**KATHMANDU UNIVERSITY**

**SCHOOL OF ENGINEERING**

**DEPARTMENT OF GEOMATICS ENGINEERING**

A PROJECT REPORT ON

# PREPARATION OF IONOSPHERIC MODEL (TOTAL ELECTRON CONTENT MAP) OF NEPAL USING THE CORS DATA

In Partial Fulfillment of the Requirements for the Bachelor's Degree in Geomatics Engineering


<u>Project Group Members</u>

| | |
|---|---|
| Dhana Bhatta | [018966-15] |
| Binabh Devkota | [018968-15] |
| Ashim Sharma | [018983-15] |
| Narendra Bhatta | [015972-13] |

<u>Supervisors</u>

Mr. Niraj Manandhar

Mr. Basant Awasthi


September 2019

# AUTHORIZATION

We, hereby declare that Binabh Devkota, Ashim Sharma, Dhana Bhatta and Narendra Bhatta are the authors of this project. We fully authorize the Kathmandu University to lend this project report to other institutions or individuals for the purpose of scholarly research. We further authorize the Kathmandu University to reproduce this document by photocopying, electronic replication or by other means, in total or in part without altering the authorship or the contents of the documents, as requested by other institutions or individuals for the purpose of scholarly research.

Binabh Devkota

Ashim Sharma

Dhana Bhatta

Narendra Bhatta

September 2019

# DISSERTATION EVALUATION

["PREPARATION OF IONOSPHERIC MODEL (TOTAL ELECTRON CONTENT MAP) OF NEPAL USING THE CORS DATA"]

by

Binabh Devkota, Ashim Sharma, Dhana Bhatta, Narendra Bhatta

This is to certify that I have examined this thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the Final Independent Project (GEOM 410) examination committee have been made.

Mr. Niraj Manandhar, Supervisor

Mr. Basant Awasthi, Supervisor

External Examiner

Dr. Subash Ghimire, Acting Head

Department of Geomatics Engineering

September 2019

# ACKNOWLEDGEMENT

Many people directly and indirectly have helped us for the completion of this project. We would like to take an opportunity to acknowledge every one of them.

Firstly, we would like to acknowledge Kathmandu University and Department of Geomatics Engineering for their continuous support, encouragement and guidance to carry out this project. The CORS established in the premises of Kathmandu University have motivated us to join the cause. Also special thanks to UNAVCO website for making their data publicly available.

We would like to express our immense gratitude and acknowledgement to Mr. Niraj Manandhar for his admiring contribution and full support throughout the project as the first supervisor. We are also indebted to our second supervisor Mr. Basant Awasthi for his valuable suggestion, guidance and comments in our project. Without their guidance and support, this project would not have been completed.

We would also like to put forward our gratitude to Mr. Bigyan Banjara who have helped us with a lot of practical suggestions from the depth of his expertise and Mr. Kutubuddin Ansari for his invaluable suggestions from his experience in this sector.

We are also very thankful towards our senior Mr. Shangharsha Thapa for his kind cooperation and help with all our queries, access to the study materials and his inspiring words.

Also, we would like to thank all faculty members of Kathmandu University who have directly and indirectly helped us in our project.

Finally, we would like to thank all of our classmates (batch 2015) for the support and willingness to spend some time with us for the completion of this project. Last but not least, thanks to our family members and all helping hands for creating environments where we could work on whatever we intended to make this project a complete success.

# ABSTRACT

Monitoring and modelling the ionosphere is one of the important aspect in GNSS (Global Navigation Satellite System). Among various factors that affect GNSS signals, ionosphere is one of the most influencing factors. This ionosphere keeps on varying with respect to time, day, location and seasons due to various forces from both top and bottom sides of ionosphere. Although dual frequency GNSS receivers can minimize the ionospheric error to greater extent, single frequency receivers fail to mitigate these errors and depends on different techniques or model estimations.

The present study investigates the interplanetary space-dependent drifts in the ionospheric irregularities which cause predominant ranging errors in the GPS signals for the region of Nepal. The development of GNSS, especially of GPS, has led to the operation of CORS (Continuous Operating Reference Stations) that acquire GPS signals without any interruption. Despite the availability of more than 60 CORS in Nepal our country still lacks its own regional ionospheric model and we have to rely on the Global Ionospheric Models.

So, this study has incorporated these CORS data from various stations in Nepal to make a tool that can generate the TEC map of our study area. For that, different data were used like Hatanaka files and navigation files from CORS available in UNAVCO website. Development of python tool was done in order to download, process and visualize the TEC maps from GPS observables. This tool also downloads the required DCB (Differential Code Bias) file from CODE and applies the necessary satellite bias corrections. For, the corrections of GPS data, the tool developed generates IONEX file. Also we have developed dynamic map showing variation of TEC over a day. Based upon our analysis, we found that the ionospheric activity is varying with solar activity and changing constantly over a period of time. However, distinct pattern of the ionospheric activity was hard to estimate perhaps due to limitation in data points over the study area and very simplistic approach of TEC estimation.

Hence, Nepal is in a need of its own regional ionospheric model and this tool and the result of this project will be helpful in order to study the space weather of the Himalayan region also the results of this project can be used in various other applications.


**Keywords: TEC maps, Ionosphere, GPS, CORS, IONEX**

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF EQUATIONS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| C/A | Course Acquistion |
| CODE | Center for Orbit Determination in Europe |
| CORS | Continuously Operating Reference Stations |
| ECEF | Earth Centered Earth Fixed Frame |
| ESA | European Space Agency |
| GIM | Global Ionospheric Maps |
| GNSS | Global Navigation Satellite System |
| GPS | Global Positioning System |
| IONEX | IONosphere map EXchange format |
| IPP | Ionospheric Pierce Point |
| IRI | International Reference Ionosphere |
| QZSS | Quasi-Zenith Satellite System |
| RINEX | Receiver Independent Exchange Format |
| RNSS | Regional Navigation Satellite System |
| SPIM | Standard Plasmasphere Ionosphere Model |
| STEC | Slant Total Electron Content |
| TEC | Total Electron Content |
| TECU | Total Electron Content Unit |
| UNAVCO | University NAVSTAR Consortium |

# 1. INTRODUCTION

## 1.1. Background

The ionosphere is the layer of the Earth's upper atmosphere containing high concentration of electrons and ions, mostly caused by solar radiation producing free electrons from the existing atoms of Earth's atmospheric gases (Panda, Gedam, & Rajaram, 2015). This layer of ionosphere extends from 50 km to more than 1000 km altitude above the earth's surface. They are able to reflect radio waves which plays vital role to create challenges in communication, navigation and surveillance. The free electrons in the ionosphere is quantified in terms of Total Electron Content (TEC) which is the total number of free electrons contained in a column with cross-sectional area of 1 square meter and its unit is TECU, where 1 TECU is $10^{16}$ electrons/$m^2$ (Schaer, 1999; Dach et. al., 2015). A change in TEC value of about 1 TECU can create a delay of about approximately 0.16 meters (Takahashi et al. 2016). The spatial and temporal behavior of TEC has a great coverage in regional and global level.

Monitoring the ionosphere is one of the important topic for GNSS (Global Navigation Satellite System). Among various factors that affect GNSS signals, ionosphere is one of the influencing factors. GNSS is common collective term to refer to satellite constellation system like GPS (Global Positioning System) owned by USA, GLONASS by Russia, Galileo by Europe and Beidou by China. Also there are RNSS systems such as IRHSS by India and QZSS by Japan. The use of these GNSS/RNSS systems has been highly emphasized in recent days for the purpose of high-precision positioning (Ansari and Park 2018). For satellite communication, we need frequencies which will penetrate the ionosphere in different to its concentration. Also as GNSS uses a signals with a higher frequency, ionosphere still bends and disturbs the signals which influences the position determination by reducing its precision up to several kilometers, that's why instead of using one frequency, GNSS uses two frequencies L1 and L2 where L1= 1575.42 MHz and L2= 1227.60 MHz. Each of these frequencies is altered by the ionosphere on its way to the surface of the earth. Due to dispersive property of ionosphere, these two frequencies travel at different velocities. For a GPS (Global Positioning System) signal, the ionosphere has a significant effect particularly for the single frequency users as they are affected to a greater extent. However, the varying nature of error with respect to time, day, location and seasons due to various forces from both top and bottom sides of ionosphere makes it more challenging to estimate (Sharma, Ansari, and Panda 2018). As for dual frequency GNSS receivers, these observations can be used to eliminate almost all of the ionosphere's effect.

The development of GNSS, especially of GPS, has led to the operation of continuous operating reference stations (CORS) that acquire GPS signals without any interruption (Volker SCHWIEGER, 2009). CORS are receiver stations established that can continuously receive GNSS data that can be useful for number of applications. In the context of Nepal, 60 CORS distributed all over Nepal can be found at UNAVCO DAI and data of these stations are available to use for free. So, the purpose of our project is at first to get all of the data and make use of all usable data in order to develop our TEC map. Our main theme is to determine TEC values from dual frequency GNSS observables above the region of Nepal by benefiting CORS which have uninterrupted data from the year January 2018 to current date in order to prepare the regional ionospheric map.

The daily TEC maps generated can then be used in correction while processing GPS data of single frequency receivers. It is estimated that the outcome of the project can be helpful in enhancing the accuracy of GPS survey works by making the use of data available freely. Having a regional ionospheric models will open up doors to new possibilities in the navigation science and engineering in Nepal.

## 1.2. Problem Statement

In the recent years, many research has been done in ionosphere for investigating the ionospheric TEC from regional or global network of ground based GNSS. In GNSS based positioning, ionosphere is one of the largest error sources causing a delay in the arrival of navigation signal. Further, the corresponding error keeps on varying with respect to time, day, location and seasons due to various forces from both top and bottom sides of ionosphere. Although dual frequency GPS receivers can minimize the ionospheric error to greater extent, single frequency receivers fail to mitigate these errors and depends on different techniques or model estimations. Nowadays, there exists several global and regional ionospheric models such as International Reference Ionosphere (IRI), Standard Plasmasphere Ionosphere Model (SPIM), Global Ionosphere Map (GIM) etc. However, reliability of these models is limited to the experimental datasets and techniques (Sharma, Ansari, & Panda, 2018).

Despite the availability of more than 60 CORS in Nepal our country still lacks its own regional ionospheric model and we have to rely on the Global Ionospheric Models. The global models are accurate and standard however in a study done by BaşçiFtçi et al. (2017), it was shown that as IRI model components are timely updated with better estimation capacity, values given by IRI model were significantly lower than that of regional model from the study and other global models also. So, there is a need as well as opportunity for the development of regional ionospheric model for Nepal.

Also most of the research regarding total electron content and preparation of ionospheric map require several mapping algorithms, which have been done in the software's like MATLAB, which are not free to use. As there are freely available software's like GOPI GPS, which are free to use however they are not open source. So we have developed a tool based on python which can be used for generating total electron content maps which is both free and open source. This will help the researchers in this sector to better understand the fundamentals of signal propagation through ionosphere and its dynamics.

## 1.3. Objectives

The following subsections highlight the main and sub objectives of the project.

### 1.3.1. Main Objective

The main objective of the project is to make the TEC map of our study area using the CORS data from various stations in Nepal.

### 1.3.2. Sub Objectives

To support the main objectives, it is divided into the following sub objectives.

- To identify the correct methods and ways of acquiring data, processing, storing and generating the outputs in standard forms to meet the main objective of the project.
- To develop an interactive python tool capable of generating the TEC maps and IONEX file based on user inputs.
- To prepare TEC map and IONEX file using the tool developed.

## 1.4. Scope of work

As mentioned in the objectives section, the main objective of our project is to develop a TEC map that serves as a means for determining the ionospheric behavior above the region of Nepal. This project studies how the ionosphere is structured above the atmosphere of Nepal. After the achievement of the objectives of this project, our work to have some impact on GNSS sector of Nepal and we will get our own regional ionospheric model. This model can be helpful for study of space whether of Nepal as well as IONEX file achieved can be used as correction model for data from GPS survey done in area inside the area of our study.

While discussing about the shortcomings the model will not be evenly reliable to be used for all parts of Nepal. This is due to the fact that many of the CORS in Nepal are in non-functional state and also that the CORS are not in a definite pattern. So the reliability of the model will heavily depend upon the density of functional CORS in the area.

## 1.5. Report Outline

The report of five chapters with some appendices is the output of this project. The list of chapters is given as follows.

**Chapter 1: Introduction**

In this chapter, it introduces the project. It also presents a basic overview of the project including objectives, sub objectives, as well as scope of work.

**Chapter 2: Literature Review**

This chapter gives a brief overview about theoretical/conceptual framework of the project including a brief summary about important contents of this project like ionosphere, regions of ionosphere, how TEC value is determined and basic overview of contents.

**Chapter 3: Methodology**

This chapter focuses to case study area and reports the methods carried out for the planning, data collection, data processing, and the methods involved while carrying out this project.

**Chapter 4: Results and Discussion**

This chapter shows the detailed explanation of the result obtained from our project and discusses on the assessment results and also discusses on the possibility regarding its use, expansion and other things.

**Chapter 5: Conclusion and Recommendations**

This final chapter includes the conclusion drawn out from the project along with some recommendations that helps us and others for future works.

# 2. LITERATURE REVIEW

## 2.1. Temperature and electron density profile of layers of atmosphere

Different parts of atmosphere have different characteristics based on their temperature and electron content. However while dealing with signal propagation to layers are significant which are troposphere and ionosphere (Memarzadeh 2009).Shown below is a diagrammatic representation of temperature and electron density profile of various layers of atmosphere.

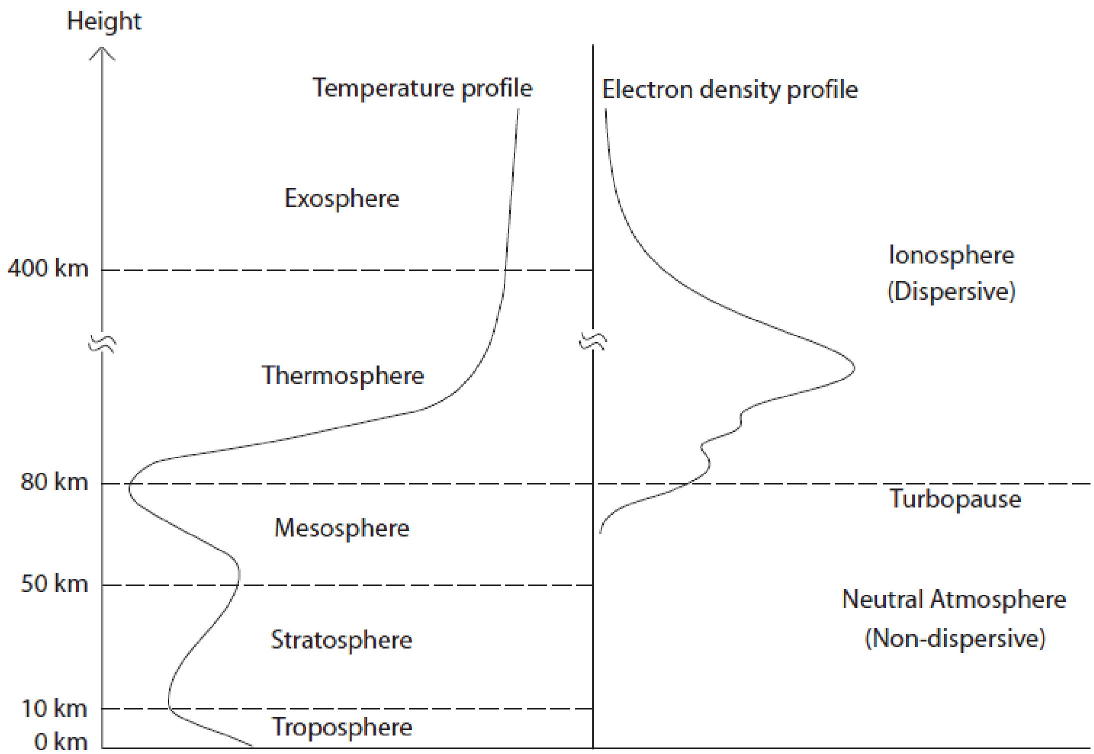



Figure 1: Temperature and Electron density profile of various layers of atmosphere

Source: (Memarzadeh 2009)

## 2.2. Vertical Profile of Ionosphere

Ionosphere is the layer of earth's atmosphere extending from 50 km to 1000 km where the ionosphere is ionized by solar radiation. This layer is the composition of D, E, F1 and F2 layers which are also called bottom side of ionosphere. Between F2 layer and upper boundary of ionosphere, it is known as topside of ionosphere. The electron density occurs maximum in this F2 layer due to more absorption of ultraviolet light and increase in atmospheric density as the altitude decreases.

