.xlabel('Longitude')
6. pylab.ylabel('Electron Density')
7. pylab.title('at +/- 10 latitude and alt=200km')
8. plt.plot(lon_new, ed_new, 'ro')
9. plt.show()

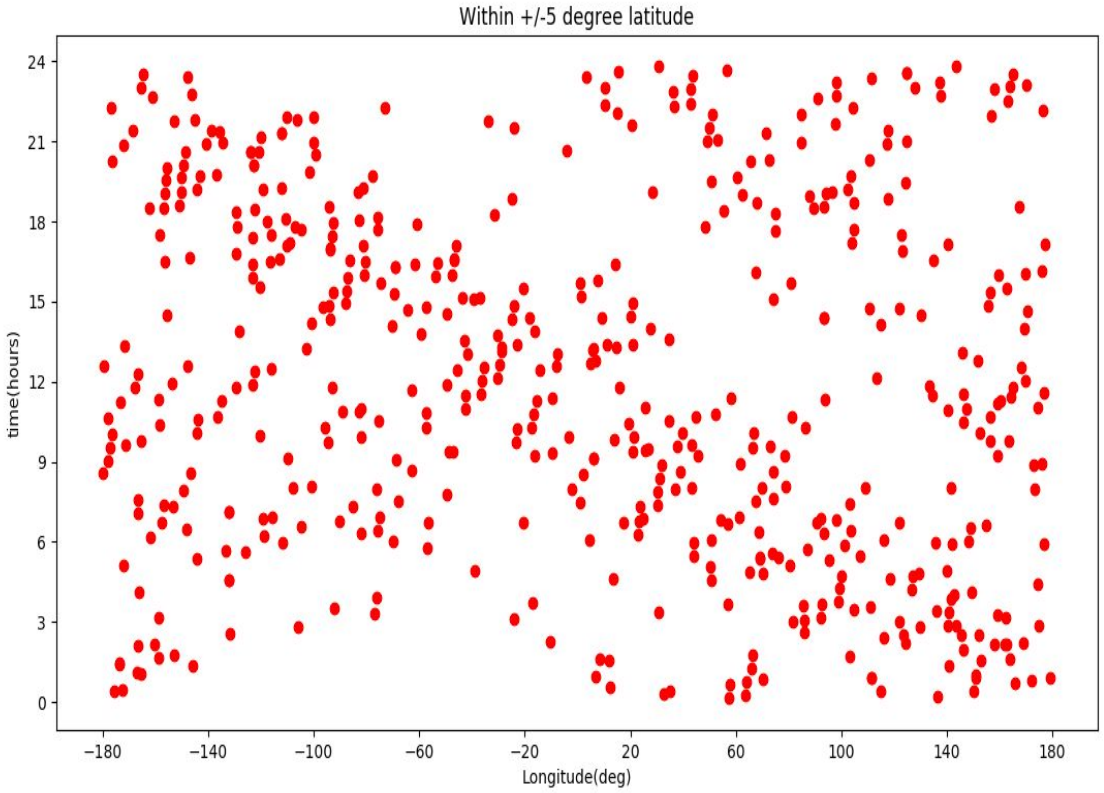


Example2

```
for i in range(len(msl_alt)):
        if 199<(msl_alt[i])<201  and -10<=g_lat[i]<=10 and ed[i]>=0:
                hour.append(hour)
                minute.append(min)
                second.append(sec)
                lon_new.append(g_lon[i])
                ed_new.append(ed[i])
```

```
for i in range(len(hour)):
    time.append((hour[i])+(float(minute[i])/60)+(second[i]/3600))
count=0
n=0
ed_sum=0
h=0
l=0
inner=outer=0
while outer<8:
    inner=0
    l=0
    while inner<9:
        for i in range(len(lon_new)):
        if (-180+l)<=lon_new[i]<=(-140+l) and (0+h)<=time[i]<=(3+h):
                    count=count+1
                    if count>1:
                        lon_final.append(-180+l)
                        time_final.append(0+h)
                        break
        if count<=1:
```

```
                sub_lon.append(-180+l)
                sub_time.append(0+h)
        l=l+40
        ed_sum=0
        inner=inner+1
        count=0
    h=h+3
    outer=outer+1

fig, ax = plt.subplots()
ax.set_axisbelow(True)
ax.xaxis.set_ticks(np.arange(-180,181, 40))
ax.yaxis.set_ticks(np.arange(0,25, 3))
plt.xlabel('Longitude(deg)')
plt.ylabel('time(hours)')
plt.title('Within +/-5 degree latitude')
plt.plot(lon_new,time, 'ro')
plt.show()
```



This graph is, a output of given above code where the dot points are signify that there is electron density value for corresponding longitude and time value.

Chapter 5 :Curve Fitting

5.1. Curve fitting methods

Curve fitting is the process of constructing a curve , or mathematical function, that has the best fit to a series of data points possibly subject to constraints. Curve fitting can involve either interpolation where an exact fit to the data is required, or smoothing in which a "smooth" function is constructed that approximately fits the data. A related topic is regression analysis which focuses more on questions of statistical inferences such as how much uncertainty is present in a curve that is fit to data observed with random errors.

Most commonly, one fits a function of the form $y=f(x)$.

Starting with a first degree polynomial equation:

$y = a x + b$.

This is a line with slope *a*. A line will connect any two points, so a first degree polynomial equation is an exact fit through any two points with distinct x coordinates.

If the order of the equation is increased to a second degree polynomial, the following results:

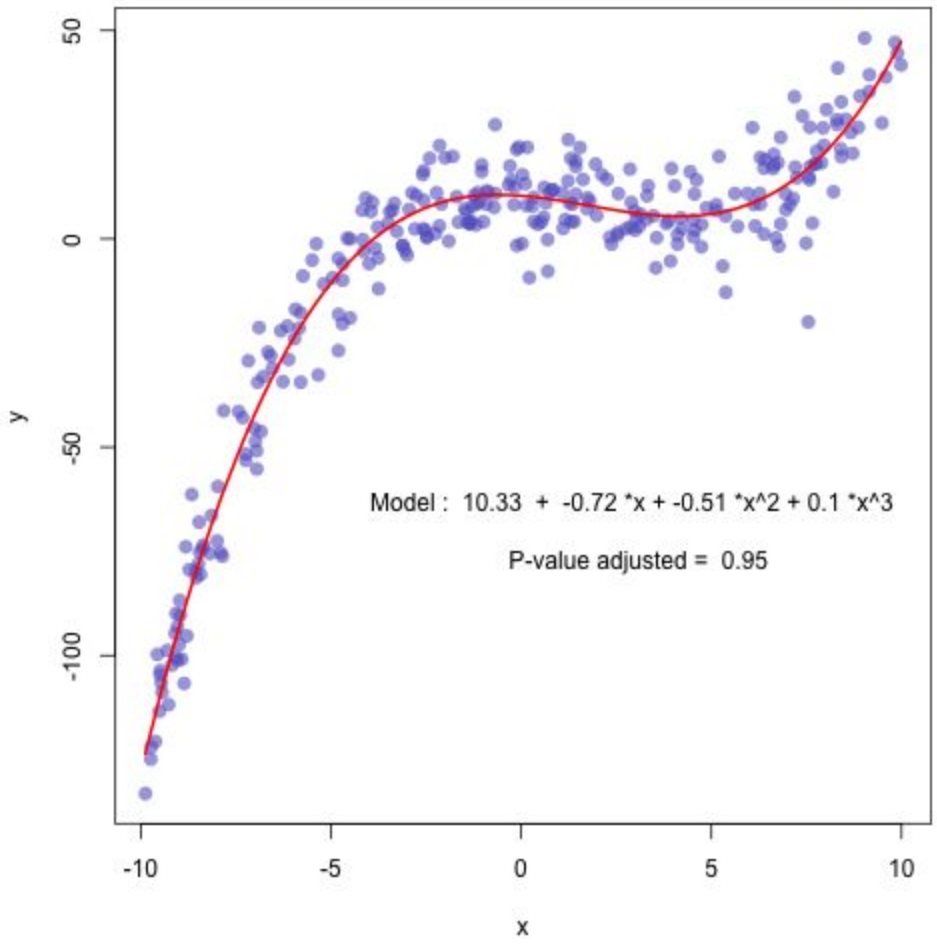
$y = ax^2+bx+c$

This will exactly fit a simple curve to three points.

If the order of the equation is increased to a third degree polynomial, the following is obtained:

y = a x ^3 + b x^ 2 + c x + d

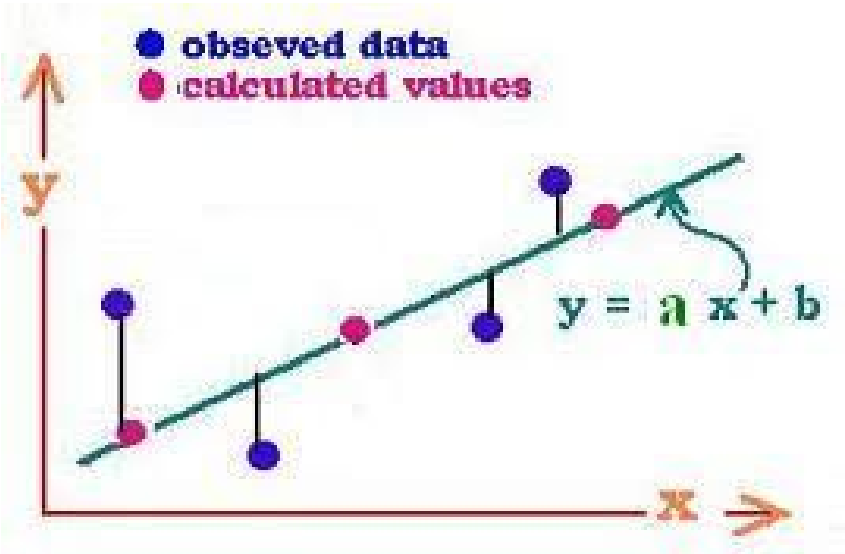
This will exactly fit four points.



Least Square Fitting method

A mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the offsets ("the residuals") of the points from the curve. The sum of the *squares* of the offsets is used instead of the offset absolute values because this allows the residuals to be treated as a continuous differentiable quantity. However, because squares of the offsets are used, outlying points can

have a disproportionate effect on the fit, a property which may or may not be desirable depending on the problem at hand.



Example:

In this project we fitted a sine curve on the data point of longitude against electron density shown earlier .

We fitted the sine curve on the obtained data point plots for each day at 200 km altitude by using least square fitting method .

Lmfit and scipy.optimize modules are used for curve fitting in python.

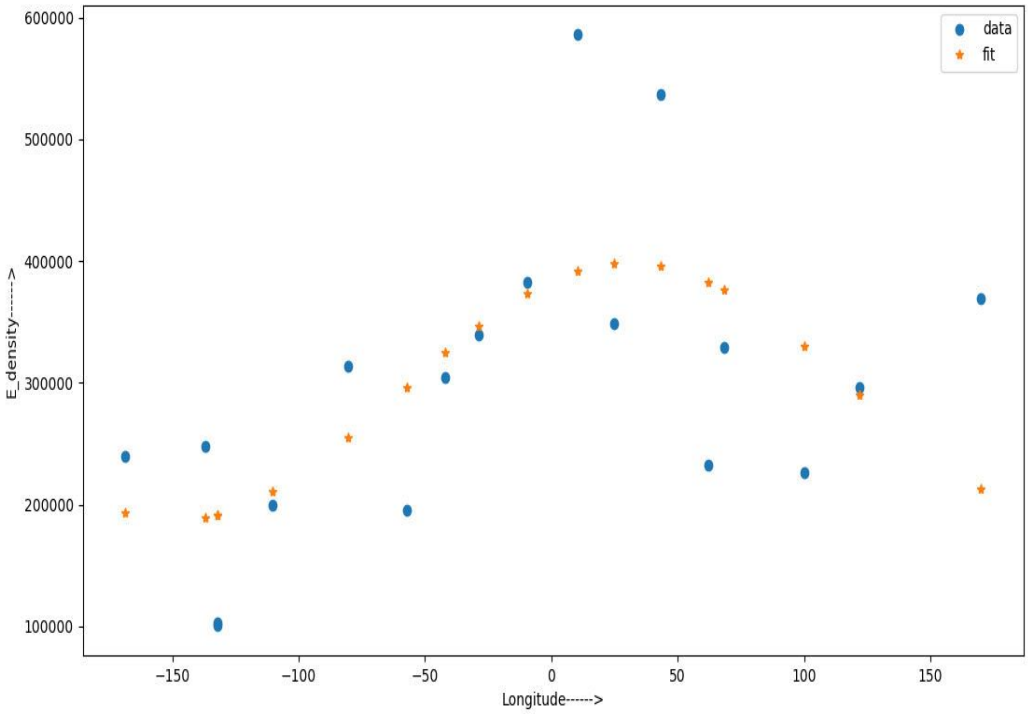
Here we fitted a sine curve having equation :

a+bsin(3.14/180*longitude+c)

where a is a constant ,b is the amplitude and c is the phase.

```
1. xdeg=lon_new
2. y=ed_new
3. def sinefunction(x, a, b, c):
4. return a + b * np.sin(x*np.pi/180.0 + c)
5. smodel = Model(sinefunction)
6. result = smodel.fit(y, x=xdeg, a=0, b=70000, c=0)
7. amp.append(abs(result.params['b'].value))
8. plt.plot(xdeg, y, 'o', label='data')
9. plt.plot(xdeg, result.best_fit,'*', label='fit')
10.    pylab.xlim(-180,180)plt.yticks(np.linspace(0,max(ed),10,endpoi
   nt=True))
11.    plt.xticks(np.linspace(-180,180,20,endpoint=True))
12.    pylab.xlabel('Longitude')
13.    pylab.ylabel('Electron Density')
14.    pylab.title('at +/- 10 latitude and alt=200km')
15.    plt.legend()
16.    plt.show()
```

Graph obtained after fitting the sine curve of given equation on the data points obtained by plot of Electron density against longitude for day 29.