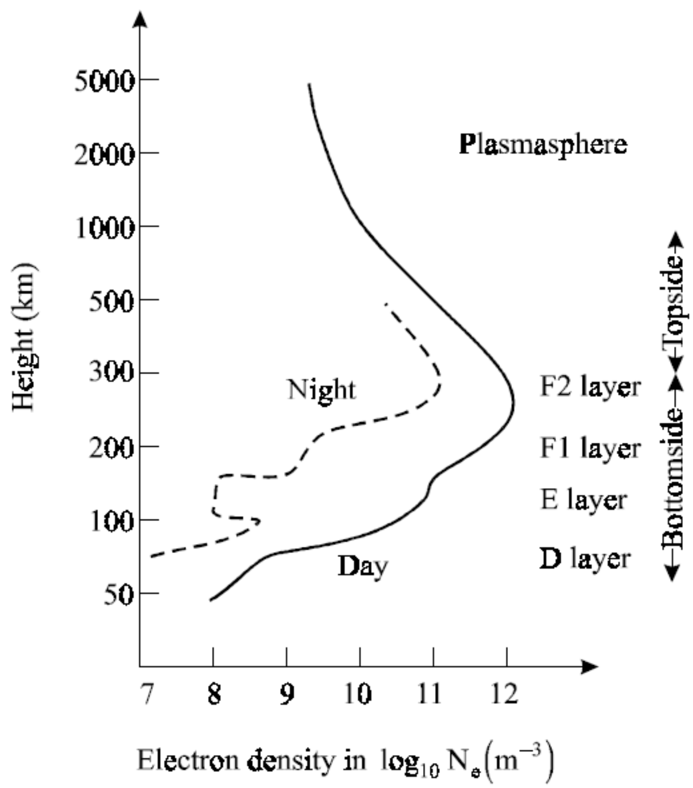



Figure 2: Typical vertical profile of ionosphere

Source: (Komjathy 1997)

### 2.2.1. The D layer

It is the lowest layer of the ionosphere which exists at altitude between 50 to 90 km. As it deflects shorter radio waves only, longer radio waves are not deflected much by this layer. By night, the electrons are attached to some molecules and atoms which results into formation of negative ions due to which it may disappear. However, in daylight, it re-appears again which results into the diurnal variation in electron density.

### 2.2.2. The E layer

This layer is thin and dense which exists at altitude in between 90 km to 150 km above the ground surface (BaşçiFtçi, İNal, Yildirim, & Bülbül, 2017). Its behavior depends on the solar activity and sun zenith angle. This layer doesn’t completely vanish at night however it is assumed that the electron density of E layer is zero during night time for practical purposes. Global radio communication is possible due to E region which remain charged at night.

### 2.2.3. The F1 layer

This layer is located at an altitude of 150 km above the ground surface. The F layer is mainly divided into these two layers; F1 and F2. 10 % of GNSS signal’s delay at ionosphere layer is caused by F1 layer (BaşçiFtçi, İNal, Yildirim, & Bülbül, 2017). This layer is observed only during the day since the electron densities are controlled by the zenith angle of sun.

### 2.2.4. The F2 layer

Located in between 200 km to 1000 km above the ground surface, the F2 layer is most important for GNSS measurements. Hence, this layer is much unstable at equatorial region and during night time, the electron density is more than noon time.

## 2.3. Major Geographic reasons of Ionosphere

Ionosphere can further be divided into several regions according to the latitude regions of the earth. They are equatorial, mid-latitude and high-latitude regions. The characteristics of these layers are described below:

### 2.3.1. Equatorial Region

This is the region where the highest value of peak electron density occurs. The signal amplitude and phase are changed frequently in this region. The equatorial region lies between 0 to 20 degrees’ geomagnetic latitude which causes the decrease in electron density at geomagnetic equator also termed as equatorial anomaly. Daily equatorial anomaly starts in local time between 9:00 – 10:00 and reaches maximum at 14:00 – 15:00 (Gizawy, 2003). This is also the region where our study area is located.

### 2.3.2. Mid-Latitude Region

Mid latitude region lies between 20 to 60 degrees’ geomagnetic latitude. Most of the research and observations have been done considering this region due to the fact that most of the countries having ionospheric sensing instruments are located in this region.

### 2.3.3. High-Latitude Region

This region lies between 60 to 70 degrees’ geomagnetic latitude. In this region, collisional ionization is takes place which is another source of ionization. This is because the geomagnetic field lines are nearly vertical in this region leading to charged particles descending to E layer altitudes (about 100 km).

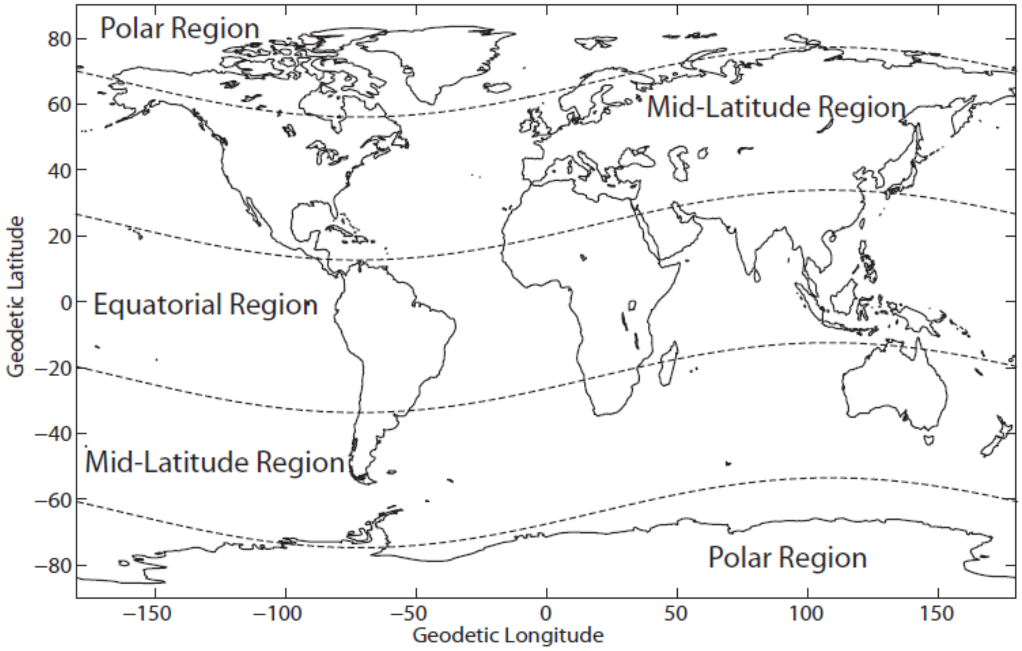



Figure 3: Geographic regions of ionosphere according to geodetic latitudes

Source: (Memarzadeh 2009)

## 2.4. Ionospheric Total Electron Content (TEC)

Total electron content is a derived parameter from electron density and it is defined as the line integral of electron density on a ray path.

$$\text{TEC} = \int n_e d\rho$$

Equation 1: Total Electron Content

where $n_e$ is the electron density or number density of electron and $\rho$ is mass density of Earth's atmosphere.

TEC corresponds to the total number of electrons in a cylindrical tube with 1 $m^2$ cross section. It is expressed in terms of TECu where 1 TECu refers to $10^{16}$ electrons/$m^2$ (Psiaki, Bust, and Mitchell 2015). TEC contains the projection of ionospheric electron distribution and it can be used to model, reconstruct and predict ionospheric variability. The wide spread use of GPS dual frequency receivers provides a cost-effective solution to estimate TEC. Using TEC values, short and long term variations in the ionosphere and ionospheric disturbances can be analyzed. It depends upon the satellite elevation angle. As the length of signal path from the ionosphere varies with satellite position, lower elevation results in higher TEC due to longer signal path. The visual representation of TEC values is demonstrated below:

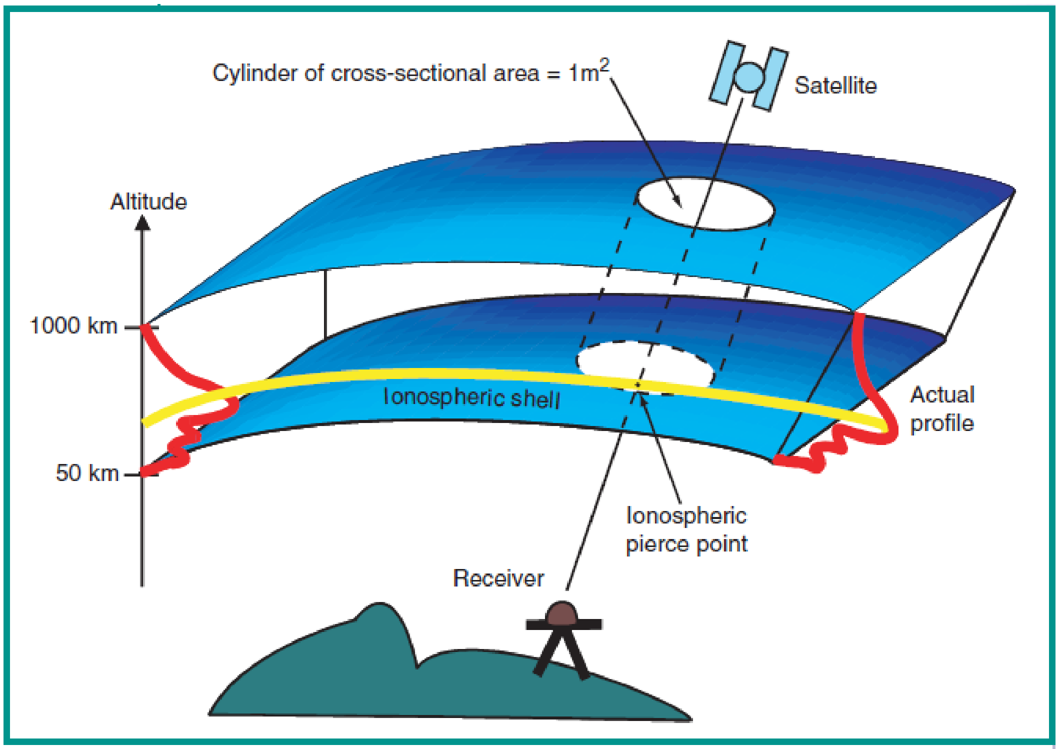



Figure 4: Visual representation of TEC values

Source: (Fedrizzi et al. 2001)

### 2.4.1. Single Layer Ionospheric approximation

Electron density cannot be measured directly. Indirect methods include mathematical modelling and measurement uncertainties and errors. For the simplicity of ionosphere modelling, ionosphere is considered as a single thin layer at an altitude of 400 km above the earth's surface surrounding the earth at a fixed height for which all the free electrons in the ionosphere are assumed to be concentrated in this single layer. There are different mapping functions available for finding vertical TEC values from slant TEC and vice-versa.

- **Ionospheric pierce point**

The signal transmitted from the satellite to the receiver crosses the ionospheric shell in the so-called ionospheric pierce point (IPP). The zenith angle at the IPP is $z'$ and the signal arrives at the receiver with zenith angle $z$. Here $R$ is the mean Earth radius, $H$ is the mean height of the ionosphere shell (Wielgosz, Grejner-Brzezinska, & Kashani, 2003).

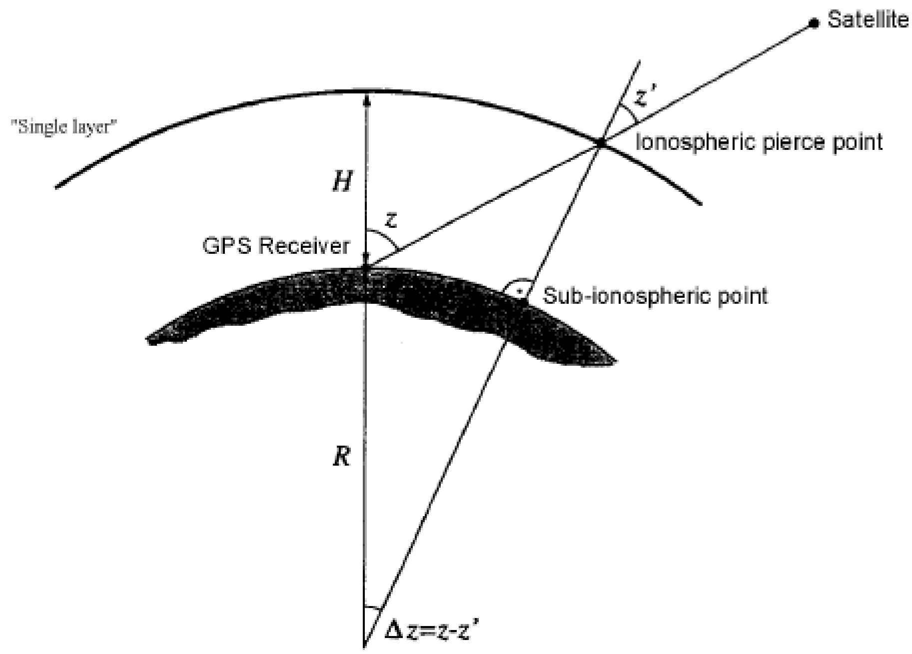



Figure 5: Single layer model of the ionosphere

Source: (Wielgosz, Grejner-Brzezinska, & Kashani, 2003)

## 2.5. Obtaining TEC values with GNSS measurements

For understanding the structure of ionosphere, obtaining TEC values from GNSS signals is fast and cheap technique. From the TEC value obtained from GNSS signals, we can develop an improved regional ionospheric model which can be used for single frequency GPS users for analyzing and correcting ionospheric effect. Now, for easy exchange of collected raw GNSS data, GNSS uses different file formats which are described below:

### 2.5.1. RINEX (Receiver Independent Exchange format)

For easy exchange of the GPS data and for processing in various software, Astronomical Institute of the University of Berne is 1989 developed this file format which allows the usage of measurements generated in the receiver, and their further analysis (e.g. development of better ionospheric models). This file format consists of several ASCII file types:

| File Types | All platforms | UNIX | VMS | DOS |
|---|---|---|---|---|
| | uncompressed | | compressed | |
| Obs Files | .yyO | .yyO.Z | .yyO_Z | .yyY |
| Obs Files (Hatanaka compressed) | .yyD | .yyD.Z | .yyD_Z | .yyE |
| GPS Nav Files | .yyN | .yyN.Z | .yyN_Z | .yyX |
| GLONASS Nav File | .yyG | .yyG.Z | .yyG_Z | .yyV |
| Galileo Nav File | .yyL | .yyL.Z | .yyL_Z | .yyT |
| GEO Nav Files | .yyH | .yyH.Z | .yyH_Z | .yyU |
| GEO SBAS Broadcast Files (sep. doc.) | .yyB | .yyB.Z | .yyB_Z | .yyA |
| Met Data Files | .yyM | .yyM.Z | .yyM_Z | .yyW |
| Clock Files (see sep.doc.) | .yyC | .yyC.Z | .yyC_Z | .yyK |

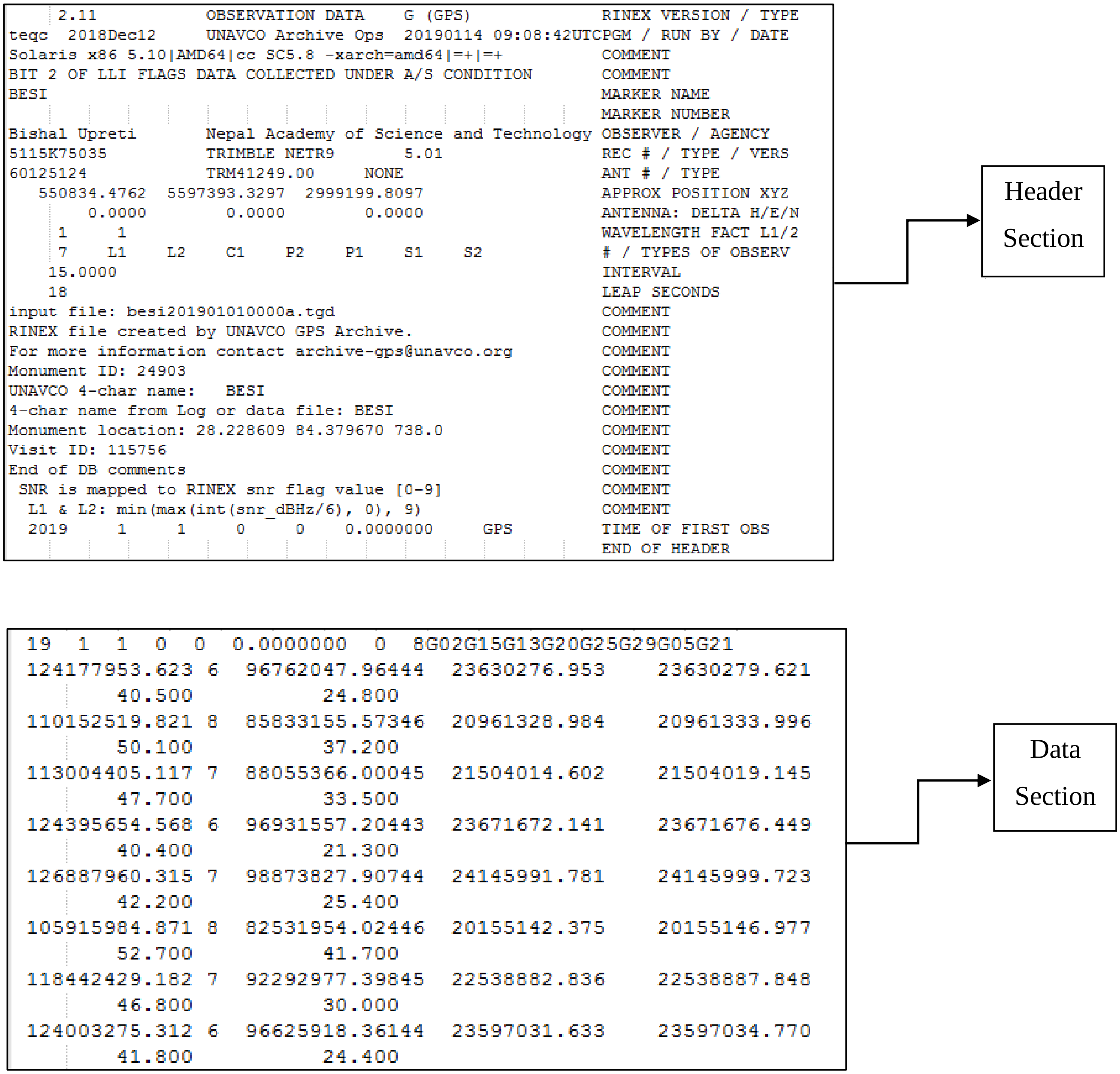
Figure 6: Rinex ASCII file types

Source: (Br, 2013)

Among these file types, we need mainly navigation message file and observation data files.