5.2. Extraction of Parameters

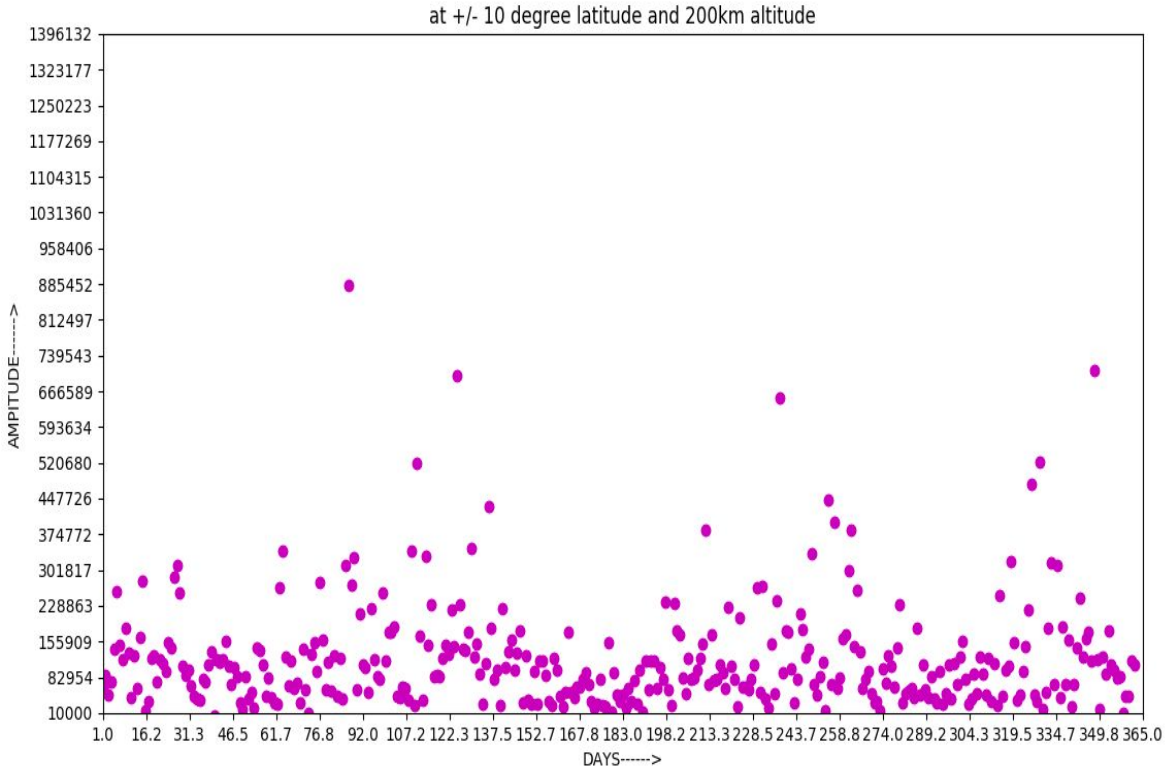
Code line number 7 extract value of parameter b(i.e amplitude) and appends to the array amp.

For whole 1 year (i.e 365 days) of we fitted a sine curve of the given equation and obtained amplitude (value of parameter b).

Similarly other parameters a and c can also be obtained.

Finally, we have drawn the graph of obtained Amplitude against All days of the year at 200 km altitude .

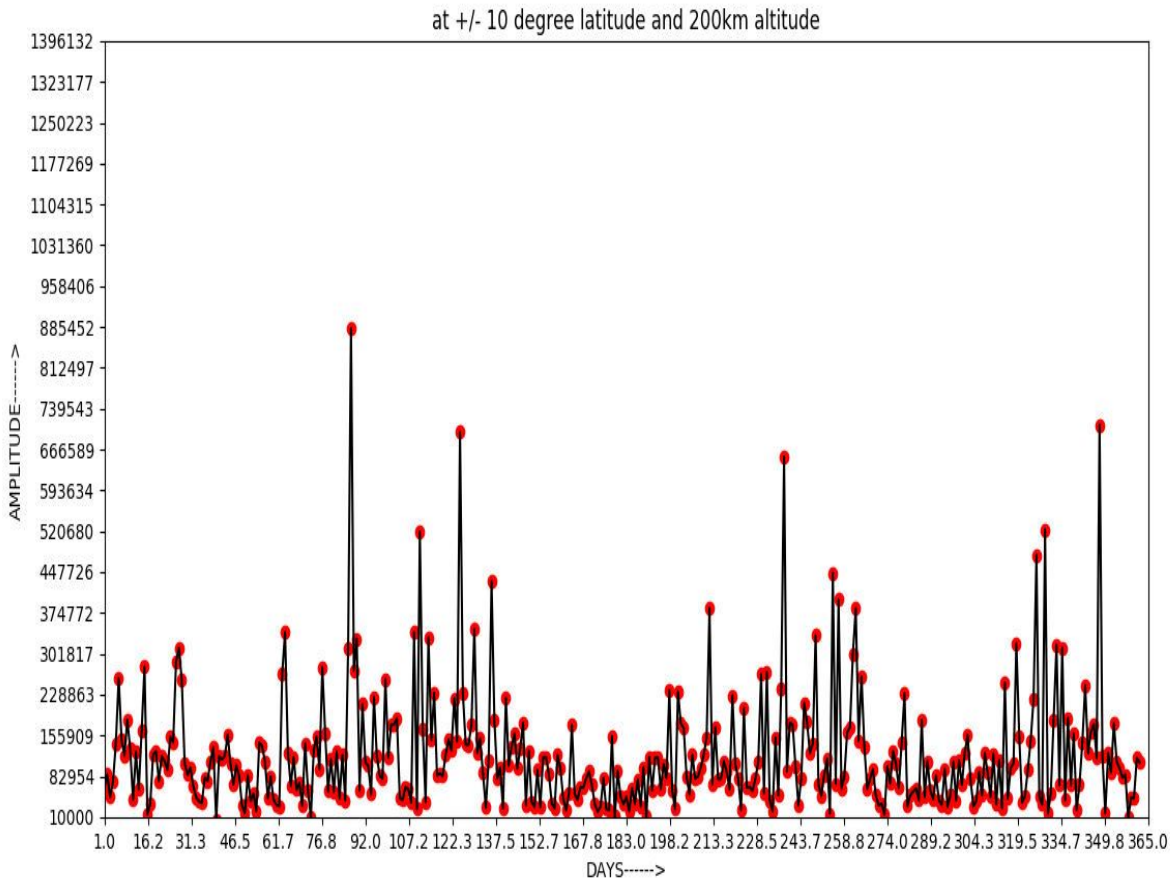
1st graph is just showing the data points obtained in the plot of days against amplitude.



We plotted the graph between days of the year to amplitude of fitted sine curve.

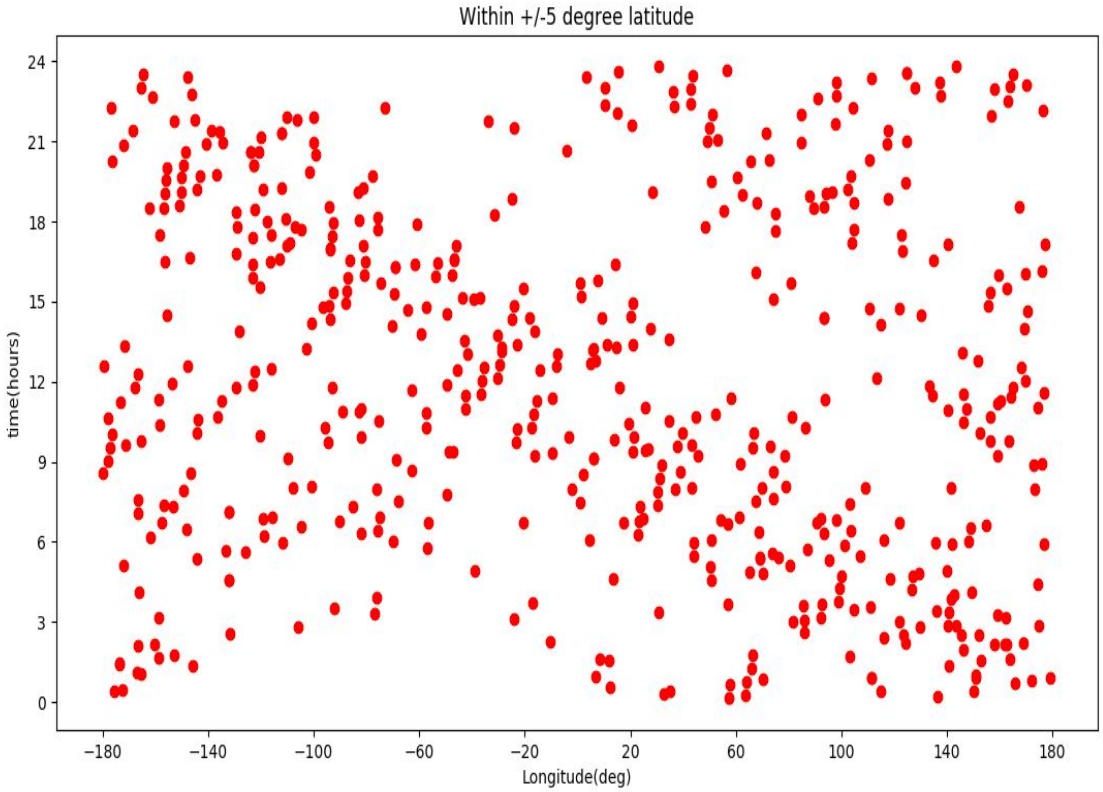
```
x=np.linspace(1,365,365)

pylab.ylim(10000,max(amp))

pylab.xlim(1,365)

plt.yticks(np.linspace(10000,max(amp),20,endpoint=True))

plt.xticks(np.linspace(1,365,25,endpoint=True))pylab.xlabel('DAYS------>')
```

```
pylab.ylabel('AMPITUDE------>')
pylab.title('at +/- 10 degree latitude and 200 km altitude')
plt.plot(x, gamp, 'mo')
plt.plot(x,gamp,'k-')
plt.show( )
```



Chapter 6 :Binning and Grid Formation on Data

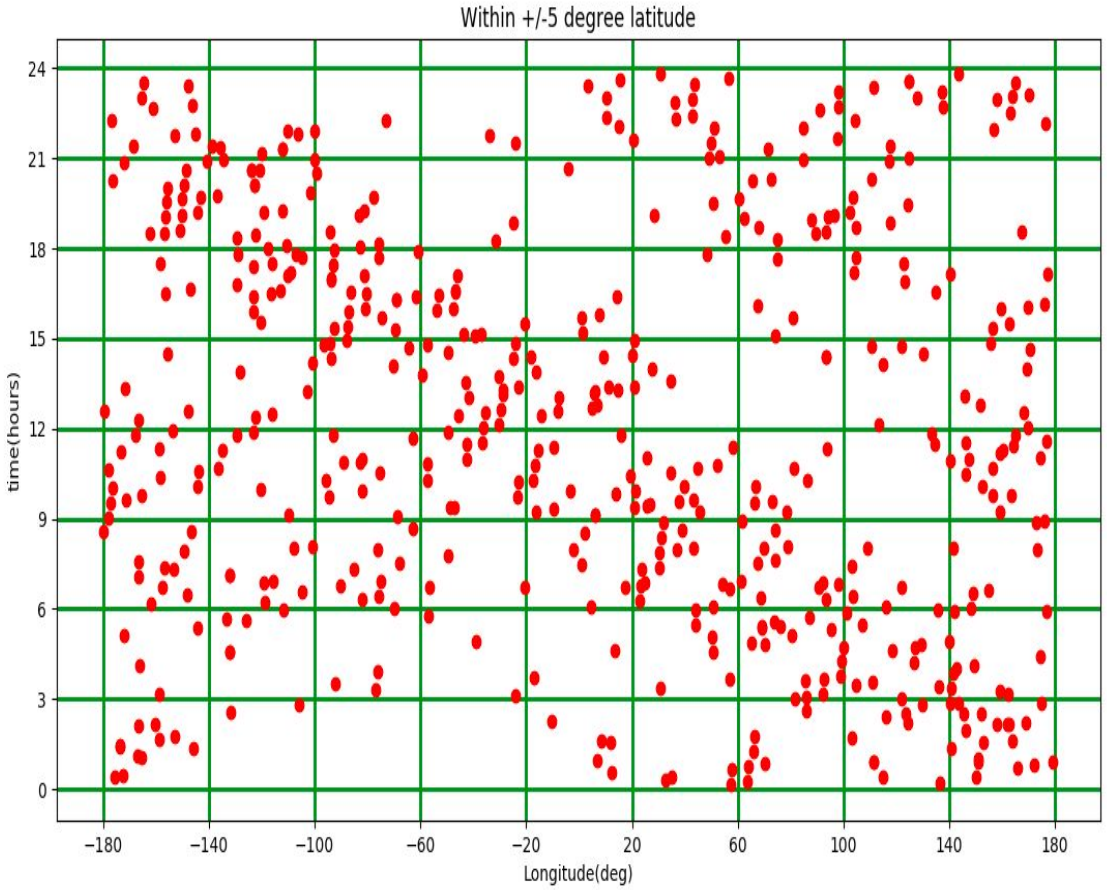
We used bins to get a uniform sampling of data with reasonable resolution.



We have chosen our bin-size as 40 degree on longitude and 3 hours on time.