- **Observation Data file**

These are the data files with extension named ".yyo"(yy= year). Each of this file type contains two section: Header and Data section. The header section consists of global information for the entire file and data consists of the data taken from the respective station. The data taken from one station have one observation file.



```
     2.11           OBSERVATION DATA    G (GPS)             RINEX VERSION / TYPE
teqc  2018Dec12     UNAVCO Archive Ops  20190114 09:08:42UTCPGM / RUN BY / DATE
Solaris x86 5.10|AMD64|cc SC5.8 -xarch=amd64|=+|=+          COMMENT
BIT 2 OF LLI FLAGS DATA COLLECTED UNDER A/S CONDITION       COMMENT
BESI                                                        MARKER NAME
                                                            MARKER NUMBER
Bishal Upreti       Nepal Academy of Science and Technology OBSERVER / AGENCY
5115K75035          TRIMBLE NETR9       5.01                REC # / TYPE / VERS
60125124            TRM41249.00     NONE                    ANT # / TYPE
   550834.4762  5597393.3297  2999199.8097                  APPROX POSITION XYZ
        0.0000        0.0000        0.0000                  ANTENNA: DELTA H/E/N
     1     1                                                WAVELENGTH FACT L1/2
     7    L1    L2    C1    P2    P1    S1    S2            # / TYPES OF OBSERV
    15.0000                                                 INTERVAL
    18                                                      LEAP SECONDS
input file: besi201901010000a.tgd                           COMMENT
RINEX file created by UNAVCO GPS Archive.                   COMMENT
For more information contact archive-gps@unavco.org         COMMENT
Monument ID: 24903                                          COMMENT
UNAVCO 4-char name:   BESI                                  COMMENT
4-char name from Log or data file: BESI                     COMMENT
Monument location: 28.228609 84.379670 738.0                COMMENT
Visit ID: 115756                                            COMMENT
End of DB comments                                          COMMENT
 SNR is mapped to RINEX snr flag value [0-9]                COMMENT
  L1 & L2: min(max(int(snr_dBHz/6), 0), 9)                  COMMENT
  2019     1     1     0     0    0.0000000     GPS         TIME OF FIRST OBS
                                                            END OF HEADER
```

```
 19  1  1  0  0  0.0000000  0  8G02G15G13G20G25G29G05G21
 124177953.623 6  96762047.96444  23630276.953    23630279.621
        40.500           24.800
 110152519.821 8  85833155.57346  20961328.984    20961333.996
        50.100           37.200
 113004405.117 7  88055366.00045  21504014.602    21504019.145
        47.700           33.500
 124395654.568 6  96931557.20443  23671672.141    23671676.449
        40.400           21.300
 126887960.315 7  98873827.90744  24145991.781    24145999.723
        42.200           25.400
 105915984.871 8  82531954.02446  20155142.375    20155146.977
        52.700           41.700
 118442429.182 7  92292977.39845  22538882.836    22538887.848
        46.800           30.000
 124003275.312 6  96625918.36144  23597031.633    23597034.770
        41.800           24.400
```



Figure 7: Rinex observation data file

## Rinex observation file (Header Section):

Table 1: Rinex observation file header section

| | |
|---|---|
| L1 | Phase measurements on L1 |
| L2 | Phase measurements on L2 |
| C1 | Pseudorange using C/A code on F1 |

| | |
|---|---|
| P2 | Pseudorange using P-code on F2 |
| P1 | Pseudorange using P-code on frequency 1 |
| S1 | Signal strength on F1 (ranging from 1-9) |
| S2 | Signal strength on F2 (ranging from 1-9) |
| PGM / RUN BY / DATE | Program, Agency, date of creating the file |
| REC # / TYPE / VERS | Receiver Number, type, version |
| ANT # / TYPE | Antenna Number, TYPE |
| APPROX POSITION XYZ | Approximation marker position (in WGS84) |
| ANTENNA: DELTA H/E/N | Antenna height, Eccentricities of antenna centre relative to marker in east and north (in meters) |
| WAVELENGTH FACT L1/2 | Wavelength factors for L1 and L2 |
| # / TYPES OF OBSERV | Number of observation types, observation types |
| TIME OF FIRST OBS | Time of first observation record |

**Rinex observation file (Data Section):**

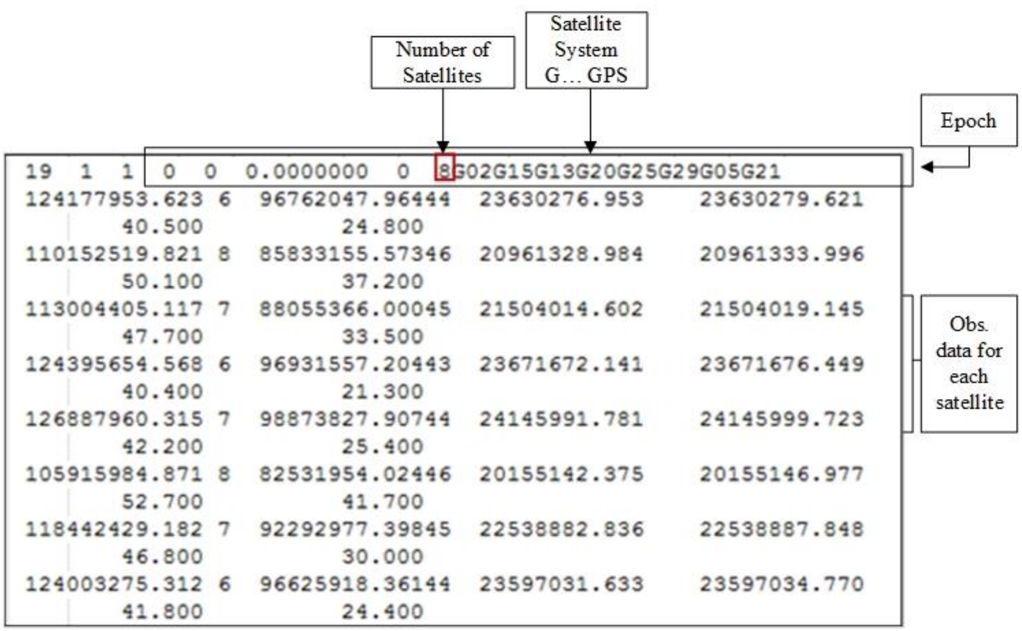



Figure 8: Rinex observation file data section

- **Navigation message file**

These are the data files with extension named ".yyn"(yy = year). Similar to observation file, this file type also contains header and data section. These files contain velocity, clock information, position for all the satellites of GPS constellation.

```
    2.11           N: GPS NAV DATA                         RINEX VERSION / TYPE
teqc  2018Dec12     UNAVCO Archive Ops  20190114 09:08:42UTCPGM / RUN BY / DATE
Solaris x86 5.10|AMD64|cc SC5.8 -xarch=amd64|=+|=+         COMMENT
    7.4506D-09 -1.4901D-08 -5.9605D-08  1.1921D-07          ION ALPHA
    9.2160D+04 -1.1469D+05 -1.3107D+05  7.2090D+05          ION BETA
    9.313225746155D-10 7.993605777301D-15   319488     2034 DELTA-UTC: A0,A1,T,W
                                                            END OF HEADER
 2 19  1  1  0  0  0.0-1.009884290397D-04-1.023181539495D-11 0.000000000000D+00
    4.300000000000D+01-9.687500000000D+00 4.555904057405D-09-1.634266936772D+00
   -3.352761268616D-07 1.868473272771D-02 9.741634130478D-06 5.153622894287D+03
    1.728000000000D+05 4.153698682785D-07-4.135489163952D-01 3.501772880554D-07
    9.532735706191D-01 1.859375000000D+02-1.797359936056D+00-7.906757919633D-09
    3.032269163325D-10 1.000000000000D+00 2.034000000000D+03 0.000000000000D+00
    2.000000000000D+00 0.000000000000D+00-2.048909664154D-08 4.300000000000D+01
    1.656180000000D+05 4.000000000000D+00
```

Figure 9: Rinex navigation message file

**Navigation message file (Header Section)**

Table 2: Navigation message file header section

| | |
|---|---|
| RINEX VERSION / TYPE | Format version (2.11), file type ('N' for navigation data) |
| PGM / RUN BY / DATE | Name of program creating current file, name of agency creating current file, date of file creation |
| ION ALPHA | Ionospheric parameters A0-A3 of almanac |
| ION BETA | Ionosphere parameters B0-B3 of almanac |
| DELTA-UTC: A0,A1,T,W | Almanac parameters to compute time in UTC<br>A0, A1: terms of polynomial<br>T: reference time for UTC data<br>W: UTC reference week number. |

**Navigation message file (Data Section)**

| Parameter | Explanation |
|---|---|
| $t_{oe}$ | Ephemerides reference epoch in seconds within the week |
| $\sqrt{a}$ | Square root of semi-major axis |
| $e$ | Eccentricity |
| $M_o$ | Mean anomaly at reference epoch |
| $\omega$ | Argument of perigee |
| $i_o$ | Inclination at reference epoch |
| $\Omega_0$ | Longitude of ascending node at the beginning of the week |
| $\Delta n$ | Mean motion difference |
| $\dot{i}$ | Rate of inclination angle |
| $\dot{\Omega}$ | Rate of node's right ascension |
| $c_{uc}, c_{us}$ | Latitude argument correction |
| $c_{rc}, c_{rs}$ | Orbital radius correction |
| $c_{ic}, c_{is}$ | Inclination correction |
| $a_0$ | SV clock offset |
| $a_1$ | SV clock drift |
| $a_2$ | SV clock drift rate |

Figure 10: Navigation message file data section

Source: (ESA, 2011)

### 2.5.2. IONEX format files

This simple file format to store and exchange ionospheric data was purposed in 1996 at the IGS workshop and from then has been standard for representing ionospheric maps and using them for correction purpose.

Table 3: IONEX file header section description

| | |
|---|---|
| IONEX VERSION / TYPE\| | format version (1.1)<br>File type ('I' for Ionosphere maps)<br>Satellite system or theoretical model:<br>This record has to be the first one in an IONEX file! |
| PGM / RUN BY / DATE | Name of program creating current file |

| | |
|---|---|
| | Name of agency creating current file<br>Date and time of file creation |
| EPOCH OF FIRST MAP | Time of first TEC map |
| EPOCH OF LAST MAP | Time of last TEC map |
| INTERVAL | Time interval between TEC maps |
| # OF MAPS IN FILE | Number of epochs in TEC map |
| MAPPING FUNCTION | Mapping function used to obtain the VTEC from STEC values |
| ELEVATION CUTOFF | Elevation cutoff for the Satellites with respect to receivers. |
| OBSERVABLES USED | One line description of the observables used to generate TEC maps |
| BASE RADIUS | Radius of Earth used |
| MAP DIMENSION | The dimension of TEC map. It is either 2D or 3D |
| HGT1 / HGT2 / DHGT | Lower and upper height values along with the height interval |
| LAT1 / LAT2 / DLAT | Latitude extent values along with the latitude interval |
| LON1 / LON2 / DLON | Longitude extent values along with the longitude interval |

Header section is marked with END OF HEADER and then starts the data section of the map.

A sample of data section of the map is as shown in picture below:

```
     1                                                        START OF TEC MAP
  2019     1     1     0     0     0                        EPOCH OF CURRENT MAP
    87.5-180.0 180.0   5.0 450.0                            LAT/LON1/LON2/DLON/H
    8    8    8    8    8    8    7    7    7    7    7    7    6    6    6    6
    5    5    5    4    4    4    4    3    3    3    3    3    3    3    3    3
    3    3    3    3    3    3    4    4    4    4    4    5    5    5    5    5
    6    6    6    6    6    6    6    6    6    6    7    7    7    7    7    7
    7    7    7    7    7    7    8    8    8
    85.0-180.0 180.0   5.0 450.0                            LAT/LON1/LON2/DLON/H
   11   11   11   11   11   11   11   11   11   10   10   10    9    9    8    7
    7    6    5    4    4    3    2    2    1    1    1    0    0    0    0    1
    1    1    2    2    2    3    3    4    5    5    6    6    7    7    7    8
    8    8    8    8    8    8    8    8    8    8    8    8    8    8    8    8
    8    9    9    9    9   10   10   10   11
```

Figure 11: Data section of IONEX file

# 3. METHODOLOGY

## 3.1. Study Area

For monitoring the ionospheric activity, whole region of Nepal (28.3949º N, 84.1240º E) covering an area of 1,47,181 square kilometers was taken as study area. According to the regions of ionosphere, it lies in the equatorial region (0 to 20 degrees' geomagnetic latitude) which has the highest value of peak-electron density.

Below map shows the positions of the stations whose data we will be using in order to get our result. For this purpose, the ground based dual-frequency CORS situated along the different parts of Nepal were considered as a data source.

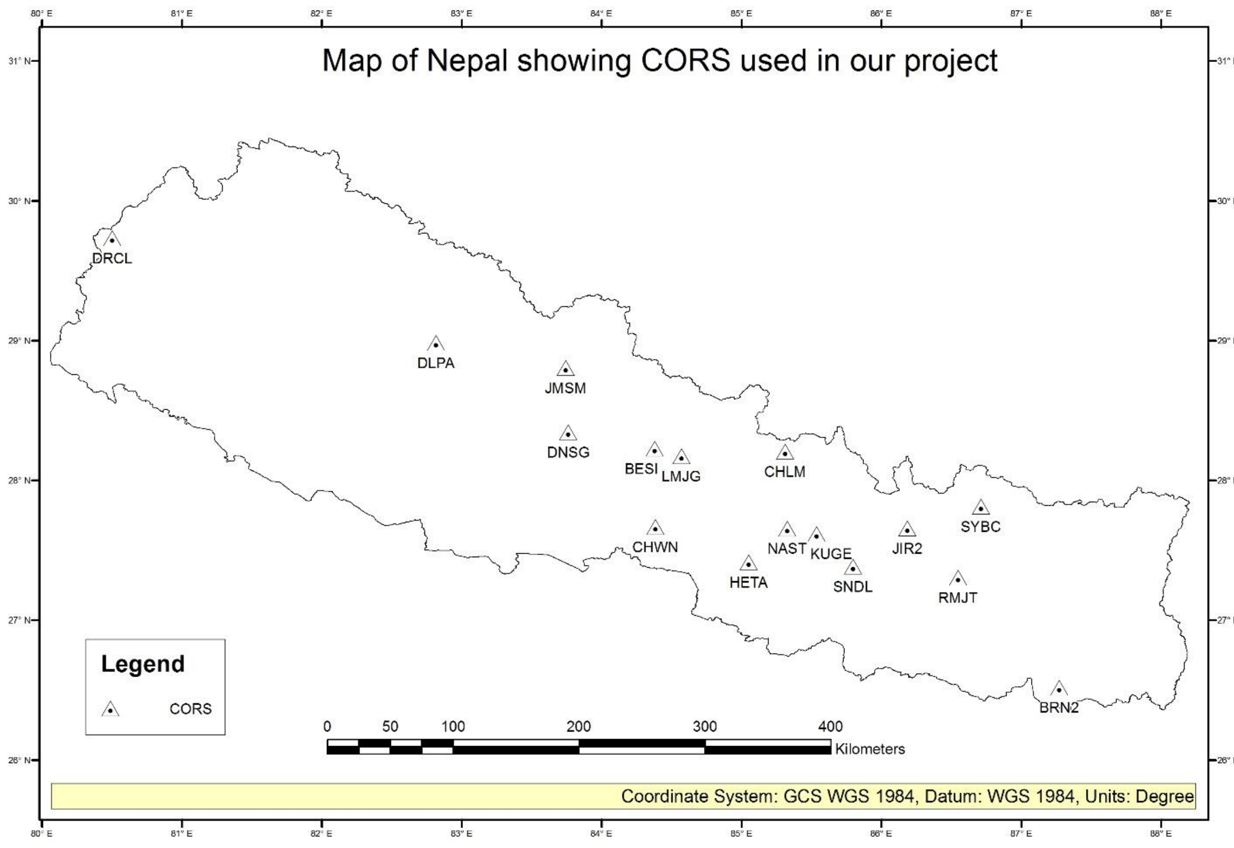



Figure 12: Project Study Area

## 3.2. Study Method/Workflow

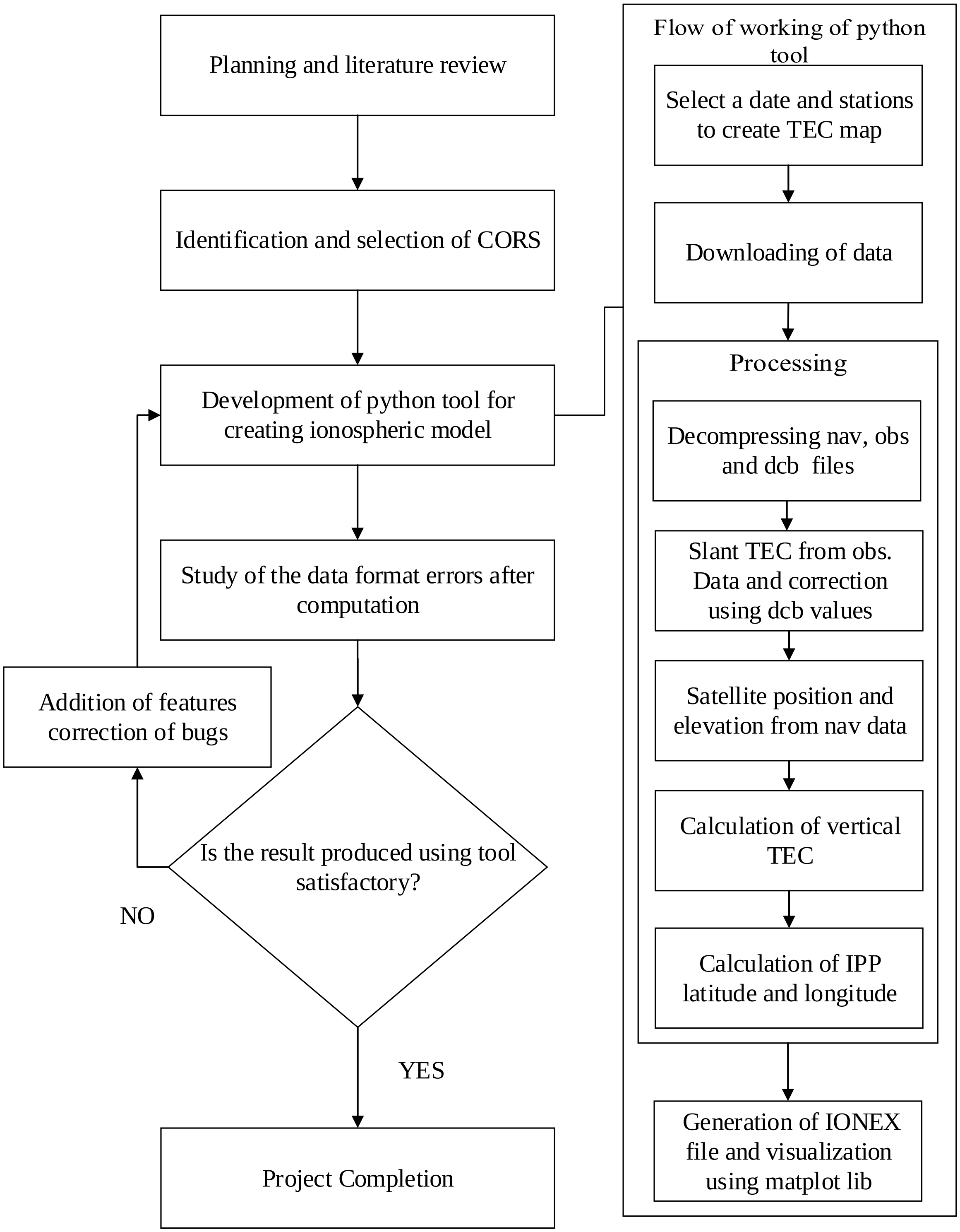



Figure 13: Flow diagram of project

## 3.3. Data Acquisition

### 3.3.1. Hatanaka Files

These are the files written and maintained by Yuki Hatanaka. These files contain GPS data in a lossless compression format. The "compact" RINEX files, which are generated by a two-step procedure (actual Hatanaka compression plus standard UNIX compression), are nearly 8 times smaller than the original ASCII RINEX files, yielding a gain of about 2.5 in disk space and transmission times compared to standard compressed files (Gurtner, 1998). These files were downloaded from the UNAVCO website. UNAVCO has been using this compression strategy for all of its RINEX observation files. Each file is UNIX-compressed, adding a '.Z' extension to the filename. Later, this file was decompressed to an ordinary RINEX observation file with the help of 'crx2rnx' software.

The following table shows the list of stations used in our project.

Table 4: List of CORS useful for our project

| 4chID | Interval | Name | Lat | Lon | Earliest data |
|---|---|---|---|---|---|
| DLPA | 15.0 sec | Dolpa | 28.9837 | 82.8179 | 2007 May 10 00 |
| DNSG | 15.0 sec | Dansing | 28.3451 | 83.7635 | 2012 Apr 29 10:30 |
| JMSM | 15.0 sec | Jomsom | 28.8053 | 83.7433 | 2004 May 02 11 |
| BESI | 15.0 sec | Besisahar | 28.2286 | 84.3797 | 2008 Jun 28 11:56 |
| CHWN | 15.0 sec | Chitwan | 27.6682 | 84.3853 | 2011 Mar 28 08 |
| LMJG | 15.0 sec | Lamjung | 28.1741 | 84.5734 | 2011 Apr 20 08:02 |
| CHLM | 15.0 sec | Chilime | 28.2072 | 85.3141 | 2004 Mar 31 08 |
| HETA | 15.0 sec | Hetuada GoN Fish Farm | 27.4149 | 85.0516 | 2015 Jun 11 05:44 |
| NAST | 15.0 sec | NAST_SciTec_2013 | 27.6567 | 85.3277 | 2014 Jan 18 00:00 |
| KUGE | 15.0 sec | KUGE_NGN_NEP2018 | 27.6184 | 85.5385 | 2018 Jun 07 04:52 |
| SNDL | 15.0 sec | Sindhuli | 27.3848 | 85.7989 | 2011 Apr 05 12:09 |
| JIR2 | 15.0 sec | JIR2 | 27.6571 | 86.187 | 2015 May 25 02 |
| RMJT | 15.0 sec | Rumjartar | 27.3051 | 86.55 | 2009 Nov 05 09 |
| SYBC | 15.0 sec | Syangboche | 27.8142 | 86.7125 | 2008 Oct 04 04:34 |
| BRN2 | 15.0 sec | Biratnagar 2 | 26.5197 | 87.2722 | 2009 May 21 06 |
| DRCL | 15.0 sec | Darchula | 29.7338 | 80.5009 | 2008 Mar 13 11:01 |

### 3.3.2. Navigation Files

These files contain satellite ephemeris, to calculate the elevation & azimuth angles of satellite; required in the conversion of slant to vertical TEC. These files are also zipped in ".Z" format and need to be decompressed in order to be used for the computation purpose.

### 3.3.3. DCB Files

These files contain monthly satellite bias values for each satellite. These files can be downloaded from the CODE's website using the FTP login anonymously. The observables used to get the TEC values in our project are C1 and P2. So the bias files for P1C1 and P1P2 are added in order to get the bias value for each satellite. The value is then converted to TECU using a conversion factor which is mentioned in detail in data processing section of this report.

## 3.4. Data Preparation

The downloaded and zipped files are then decompressed using the python library unlzw into the hatanaka and navigation files. Nav files are easily readable in python script using the georinex library however hatanaka files are needed at first to convert obs files to be able to load using python script and use its data in the python file. For the DCB files same library is used to unzip the files but no external libraries were available to specifically read them. Hence python code was used to convert the values in the file to a python dictionary. All the code used is included in the Appendix section of this report.

## 3.5. Data Processing

The obs and nav files are then read using georinex python library which converts the files into the pandas data frame and makes it easy for data accessing and manipulating. At first the computation is done for the STEC values using the observations of the obs file.

$$STEC = \frac{1}{40.3}\left({}^{1}\!/_{f_1^2} - {}^{1}\!/_{f_2^2}\right)(P_2 - P_1)$$

Equation 2: Slant TEC equation

Where, $P_1$ and $P_2$ are pseudorange values for frequency $f_1$ and $f_2$ respectively. Values for $f_1$ and $f_2$ are known for a satellite system which in our case is GPS.

The obtained STEC value is not from DCB error which is accountable to a large fluctuation in the TEC values thus it needs to be corrected using the monthly DCB values obtained from the CODE.

$$STEC_{corrected} = STEC_{computed} + Bias * 2.85$$

Equation 3: STEC correction using DCB values

Here, $STEC_{computed}$ is the value obtained from equation 2 and the Bias values are obtained from the DCB files, 2.85 is the factor to convert the DCB values in NS to TECU (Mylnikova, Yasyukevich, Kunitsyn, & Padokhin, 2015) in order to add that to the STEC. Also this thing should be noted that correction only for satellite bias are applied as the data of same for receivers was not available.

After the computation of the STEC values we need to calculate the satellite elevation. For satellite elevation we need values for the position of receiver and satellite. The receiver position is known from the observation file itself and the satellite position needs to be computed from the data of the nav file. Following is the algorithm derived from (ESA, 2011) used to calculate the satellite position from the nav data.

- Compute the time $t_k$ from the ephemerides reference epoch $t_{oe}$ ($t$ and $t_{oe}$ are expressed in seconds in the GPS week):

$$t_k = t - t_{oe}$$

Equation 4: time from ephemerides reference epoch

If $t_k >$ 302400 sec, subtract 604800 sec from $t_k$. If $t_k <$ -302400 sec, add 604800 sec.

- Compute the mean anomaly for $t_k$,

$$M_k = M_0 + (\sqrt{\mu}/\sqrt{a^3} + \Delta\mathrm{n})t_k$$

Equation 5: Mean anomaly equation

- Solve (iteratively) the Kepler equation for the eccentricity anomaly $E_k$:

$$M_k = E_k - e \sin E_k$$

Equation 6: Kepler equation for the eccentricity anomaly

- Compute the true anomaly $V_k$:

$$v_k = \arctan(\frac{\sqrt{1-e^2}\sin E_k}{\cos E_k - e})$$

Equation 7: True anomaly equation

- Compute the argument of latitude $u_k$ from the argument of perigee $\omega$, true anomaly $v_k$ and corrections $c_{uc}$ and $c_{us}$:

$$u_k = \omega + v_k + c_{uc}cos2(\omega + v_k) + c_{us}sin2(\omega + v_k)$$

Equation 8: arguement of latitude equation

- Compute the radial distance $r_k$, considering corrections $c_{rc}$ and $c_{rs}$:

$$r_k = a(1 - ecosE_k) + C_{rc}cos2(\omega + v_k) + C_{rs}sin2(\omega + v_k)$$

Equation 9: Radial distance computation

- Compute the inclination $i_k$ of the orbital plane from the inclination $i_o$ at reference time $t_{oe}$, and corrections $c_{ic}$ and $c_{is}$:

$$i_k = i_0 + i.t_k + c_{ic}cos2(\omega + v_k) + c_{is}sin2(\omega + v_k)$$

Equation 10: Inclination of orbital plane

- Compute the longitude of the ascending node $\lambda_k$ (with respect to Greenwich). This calculation uses the right ascension at the beginning of the current week ($\Omega_o$), the correction from the apparent sidereal time variation in Greenwich between the beginning of the week and reference time $t_k = t - t_{oe}$, and the change in longitude of the ascending node from the reference time $t_{oe}$:

$$\lambda_k = \Omega_0 + (\Omega - \omega_E)t_k - \omega_E t_{oe}$$

Equation 11: Longitude of ascending node

- Compute the coordinates in TRS frame, applying three rotations (around $u_k$, $i_k$ and $\lambda_k$):

$$\begin{matrix} X_k \\ Y_k \\ Z_k \end{matrix} = R_3(-\lambda_k)R_1(-i_k)R_2(-u_k) \begin{matrix} r_k \\ 0 \\ 0 \end{matrix}$$

Equation 12: Coordinate in TRS frame applying three rotations

where $R_1$ and $R_3$ are the rotation matrices defined in Transformation between Terrestrial Frames. Description of the parameters used in all the equations for the algorithm above are already presented in the literature review section.

Now after having the Position of satellite and receiver we need to have elevation angle and azimuth of the satellite with respect to the receiver. For this purpose, library named pymap3d was used that is able to do the conversion taking the position of satellite and receiver as arguments.

Now the conversion of STEC to VTEC is carried out by the use of mapping function given as:

$$VTEC = STEC * \sqrt{1 - (\frac{R\cos(e)}{R+h})^2}$$

Equation 13: Conversion of STEC to VTEC by mapping function

Here, R is the radius of earth, h is the height of IPP and e is the elevation angle.

Now for the position at the IPP at which the VTEC is estimated is found as:

$$\phi_{IPP} = \arcsin(\sin(\phi)\cos(p) + \cos(\phi)\sin(p)\,coa(a))$$

$$\lambda_{IPP} = \lambda + (\sin(p)\sin(a)\,/\cos(\phi))$$

$$here, p = \frac{\Pi}{2} - e - \arcsin(\frac{R\cos(e)}{R+h})$$

Equation 14: Estimation of VTEC for the position at IPP

Here, R is the radius of earth, h is the height of IPP and (e, a) are the elevation angle and azimuth of satellite, ($\phi$, $\lambda$) are receiver position.

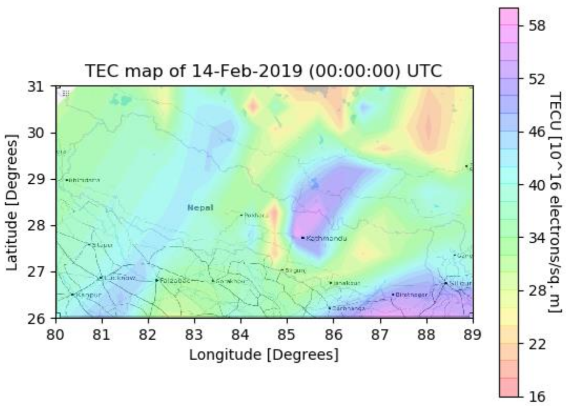
### 3.6. Visualization and Output

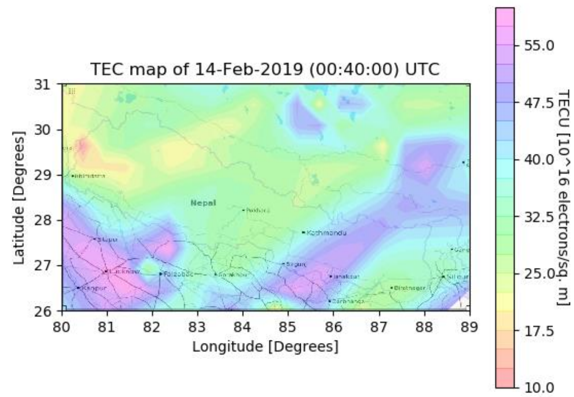
The VTEC value is represented in the form of colored contour map with position of IPP being the positional value. Base map from Openstreet maps is used in order to enhance the visualization. The interpolation used is the linear one done using the griddata function of python library scipy.interpolate. The library used for visualization is matplotlib.pyplot along with some other matplotlib libraries for the purpose of animation. The IONEX file is generated alongside the visualization which can then be used to apply the corrections to the GPS data. It also follows the same principal of interpolation and basically it is same as the visualized data but formatted in special way as per IGS guidelines so that it can be standard and can be exchanged and used by anyone.

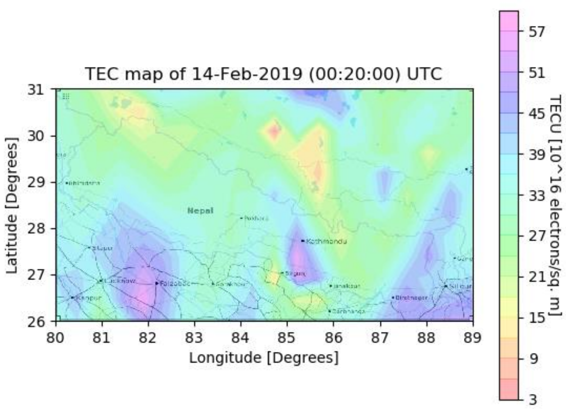
# 4. RESULTS AND DISCUSSION

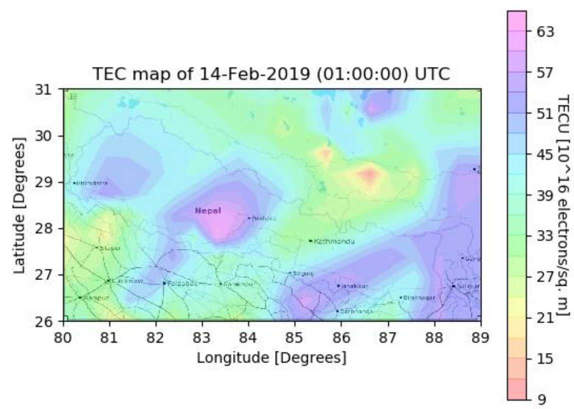
Below are the maps generated using the python tool that we developed. These maps are at a difference of 20 minutes for the date of 14th February 2019. The details of the working of the tool developed is elaborated in further part of this report.

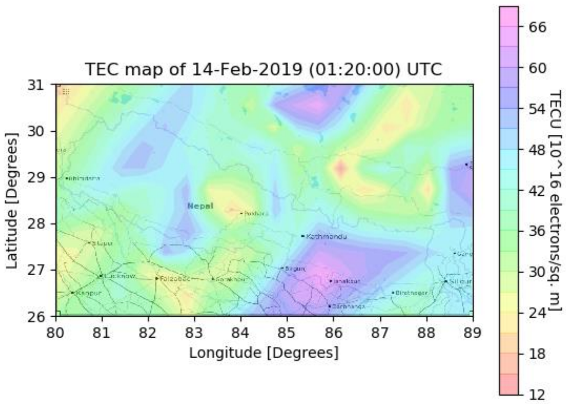
This is only a sample and desired output for any day can be easily generated using the tool. This was done as generating TEC maps for a large period of time would consume huge amount of space and take a lot of time.