We used grid formation to show bins explicitly.

```
fig, ax = plt.subplots()
ax.set_axisbelow(True)
ax.xaxis.set_ticks(np.arange(-180,181, 40))
ax.yaxis.set_ticks(np.arange(0,25, 3))
plt.xlabel('Longitude(deg)')
plt.ylabel('time(hours)')
plt.title('Within +/-5 degree latitude')
plt.plot(lon_new,time, 'ro')
ax.grid(linestyle='-', linewidth='2', color='green')
plt.grid(True)
plt.show()
```

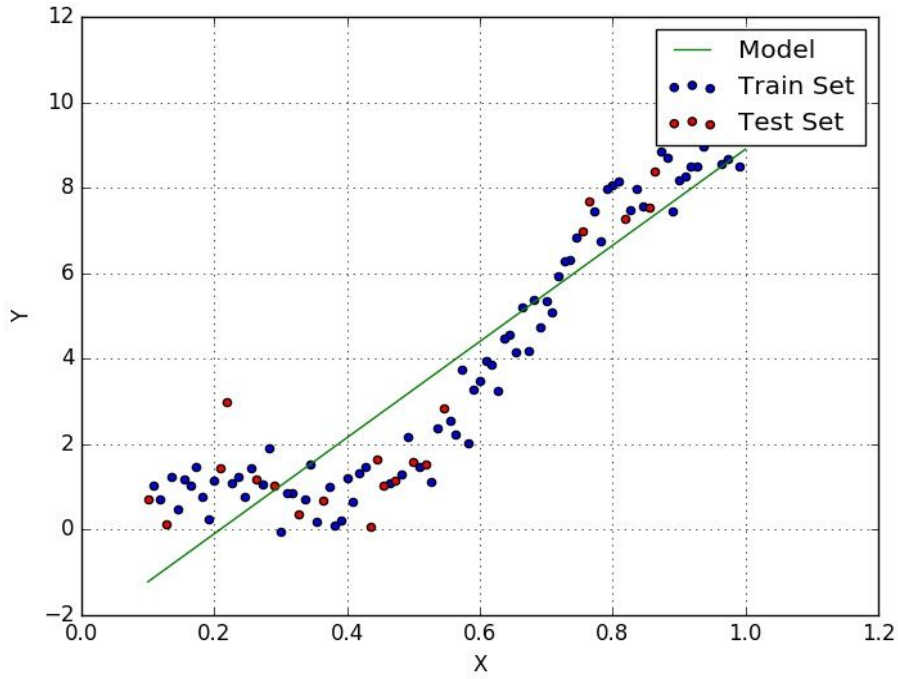

Within +/-5 degree latitude
time(hours)
Longitude(deg)
24
21
18
15
12
9
6
3
0
−180
−140
−100
−60
−20
20
60
100
140
180

Chapter 7 :Regression and Interpolation

7.1. Linear and Non-linear Regression

Linear regression is a kind of statistical analysis that attempts to show a relationship between two variables. Linear regression looks at various data points and plots a trend line. Linear regression can create a predictive model on apparently random data.

Nonlinear regression is a regression in which the dependent or criterion variables are modeled as a non-linear function of model parameters and one or more independent variables. There are several common models, such as Asymptotic Regression/Growth Model,



In this project , data points in each bins give the surety that each dot points have their corresponding electron density value. But here we assigned one subscript on the behalf all dot points which resides in one bin. And these subscripts define the average of electron densities of all points of single bin.

Assigning Subscript and Electron density

We assigned subscript to each bin.

Then for all data points within each bin we have taken Electron Density of each data point and assigned average value of Electron Density to its Subscript.

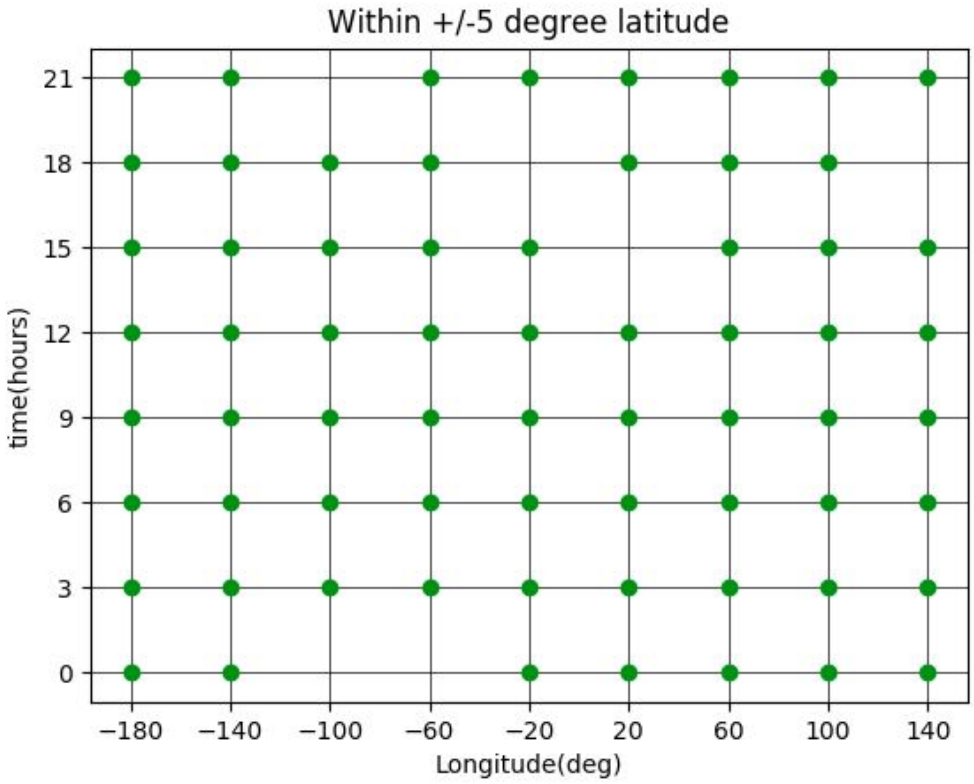
Purpose of assigning a subscript(representative) is to denote the the property of all data points in each bin .

In this project each subscript will have value of average electron densities of all data point in a bin.