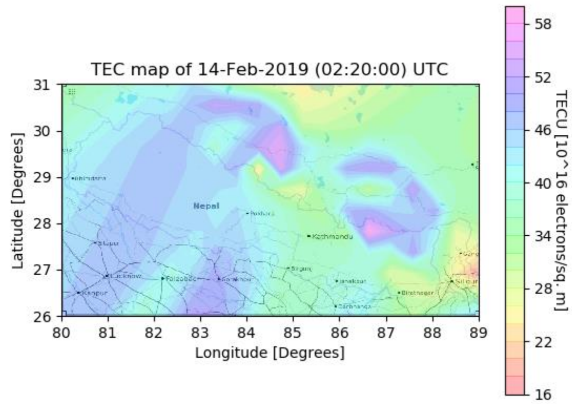


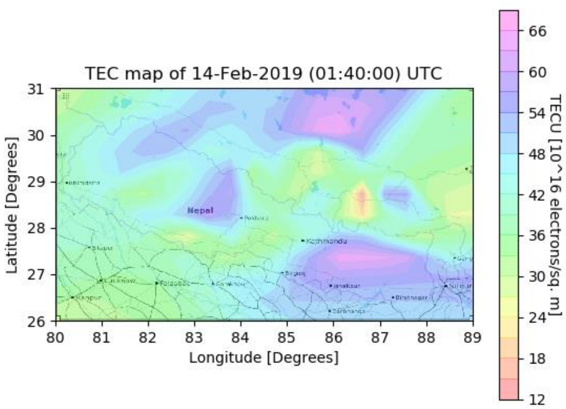



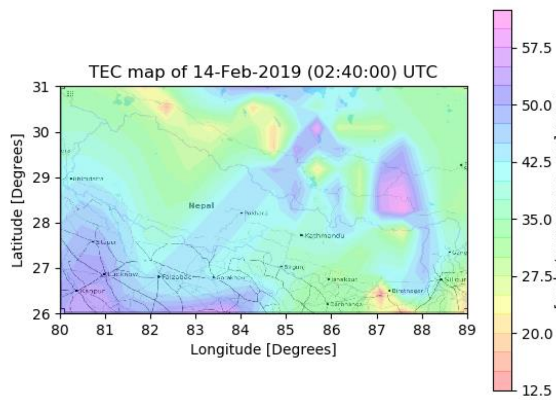



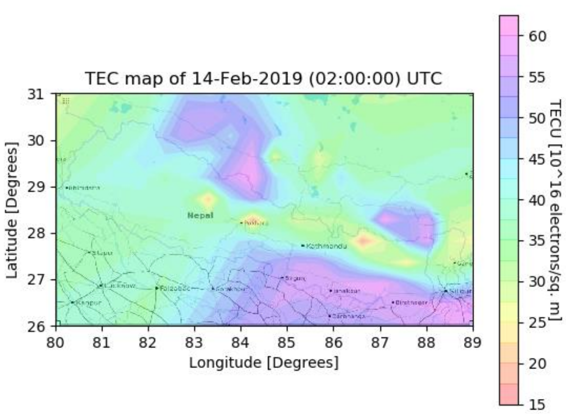



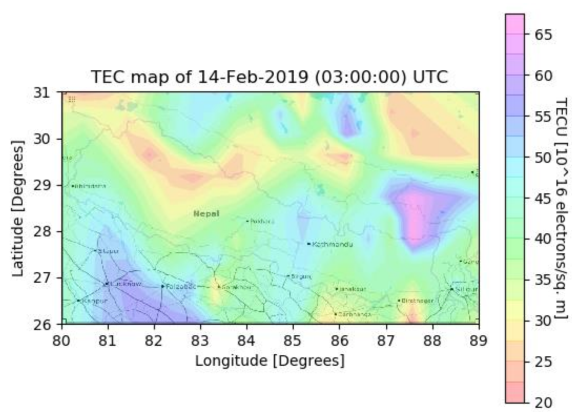

TEC map of 14-Feb-2019 (01:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

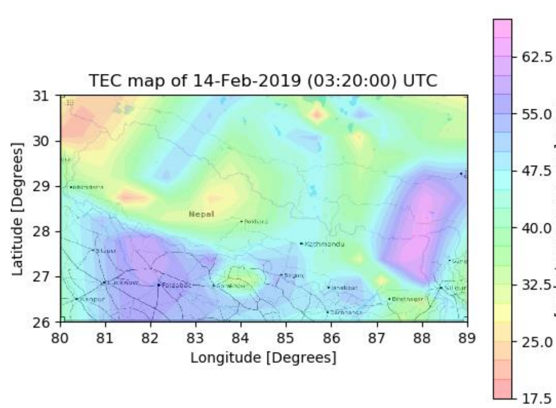

TEC map of 14-Feb-2019 (02:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

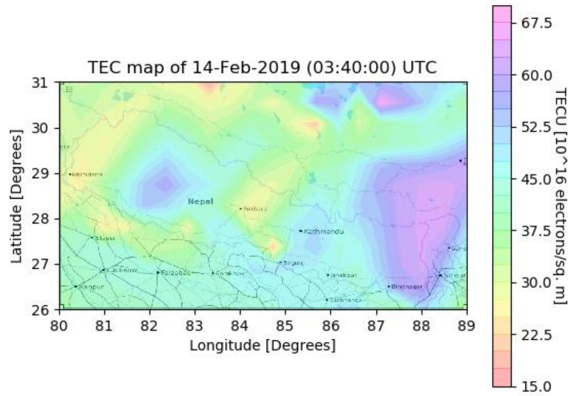

TEC map of 14-Feb-2019 (01:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

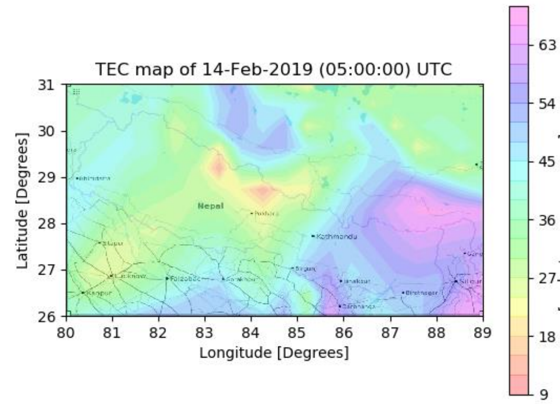

TEC map of 14-Feb-2019 (02:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

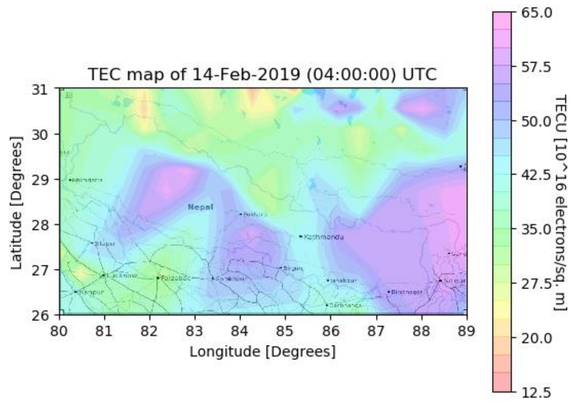

TEC map of 14-Feb-2019 (02:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

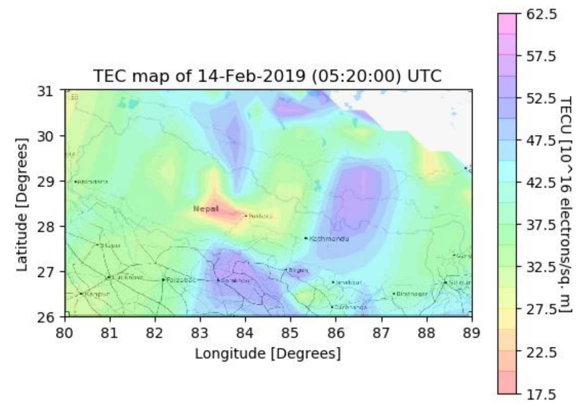

TEC map of 14-Feb-2019 (03:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

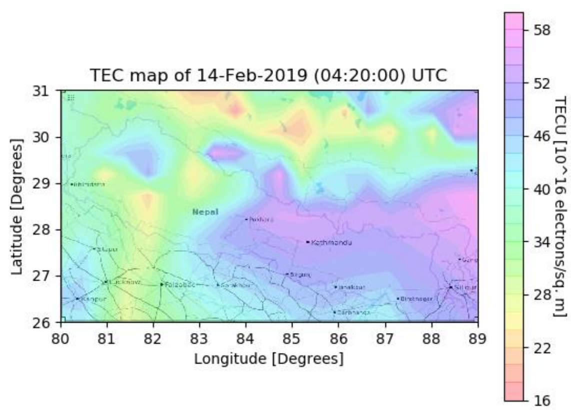

TEC map of 14-Feb-2019 (03:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

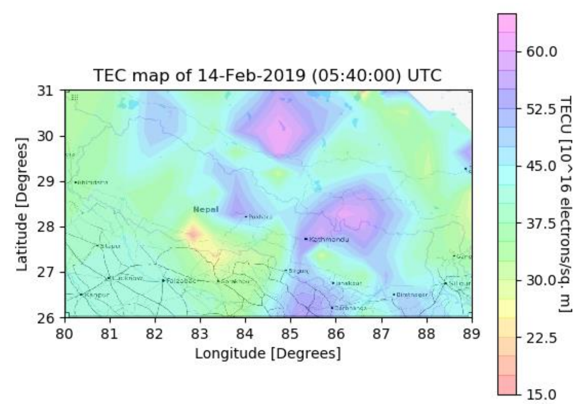

TEC map of 14-Feb-2019 (03:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

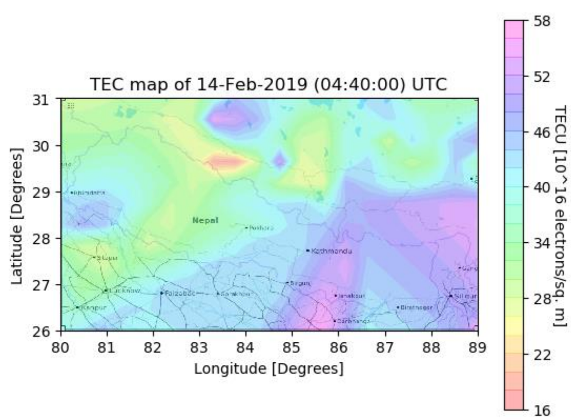

TEC map of 14-Feb-2019 (05:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

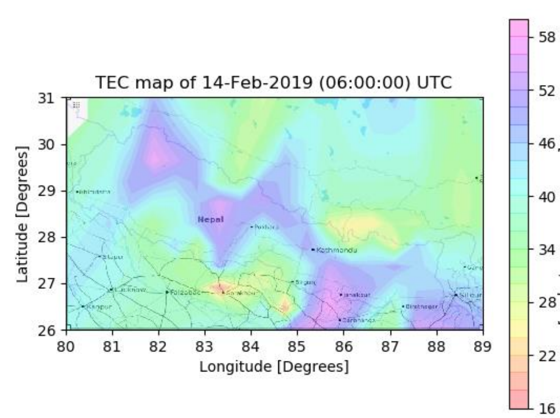

TEC map of 14-Feb-2019 (04:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

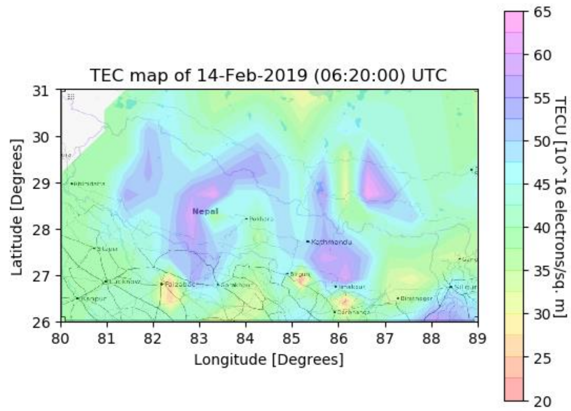

TEC map of 14-Feb-2019 (05:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]


TEC map of 14-Feb-2019 (04:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

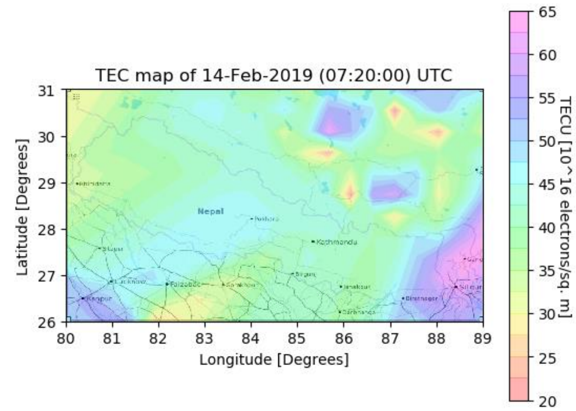

TEC map of 14-Feb-2019 (05:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]


TEC map of 14-Feb-2019 (04:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

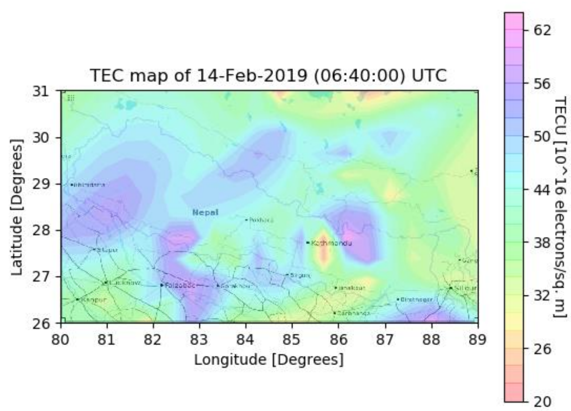

TEC map of 14-Feb-2019 (06:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]



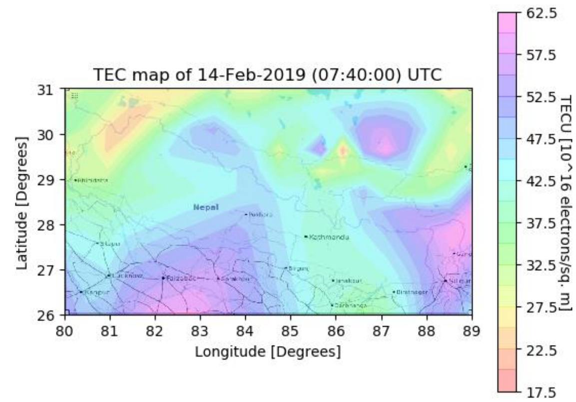
TEC map of 14-Feb-2019 (06:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

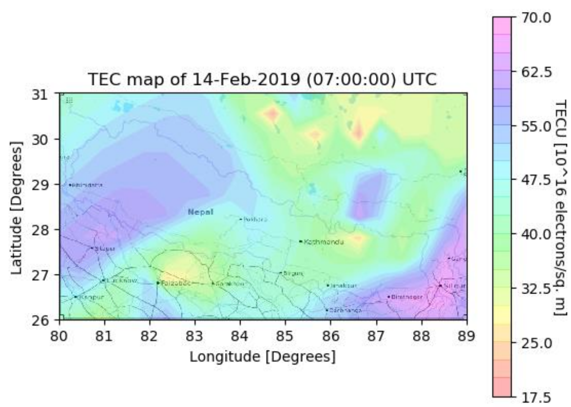
TEC map of 14-Feb-2019 (07:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

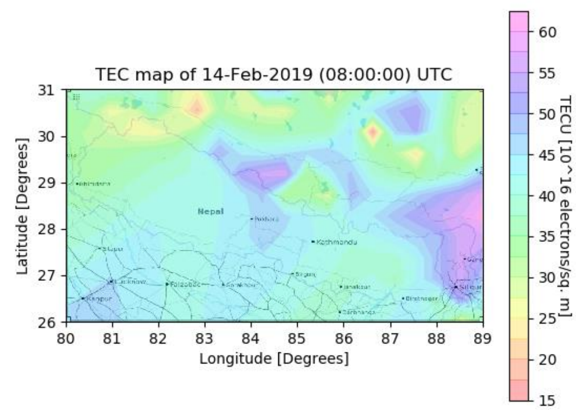
TEC map of 14-Feb-2019 (06:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

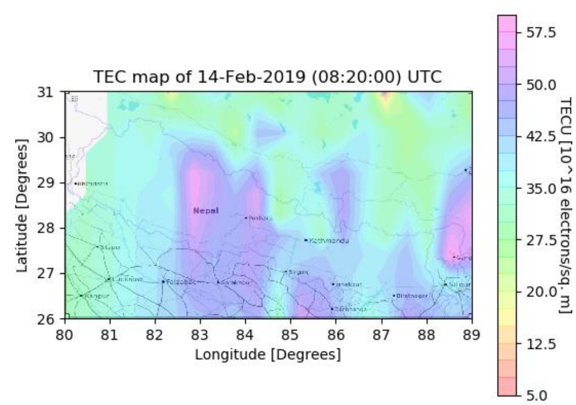
TEC map of 14-Feb-2019 (07:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

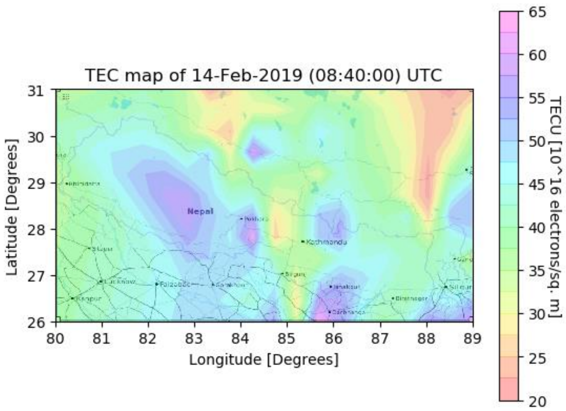
TEC map of 14-Feb-2019 (07:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