For example subscript at coordinate (-60,12) of previous plot will have value average electron density of all points in the bin(-60-20,12-15)

```
count=0
n=0
ed_sum=0
h=0
l=0
inner=outer=0
while outer<8:
    inner=0
    l=0
```

```
    while inner<9:
        for i in range(len(lon_new)):
            if (-180+l)<=lon_new[i]<=(-140+l) and
(0+h)<=time[i]<=(3+h):
                count=count+1
                if count>1:
                    lon_final.append(-180+l)
                    time_final.append(0+h)
                    break
        if count<=1:
            sub_lon.append(-180+l)
            sub_time.append(0+h)
        l=l+40
        ed_sum=0
        inner=inner+1
        count=0
    h=h+3
    outer=outer+1
```



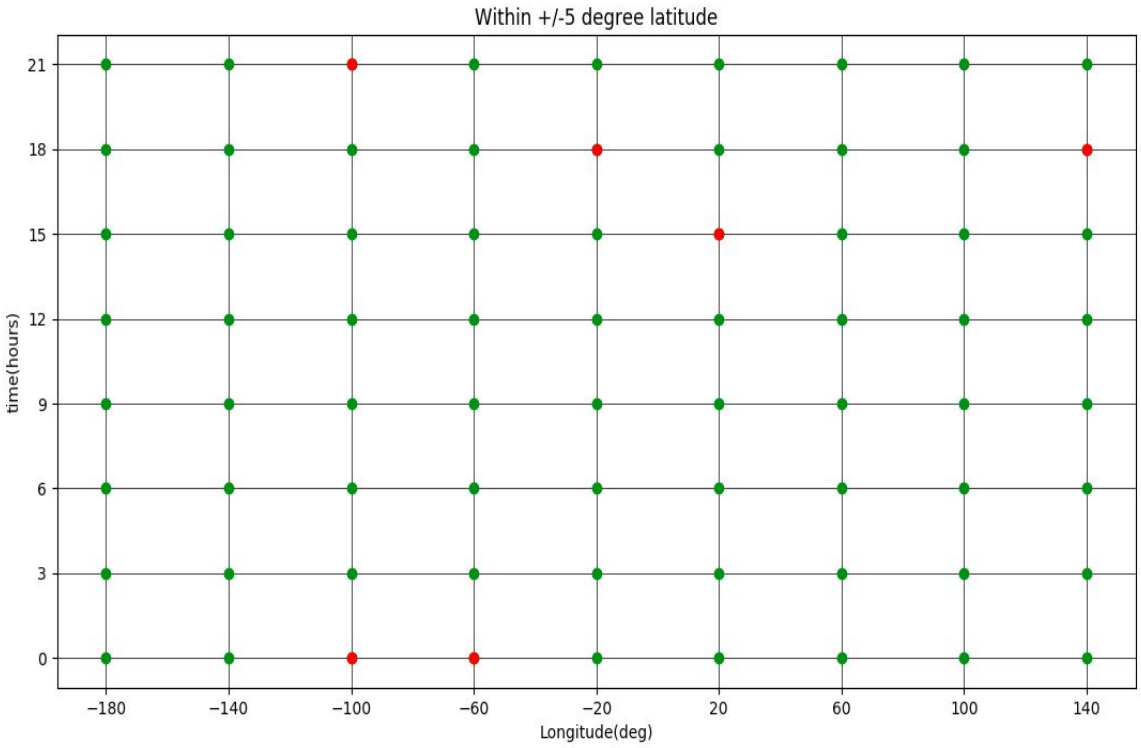
Here one can observe that few subscripts are missing .

For example (-100,0),(20,15) and others.

Reason is that we have taken a threshold number of data points which should be crossed in the bin to have a representative.We have taken 2 as the threshold number.

So if any bin contains less than 2 data points we haven't assigned any subscript for it,because taking average value of only two data points may lead to incorrect value of average electron density.

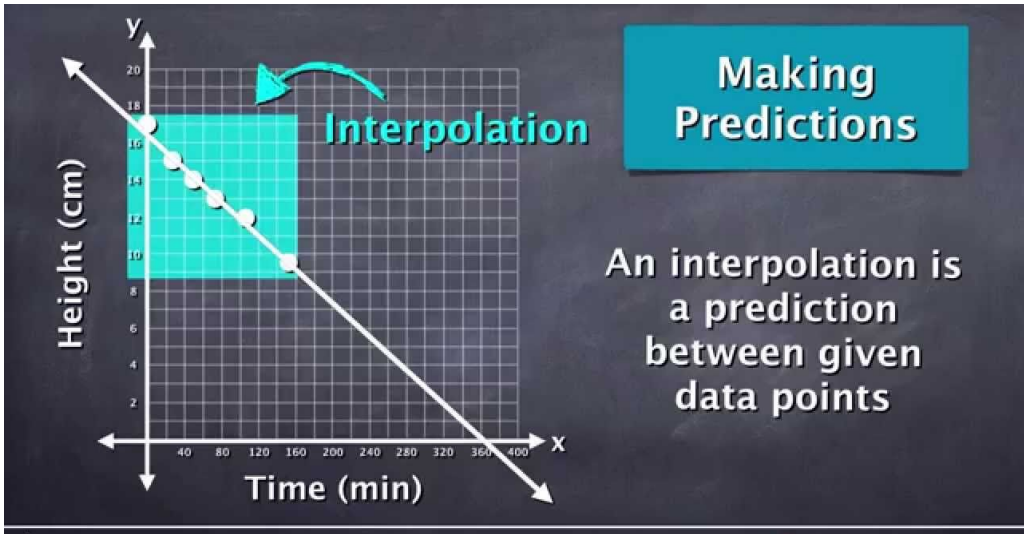
To show the missing points in the previous plot we have plotted one graph below.



As one can notice coordinates pointed in red color are the missing points

7.2 Interpolation

Interpolation is an estimation of a value within two known values in a sequence of values. Polynomial Interpolation is a method of estimating values between known data points. When graphical data contains a gap, but data is available on either side of the gap or at a few specific points within the gap, interpolation allows us to estimate the values within the gap.



We have chosen machine learning based linear regression over interpolation,because interpolation predicts the continuous function on basis of neighbouring points while with machine learning we can train our function on the whole data set and hence accuracy will be more when we will try to predict missing values.

We have used BayesianRidge() regression module of linear regression which is an inbuilt python module used to predict missing dependent variable values with respect to independent variables.

In this project longitude and time are independent variables while electron density is the dependent variable.

ed_sum=0

count=0

```
n=0
h=0
l=0
inner=outer=0
while outer<8:
    inner=0
    l=0
    while inner<9:
        for i in range(len(lon_new)):
        if (-180+l)<=lon_new[i]<=(-140+l) and (0+h)<=time[i]<=(3+h):
                    ed_sum=ed_sum+ed_new[i]
                    count=count+1

        if count>1 and (ed_sum/float(count))>0:
                ed_final.append(ed_sum/count)
                count=0

        l=l+40
        ed_sum=0
        inner=inner+1
    h=h+3
    outer=outer+1
```

Implementation of Machine learning with BayesianRegression linear model.

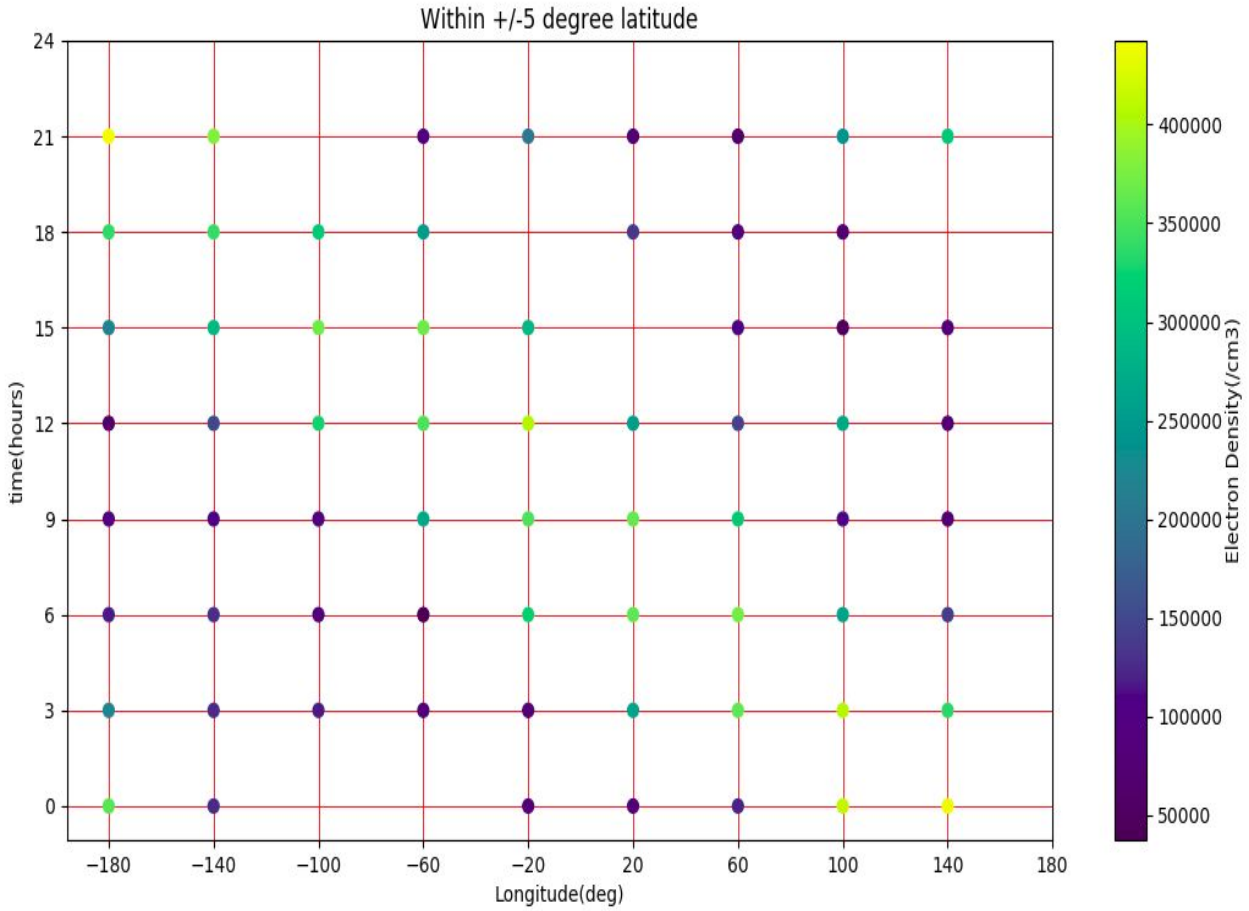
```
X1 = np.vstack((lon_final, time_final)).T

X2=np.vstack((sub_lon, sub_time)).T

Y=ed_final

reg = linear_model.BayesianRidge()

reg.fit(X1, Y)

BayesianRidge(alpha_1=1e-05, alpha_2=1e-05, compute_score=False,
copy_X=True,fit_intercept=True, lambda_1=1e-06, lambda_2=1e-06,
n_iter=300,normalize=False, tol=0.001, verbose=False)

for x in X2:

    ed_final.append(float(reg.predict ([x])))

lon_final=lon_final+sub_lon

time_final=time_final+sub_time
```

Results of this code is shown in the next chapter

Chapter : 8 Color Mapping and Contour Plot

We plotted a graph taking Longitude and Time as independent variables and Electron Density as Dependent Variable.We have used color mapping to show values of average electron densities

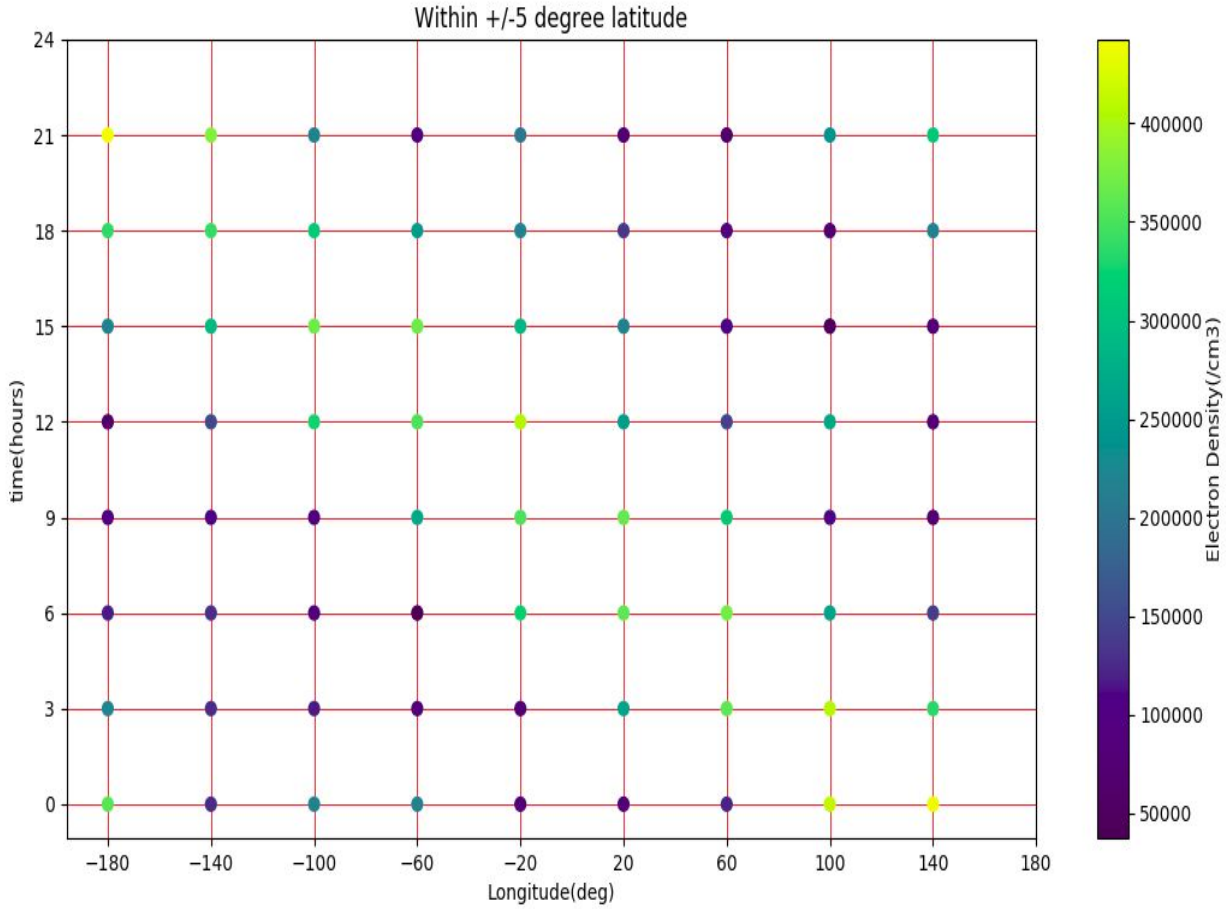
```
fig, ax = plt.subplots()
ax.set_axisbelow(True)
plt.scatter(lon_final,time_final,40,c=ed_final)
ax.xaxis.set_ticks(np.arange(-180,181, 40))
ax.yaxis.set_ticks(np.arange(0,25, 3))
plt.xlabel('Longitude(deg)')
plt.ylabel('time(hours)')
plt.title('Within +/-5 degree latitude')
ax.grid(linestyle='-', linewidth='.5', color='red')
cbar=plt.colorbar()
cbar.set_label("Electron Density(/cm3)",labelpad=2)
plt.show()
```



This plot is before finding the missing values of Electron densities using Machine learning.

One can observe that missing coordinates doesn't have average electron density value so as discussed earlier after finding the missing values of Electron densities using Machine learning.

The graph will have those missing values.



Now we are having values of missing average electron densities

Contour plot

After finding the missing values of Electron densities using Machine learning.

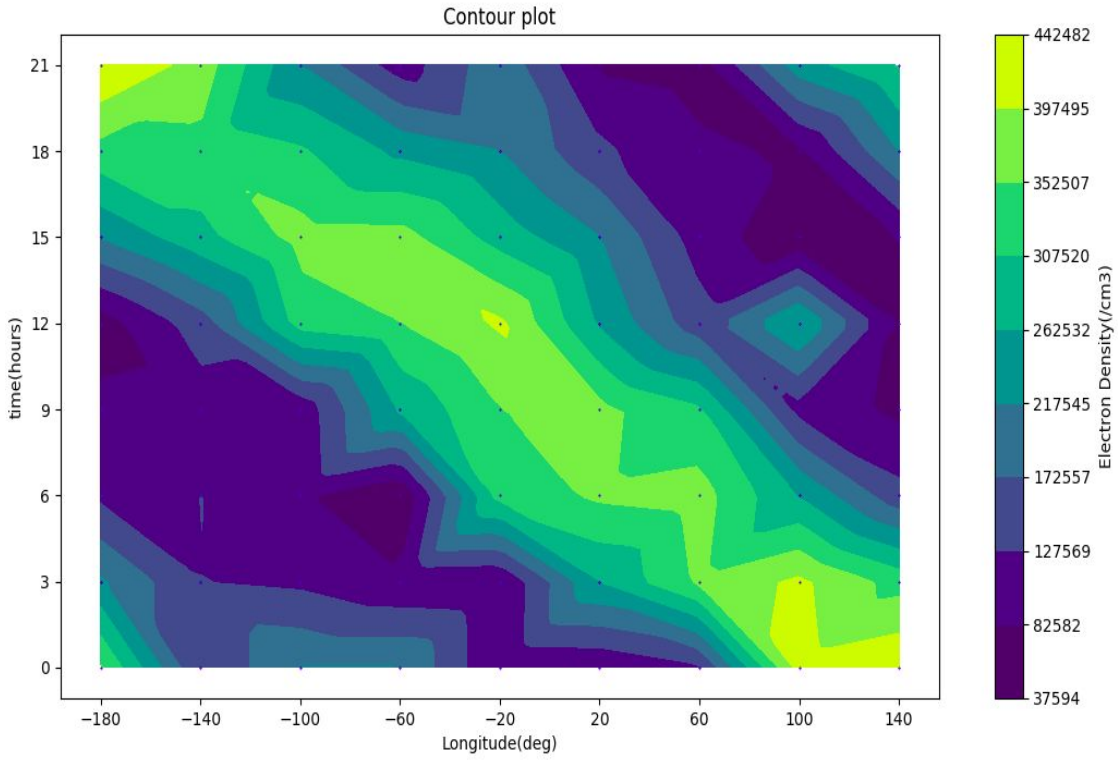
We plotted a contour plot taking Longitude and Time as independent variables and Electron Density as

Dependent Variable.

Contour plot gives more insight of variation of Electron Density in ionosphere.Previous plots also contains the same information and can draw these informations but with contour plot it is easy to conclude.

```
fig, ax = plt.subplots()

ax.set_axisbelow(True)

x=lon_final

y=time_final

z=ed_final

levels=np.linspace(min(ed_final),max(ed_final),10)

xi=np.linspace(-180,140,len(lon_final))

yi=np.linspace(0,21,len(lon_final))

zi=griddata((x,y),z,(xi[None,:],yi[:,None]),method="linear")

cs=plt.contourf(xi,yi,zi,levels=levels)

plt.xlabel('Longitude(deg)')

plt.ylabel('time(hours)')
```

```
plt.title('Contour plot')
cbar=plt.colorbar()
plt.scatter(x,y,marker='o',c='b',s=0.2)
ax.xaxis.set_ticks(np.arange(-180,141, 40))
ax.yaxis.set_ticks(np.arange(0,22, 3))
cbar.set_label("Electron Density(/cm3)",labelpad=5)
plt.show()
```



Summary and Conclusions

Although the results we found in this project is more or less known previously i.e. high solar radiation will lead to more electron density in the ionosphere but after working on 1 year(2013) of data measured by COSMIC satellite through data extraction,its storage and finally plotting these many plots and graphs now we can visualise the same concepts of electron density.

With Rotation of Earth (From Longitude 180 degrees to -180 degrees) and increase in time (0 -24 hours).

We can observe higher Electron Densities during day time and significantly low Electron Densities during night.

One of the reasons might be that during day ,radiation from sun will be higher that will lead to more ionisation in the ionosphere and hence High Electron density measurements.

References and Bibliography

Python 2.7.14 documentation:https://docs.python.org/2/

Matplotlib:https://matplotlib.org/

Scikit-learn:http://scikit-learn.org/stable/

Stackoverflow:https://stackoverflow.com/

Github:https://github.com/

Anaconda:https://anaconda.org/

Ubuntuforum:https://www.ubuntu.com/

Youtube:https://www.youtube.com/

Wikipedia:https://www.wikipedia.org/

Thanks to our guide

Our guide Dr.Uma Das ma’am has helped us throughout this project of 2 semesters till now .Since She herself has worked in this field so she has expertise and experience that helped us to understand objective of this project.

Right From the beginning she always disintegrated the project in small and singular tasks that can be easy to understand and implement.And most important is that she allowed us to implement tasks with methods we were comfortable with and she has given us enough time for implementation at the same time she has monitored the progress regularly.