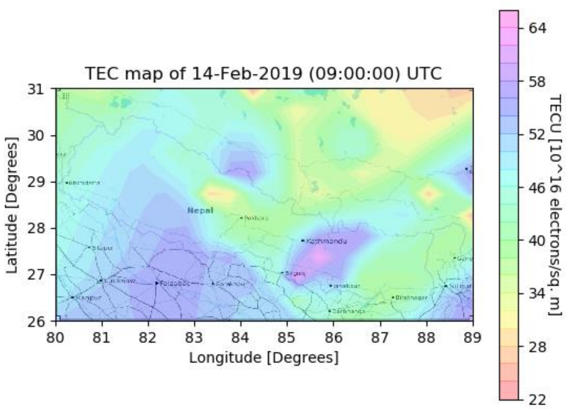



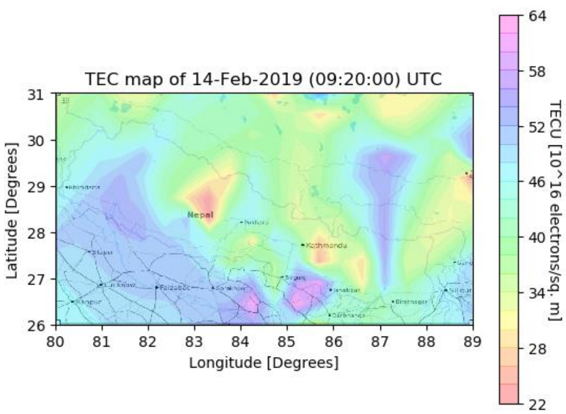
TEC map of 14-Feb-2019 (08:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

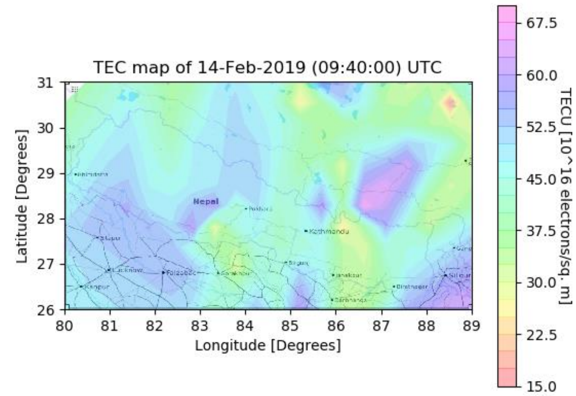



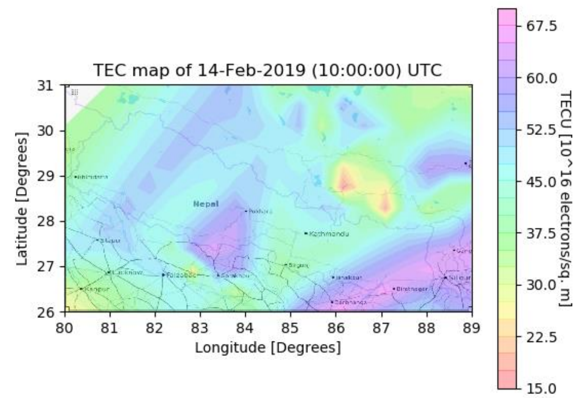
TEC map of 14-Feb-2019 (08:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

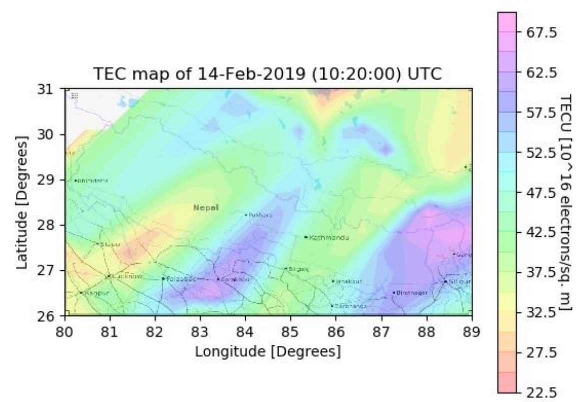

TEC map of 14-Feb-2019 (08:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

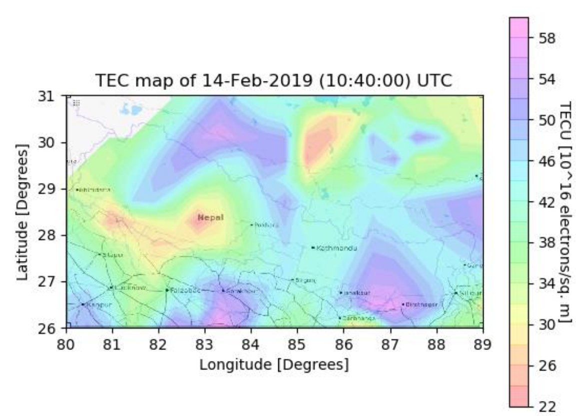

TEC map of 14-Feb-2019 (09:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

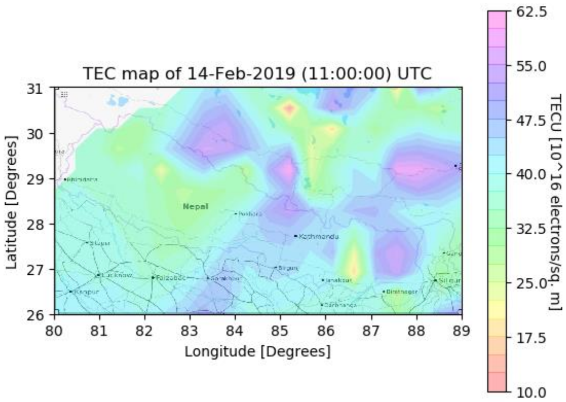

TEC map of 14-Feb-2019 (09:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

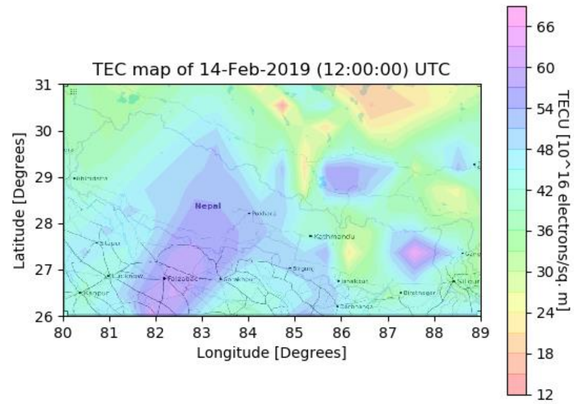

TEC map of 14-Feb-2019 (09:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

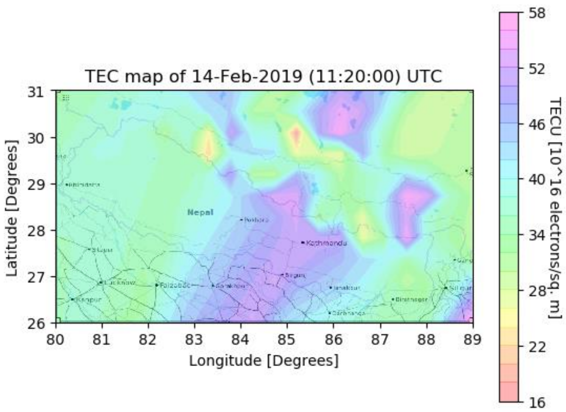

TEC map of 14-Feb-2019 (10:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

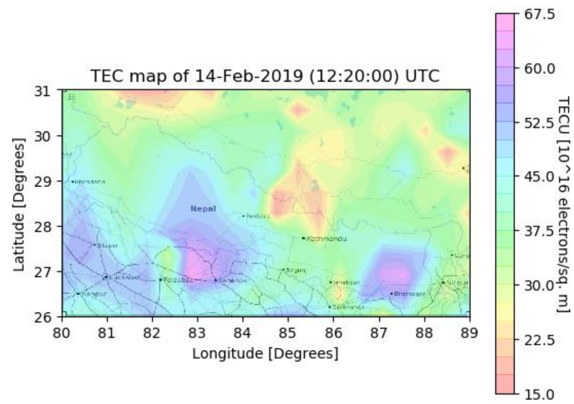

TEC map of 14-Feb-2019 (10:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

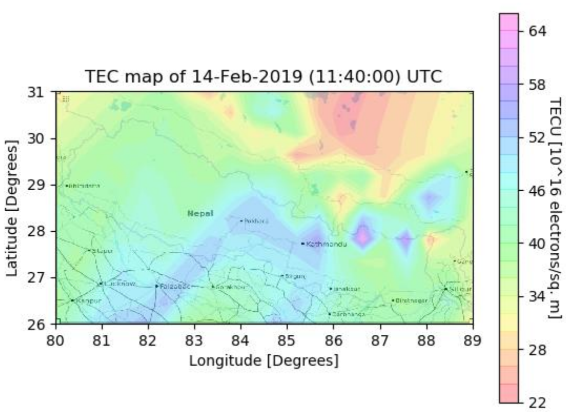

TEC map of 14-Feb-2019 (10:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

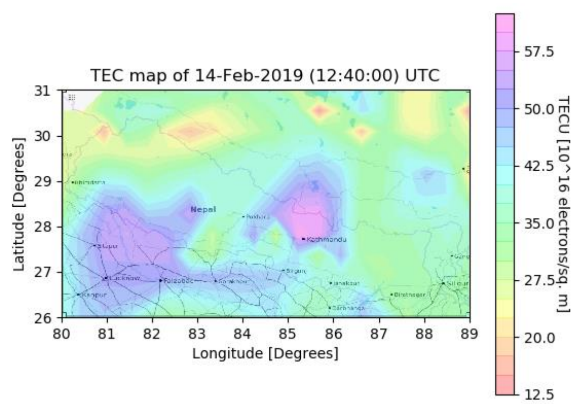

TEC map of 14-Feb-2019 (11:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal

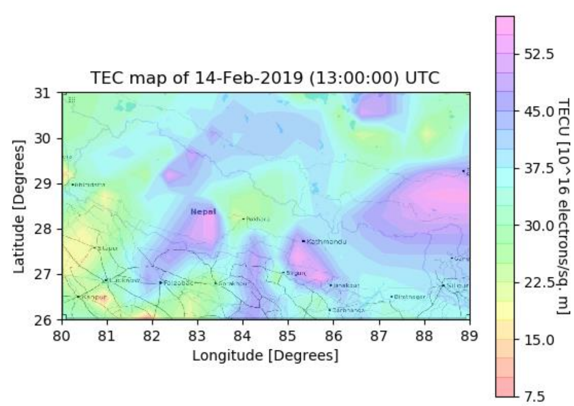

TEC map of 14-Feb-2019 (12:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal

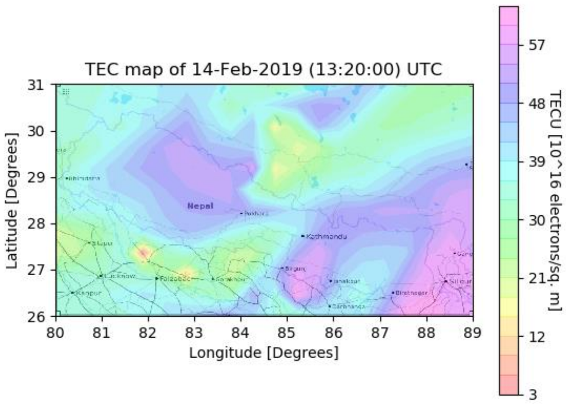

TEC map of 14-Feb-2019 (11:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal

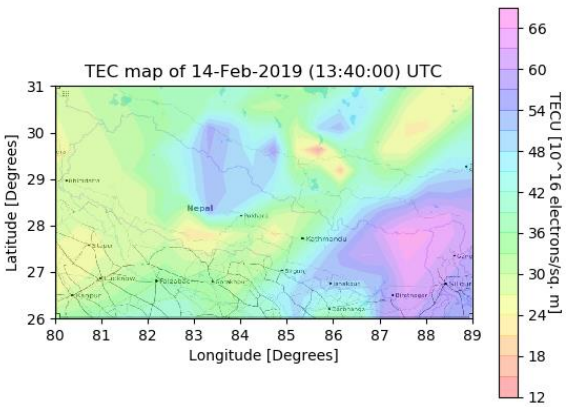

TEC map of 14-Feb-2019 (12:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal

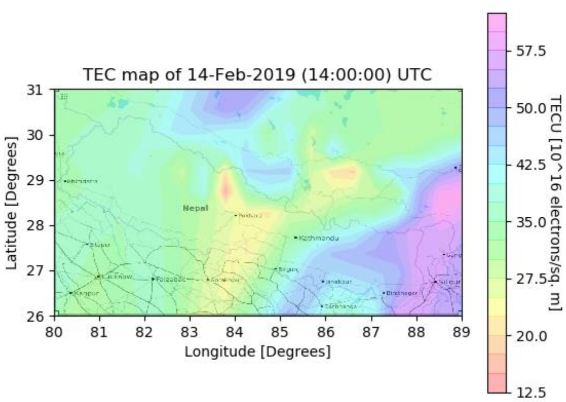

TEC map of 14-Feb-2019 (11:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal

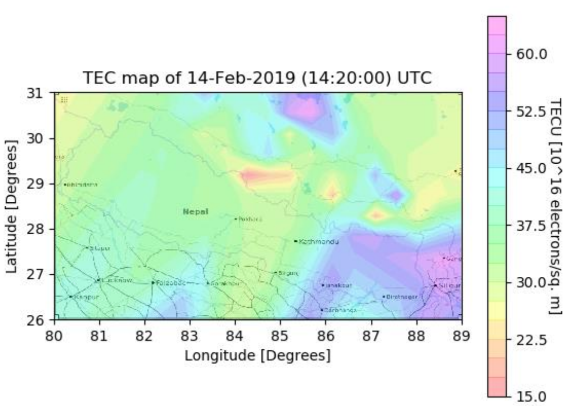

TEC map of 14-Feb-2019 (12:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal

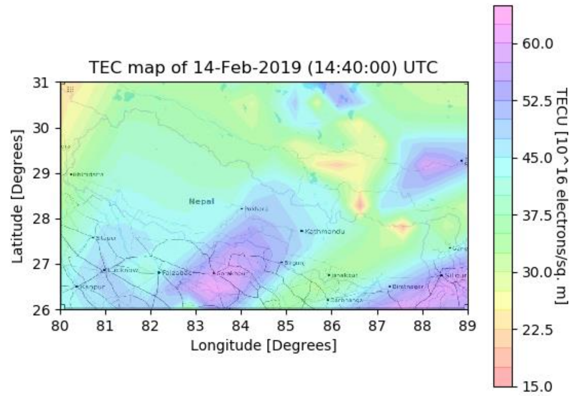

TEC map of 14-Feb-2019 (13:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal

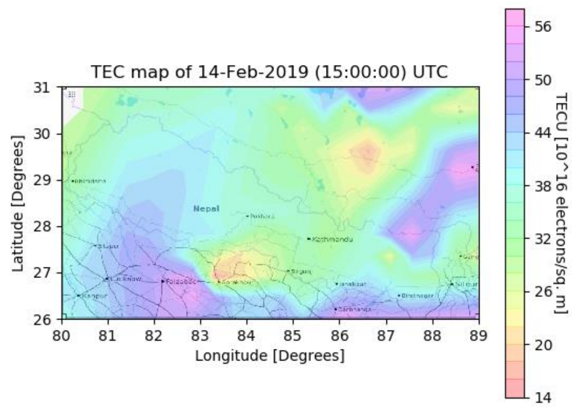

TEC map of 14-Feb-2019 (13:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

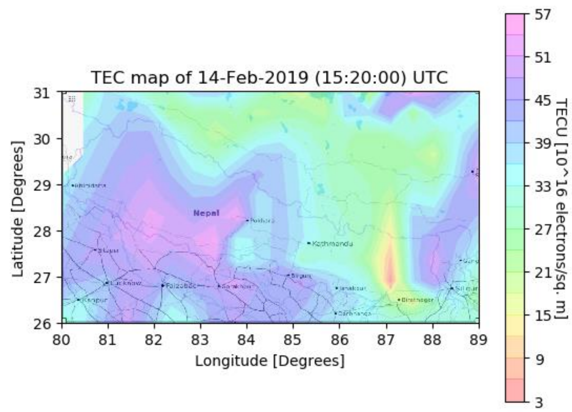

TEC map of 14-Feb-2019 (13:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

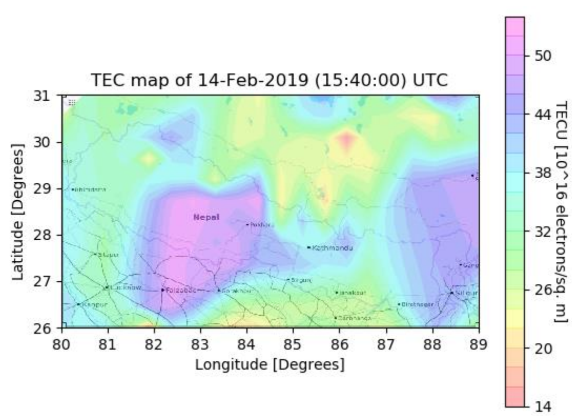

TEC map of 14-Feb-2019 (14:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

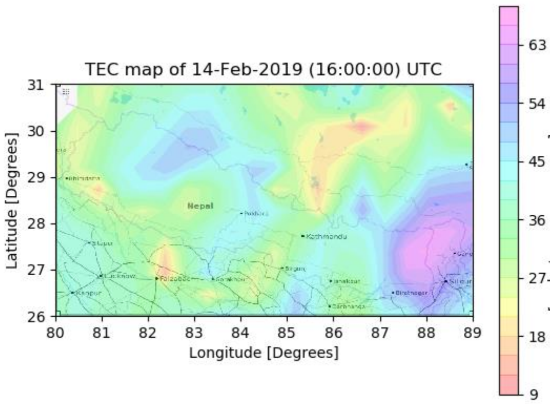

TEC map of 14-Feb-2019 (14:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

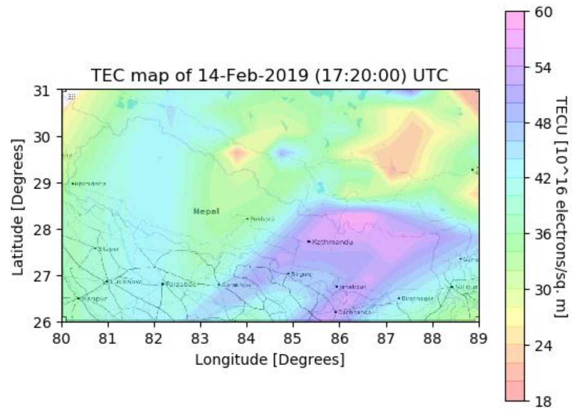

TEC map of 14-Feb-2019 (14:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

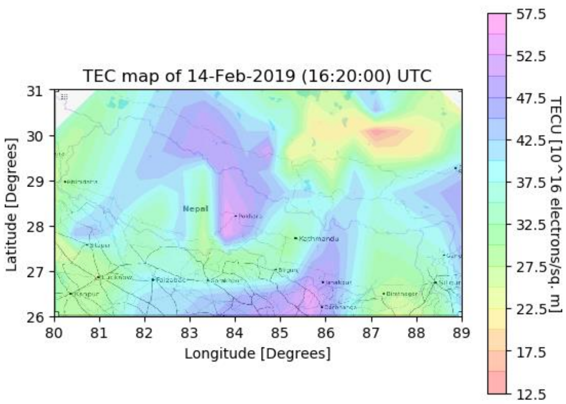

TEC map of 14-Feb-2019 (15:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

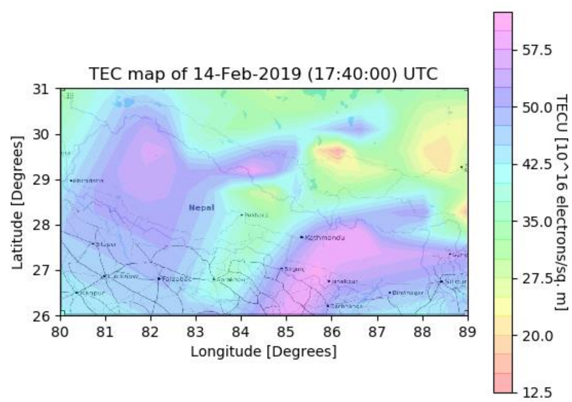

TEC map of 14-Feb-2019 (15:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

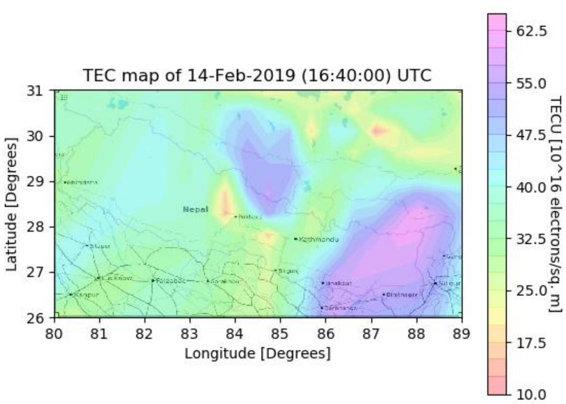

TEC map of 14-Feb-2019 (15:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
Nepal
TECU [10^16 electrons/sq. m]

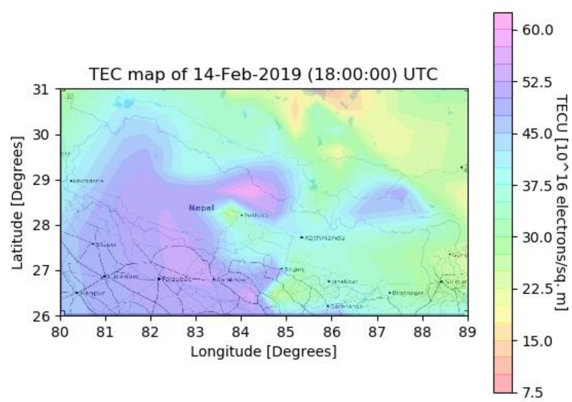

TEC map of 14-Feb-2019 (16:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

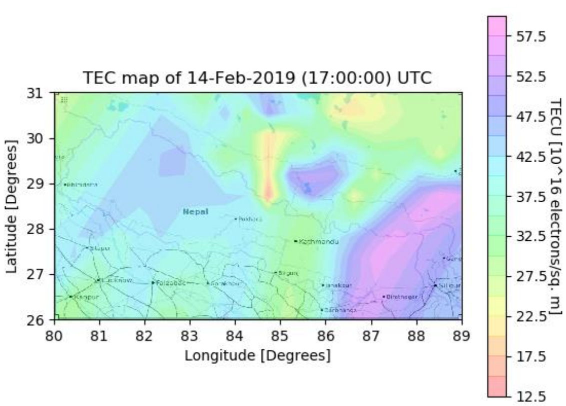

TEC map of 14-Feb-2019 (17:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

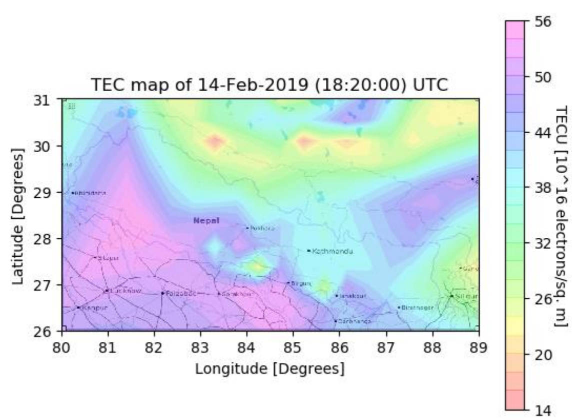

TEC map of 14-Feb-2019 (16:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

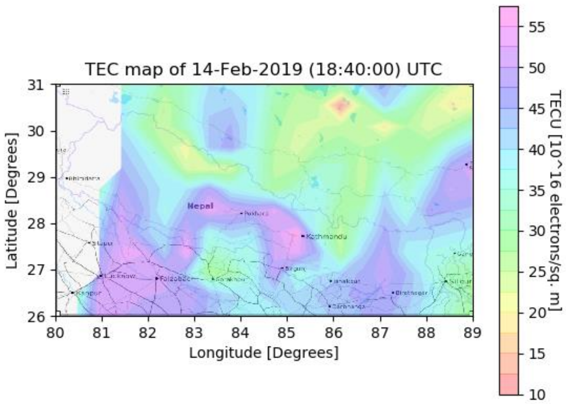

TEC map of 14-Feb-2019 (17:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

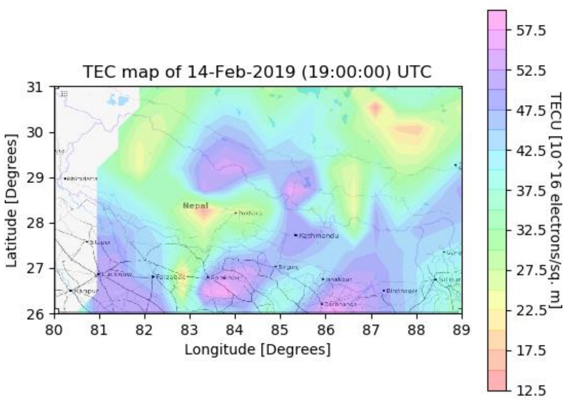

TEC map of 14-Feb-2019 (16:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

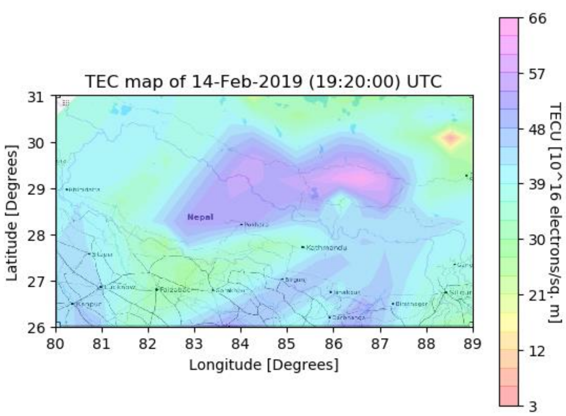

TEC map of 14-Feb-2019 (18:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

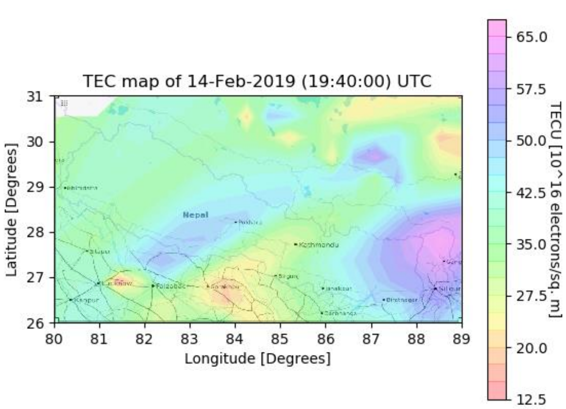

TEC map of 14-Feb-2019 (17:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

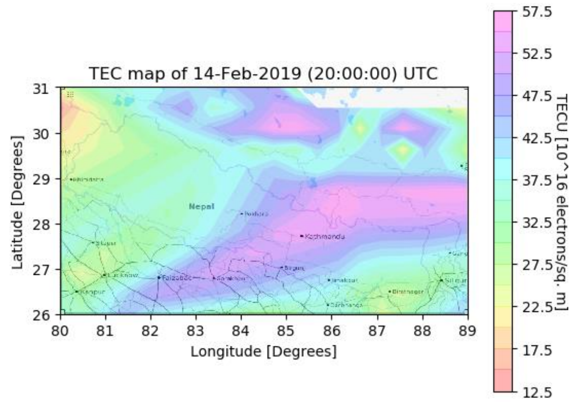

TEC map of 14-Feb-2019 (18:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

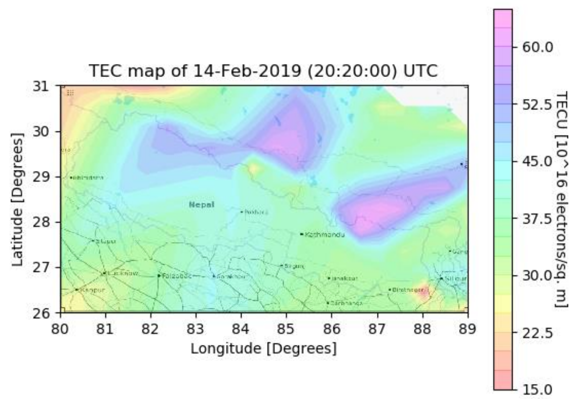

TEC map of 14-Feb-2019 (18:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

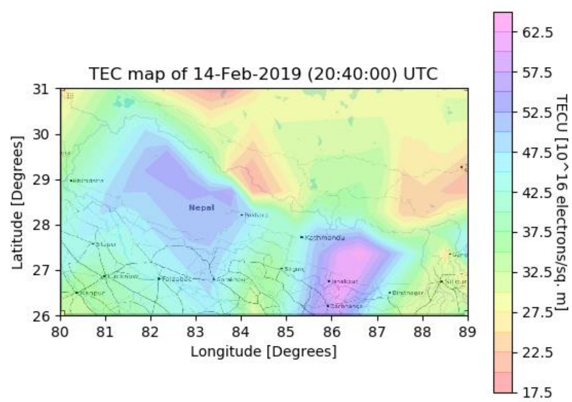

TEC map of 14-Feb-2019 (19:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

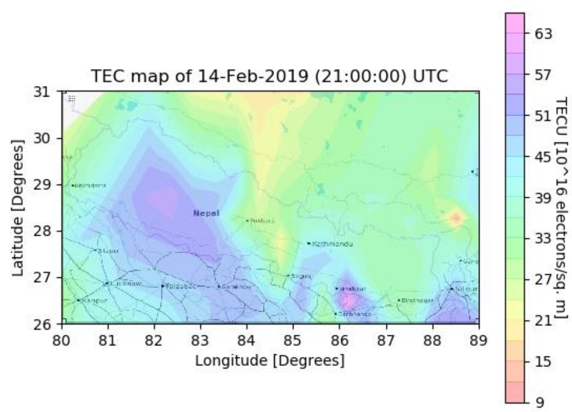

TEC map of 14-Feb-2019 (19:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

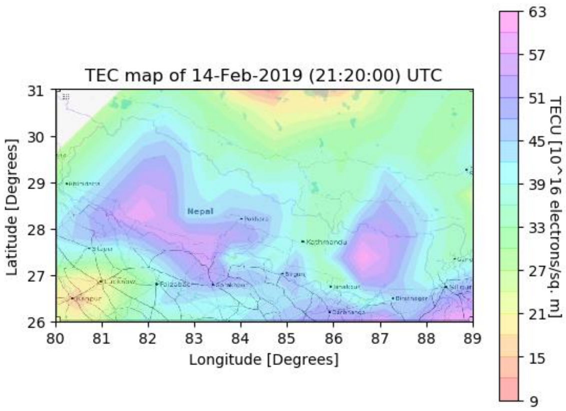

TEC map of 14-Feb-2019 (19:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

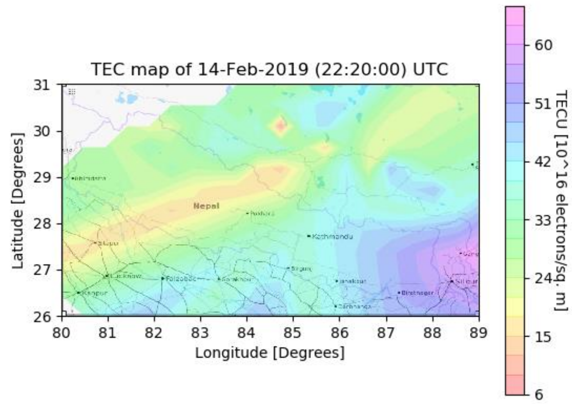

TEC map of 14-Feb-2019 (20:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

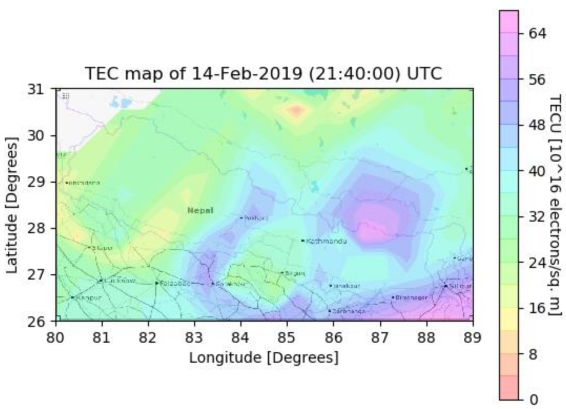

TEC map of 14-Feb-2019 (20:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

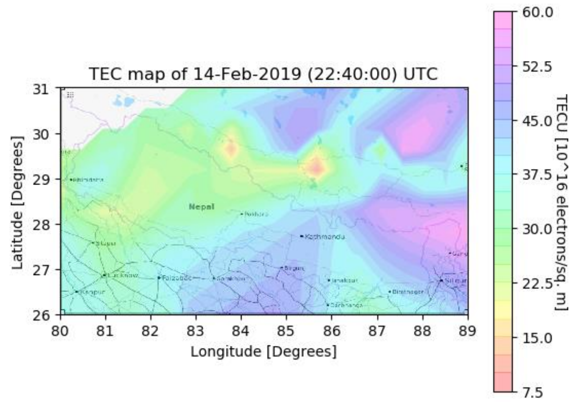

TEC map of 14-Feb-2019 (20:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

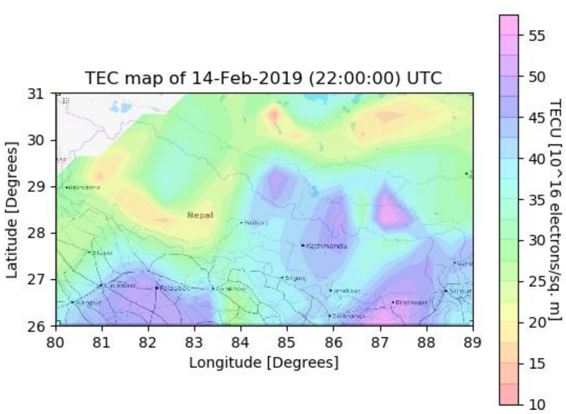

TEC map of 14-Feb-2019 (21:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

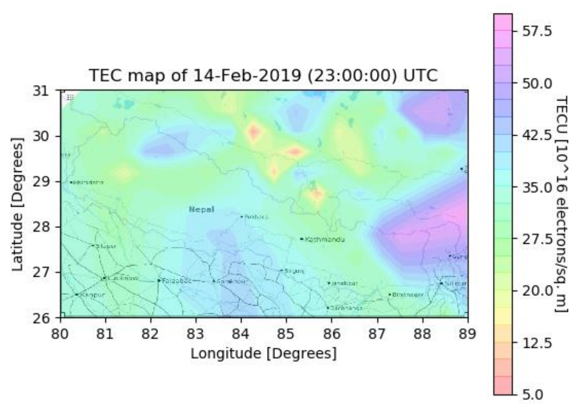

TEC map of 14-Feb-2019 (21:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu


TEC map of 14-Feb-2019 (22:20:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu


TEC map of 14-Feb-2019 (21:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu


TEC map of 14-Feb-2019 (22:40:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu


TEC map of 14-Feb-2019 (22:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu


TEC map of 14-Feb-2019 (23:00:00) UTC
Latitude [Degrees]
Longitude [Degrees]
TECU [10^16 electrons/sq. m]
Nepal
Kathmandu

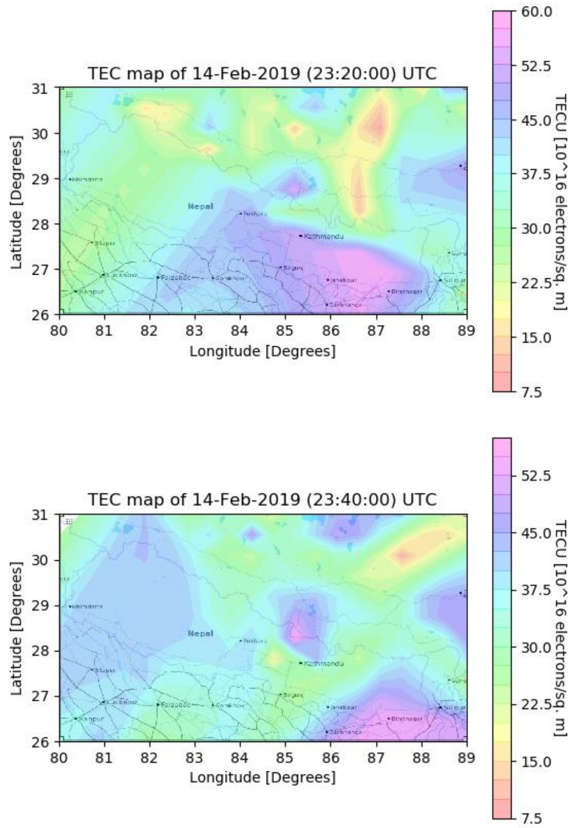



Figure 14: Generated TEC maps

The tool developed to generate these TEC maps is freely available for any type of use at https://github.com/Binabh/TEC-Maps-of-Nepal along with the usage instructions and description about the tool itself. Also the python code is attached in this document in ANNEX 3, 4 and 5 respectively.

Other output products are IONEX file and CSV file showing the values for TEC data both of which are included in ANNEX 1 and ANNEX 2 of the report. The CSV file consists of the VTEC values obtained from the stations at various times and the IONEX file consists of the interpolated values over a grid specified.

Also the screen captures showing the usage of the python tool developed is included in ANNEX 3 of this report.

# 5. CONCLUSION AND RECOMMENDATIONS

## 5.1. Conclusion

The objective of our project was met and TEC maps were generated using the python tool developed by us. Hence the regional TEC maps for desired time of day can now be obtained for Nepal using the freely available GNSS data from UNAVCO website. This method for TEC map production is economically efficient one however it is not most reliable method compared to the ones dedicated for TEC measurements. So, this project presents the method of GNSS as remote sensing of atmosphere. The project is one of its first type in case of Nepal and the results can be useful in various applications like space weather study, signal propagation path study etc.

Also, looking at the map we find that the ionospheric activity is varying with solar activity and changing constantly over a period of time. However, distinct pattern of the ionospheric activity was hard to estimate perhaps due to limitation in data points over the study area and very simplistic approach of TEC estimation.

## 5.2. Limitation of the project

Being constrained with Time and resources and our knowledge the project is not fully fledged with all the capabilities of Total Electron Content estimation. Some of its limitations are:

- The result obtained is limited within the coverage area of the receivers.
- Uneven distribution of the receivers results in some abnormal results after interpolation.
- Due to limited number of data points the interpolation at times may be rough while visualization however the numerical values are always reliable.
- Due to time constraints and inconvenience we were not able to compare our result with other models. Only visual inspection was done in order to ensure that the result is not highly abnormal.

## 5.3. Recommendation and way forward

Finally, with the experience gained during this project we would like to make following recommendations:

- The CORS distributed all over Nepal must be made functional and must come to regular operation which will serve as data bank for these types of research.

- More focus should be given towards the use of GNSS data in Nepal for study of these and other types of natural phenomena.
- The project is freely available for anyone to add features and use for their own use. So, we encourage the interested students/researchers to have a look at our work and add new features to the tool like model fitting with global models, correction formulas, inclusion of observables from other satellite systems etc.

# 6. REFERENCES


Alcay, S. and Yigit, C.O. and Seemala, G. and Ceylan, A. (2014). GPS-based ionosphere modeling: A brief review. Fresenius Environmental Bulletin. 23 (3 A): pp. 815-824.

Ansari, Kutubuddin, and Kwan-Dong Park. (2018). "Multi Constellation GNSS Precise Point Positioning and Prediction of Propagation Errors Using Singular Spectrum Analysis." *Astrophysics and Space Science* 363(12). http://link.springer.com/10.1007/s10509-018-3479-7 (January 22, 2019)

BaşçiFtçi, F., İNal, C., Yildirim, Ö., & Bülbül, S. (2017). *Determination of Regional TEC Values by GNSS Measurements, A Case Study: Central Anatolia Sample, Turkey*. 17.

Br, D. (2013). *RINEX-based GNSS positioning performance data analysis using the open source tool*. 14.

Dach, R., Lutz, S., Walser, P. ve Fridez, P. (2015). Bernese GNSS Software Version 5.2, Switzerland, Astronomical Institute, University of Bern.

ESA. (2011). GPS Galileo Coord Comp Table. Retrieved August 23, 2019, from ResearchGate website: https://www.researchgate.net/publication/279396991_CODE_five_system_solution_for_IGS_MGEX

Fedrizzi, M., Paula, E., Santos, M., Komjathy, A., (2001). "Mapping the Low-Latitude Ionosphere with GPS.": 7.

Gizawy, M. L. (2003) "Development of an ionosphere monitoring technique using GPS measurements for high latitude GPS users", Ph.D. Thesis, University of Calgary, Italy.

Gurtner, W. (1998, January 13). [IGSMAIL-1785] Introduction of the Hatanaka Compression. Retrieved August 25, 2019, from https://lists.igs.org/pipermail/igsmail/1998/003157.html

Memarzadeh, Yahya. (2009). "Ionospheric Modeling for Precise GNSS Applications." K. N. Toosi University of Technology.

Mylnikova, A. A., Yasyukevich, Yu. V., Kunitsyn, V. E., & Padokhin, A. M. (2015). Variability of GPS/GLONASS differential code biases. *Results in Physics*, *5*, 9–10. https://doi.org/10.1016/j.rinp.2014.11.002

Panda, S. K., Gedam, S. S., & Rajaram, G. (2015). Study of Ionospheric TEC from GPS observations and comparisons with IRI and SPIM model predictions in the low latitude anomaly Indian subcontinental region. *Advances in Space Research*, *55*(8), 1948–1964. https://doi.org/10.1016/j.asr.2014.09.004

Psiaki, Mark L, Gray S Bust, and Cathryn N Mitchell. (2015). "Nonlinear Estimation to Assimilate GPS TEC Data into a Regional Ionosphere Model.": 14.

Schaer, S., (1999). "Mapping and Predicting the Earth's Ionosphere Using the Global Positioning System", Ph.D Thesis, Universitat Bern, Switzerland, 228p.

Sharma, Sunil Kumar, Kutubuddin Ansari, and Sampad Kumar Panda. (2018). "Analysis of Ionospheric TEC Variation over Manama, Bahrain, and Comparison with IRI-2012 and IRI-2016 Models." *Arabian Journal for Science and Engineering* 43(7): 3823–30.

Takahashi, H. et al. (2016). "Ionospheric TEC Weather Map Over South America: IONOSPHERIC WEATHER OVER SOUTH AMERICA." *Space Weather* 14(11): 937–49.

Volker SCHWIEGER, M. L. (2009, December). *GNSS CORS - Reference Frames and Services.* Retrieved from FIG - International Federation of Surveyors: https://www.fig.net/resources/monthly_articles/2009/december_2009/december_2009_schwieger_etal.pdf

Wielgosz, P., Grejner-Brzezinska, D., & Kashani, I. (2003). Regional Ionosphere Mapping with Kriging and Multiquadric Methods. *Journal of Global Positioning Systems*, *2*(1), 48–55. https://doi.org/10.5081/jgps.2.1.48

# 7. ANNEXES

## 7.1. ANNEX 1

Sample of CSV data obtained as output. Only some portion of file is shown below:

```
Datetime,Satnum,slanttec,elevationangle,verticaltec,lat,lon
14-Feb-2019
(00:00:00),G14,61.397758687628524,15.713442973206973,26.019750560815503,21
.450319248473292,75.31621373244205
14-Feb-2019
(00:00:00),G10,60.61582027093545,41.40140151317338,42.94196869368574,30.65
250086893783,83.9728254877171
14-Feb-2019
(00:00:00),G20,48.30975732975586,56.844224462675164,41.422026854200176,30.
362270190352337,85.62247117193012
14-Feb-2019
(00:00:00),G32,64.70938757190271,34.467252449725834,40.834630875998776,25.
067898997759656,79.80573434383366
14-Feb-2019
(00:00:00),G21,46.647632629972826,80.70662812534455,46.10597179735898,27.6
9142285062713,84.29830104713254
14-Feb-2019
(00:00:00),G27,60.027961872984434,21.750160395207036,29.175758241788326,32
.03488245726581,82.35938554533949
14-Feb-2019
(00:00:00),G15,67.01384517696036,24.386709892325584,34.53614955010071,32.6
97585617308015,89.6392636129049
14-Feb-2019
(00:00:00),G24,54.14386283666504,34.59346614062457,34.24534561509193,28.39
3796863679725,88.97004256726608
14-Feb-2019
(00:20:00),G14,59.170714799480905,23.509557613524976,29.911289490312758,23
.644388166751657,77.62304611228645
14-Feb-2019
(00:20:00),G10,65.64094500617313,45.5566010507804,49.38060648128327,30.757
158752989994,84.68960090373388
14-Feb-2019
(00:20:00),G20,39.38254713977623,59.43492088701109,34.581842715110206,30.2
0676642175474,84.4509757823725
14-Feb-2019
(00:20:00),G32,70.83851698570156,42.581496660289524,51.08257658385959,26.2
21343702168237,80.8160609104582
14-Feb-2019
(00:20:00),G21,41.62250789473515,69.63532463301127,39.32799596232319,27.05
0682490022062,84.34100377883681
14-Feb-2019
(00:20:00),G27,59.47595951597653,23.921059220676305,30.33982022479177,30.7
84860762518907,81.39121299235941
14-Feb-2019
(00:20:00),G24,57.493946005308366,33.985644899527,35.96465590209793,29.240
465279639285,89.107935519036
```

14-Feb-2019
(00:40:00),G14,68.08840772394791,31.535316862892437,40.67387220847507,25.208851380600144,79.2409308701604
14-Feb-2019
(00:40:00),G10,63.40438381705824,49.841174350927375,50.39676187457052,30.724014634162458,85.26297730469004
14-Feb-2019
(00:40:00),G20,45.51167651812044,61.70584510202173,40.734525247145385,29.949283789955693,84.99801218082006
14-Feb-2019
(00:40:00),G32,68.04995343957877,49.95581129636556,54.163947298570356,27.136424395590357,81.64269768860437
14-Feb-2019
(00:40:00),G21,52.7862793092843,58.52955153051766,45.97878889035999,26.314815340618583,84.3704342441662
14-Feb-2019
(00:40:00),G27,53.33731283666505,24.304505305130924,27.43841380229154,29.631618692444583,80.31396866686535
14-Feb-2019
(00:40:00),G24,59.72098992891062,31.71368823167261,35.79825657484131,30.128414264418286,89.34061436321795
14-Feb-2019
(01:00:00),G14,45.22789713977623,39.504842876299534,31.10346695696311,26.390925840621072,80.45714815528011
14-Feb-2019
(01:00:00),G10,58.95029600530837,54.14302629636073,49.18811710129826,30.56880809975675,85.69383166639672
14-Feb-2019
(01:00:00),G20,52.2118428554071,63.21523514959939,47.2858420262940 9,29.59916856252603,85.48479961637501
14-Feb-2019
(01:00:00),G32,57.457218913062796,56.01233524334246,48.866820255610456,27.896327284189294,82.381500302489
14-Feb-2019
(01:00:00),G21,48.322674232021804,47.705410637755506,37.40164453071764,25.428843508307715,84.40127451748572
14-Feb-2019
(01:00:00),G27,56.12587638278784,22.880699311624426,27.97879480106054,28.4422721847105,79.0529571278935
14-Feb-2019
(01:00:00),G24,60.28250951597652,27.956615683421546,33.52257147081278,31.135114981126844,89.72821122380454
14-Feb-2019
(01:20:00),G14,53.59358770723533,46.91422912437892,41.0572821920308,27.33164681101516,81.44459298757822
14-Feb-2019
(01:20:00),G10,59.5118156278289,58.147392238505724,51.65640971263529,30.307780837154866,84.4438932671006
14-Feb-2019
(01:20:00),G20,47.74823774268996,63.24975097471031,43.25466757889746,29.163254053655102,85.89801019259167
14-Feb-2019
(01:20:00),G32,66.93643149550496,60.173169157010165,59.153725686997795,28.545695731996357,83.08849442481957

……………………………………………………..
14-Feb-2019
(23:40:00),G27,37.61476347699923,16.908527067994306,16.376569495234197,33.
259305090768116,85.19920314200441
14-Feb-2019
(23:40:00),G24,31.60693981174243,36.29307050125498,20.601618376962918,27.2
72641786134038,90.87061695193165
14-Feb-2019
(23:40:00),G15,37.719652186220664,31.22495547043871,22.39751229941493,31.3
88368719099265,90.78091465328752

## 7.2. ANNEX 2

Sample of IONEX file obtained as output. Only some portion of file is shown below:

```
     1.0            IONOSPHERE MAPS     GNSS                IONEX VERSION
/ TYPE
FinalProjV1         KU                  12-Sep-19 22:36     PGM / RUN BY /
DATE
Ionospheric Model generated using CORS data of Nepal        COMMENT
Regional Ionospheric Model of Nepal. This is the product    DESCRIPTION
of final year project of Undergraduate at Kathmandu         DESCRIPTION
University                                                  DESCRIPTION
Contact address: binabhdevkota@gmail.com                    DESCRIPTION
  2019     02     14     00     00     00                   EPOCH OF FIRST
MAP
  2019     02     14     23     40     00                    EPOCH OF LAST
MAP
  1200                                                      INTERVAL
    72                                                        # OF MAPS IN
FILE
    COSZ                                                   MAPPING FUNCTION
     0.0                                                   ELEVATION CUTOFF
Pseudorange values with DCB correction                         OBSERVABLES
USED
  6371.0                                                    BASE RADIUS
     2                                                      MAP DIMENSION
   400.0 400.0   0.0                                         HGT1 / HGT2 /
DHGT
    80.0 89.0  0.5                                           LAT1 / LAT2 /
DLAT
    26.0 31.0  0.5                                           LON1 / LON2 /
DLON
    -1                                                      EXPONENT
TEC/RMS values in 0.1 TECU; 9999, if no value available     COMMENT
                                                            END OF HEADER
     1                                                        START OF TEC
MAP
  2019     02     14     00     00     00                  EPOCH OF CURRENT
MAP
    26.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    40    43    46    45    35    29    29    32    39    37    34    38
44    49    55    57
    56    55    54    53
    26.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    38    40    43    46    40    31    32    32    35    37    34    31
37    42    43    50
    52    50    51    52
    27.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    37    39    40    43    45    35    34    34    34    37    34    31
29    29    33    39
    44    47    41    39
    27.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    36    38    39    40    43    41    37    37    37    37    21    38
48    32    23    28
```

```
    33    38    41    36
    28.0 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    35    37    38    40    41    43    39    40    35    43    23    58
49    37    28    29
    31    33    34    36
    28.5 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    35    36    37    39    40    42    42    36    31    38    17    52
56    44    35    30
    35    37    39    30
    29.0 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    34    35    37    38    39    41    42    40    33    32    33    46
51    51    42    36
    35    41    38    36
    29.5 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    34    34    36    37    38    40    42    44    38    27    33    40
44    50    49    43
    35    33    26    33
    30.0 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    34    34    35    36    38    37    41    44    42    30    27    34
38    40    26    21
    27    20    25    31
    30.5 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    32    33    34    35    37    31    41    43    46    35    23    26
34    22    36    34
    23    19    24    29
    31.0 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    29    32    33    35    36    25    40    43    41    17    38    25
24    20    47    37
    26    20    23    27
    31.5 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    9999    32    32    34    31    26    40    34    33    31    25    20
21    21    27    40
    30    22    23    26
     1                                                          END OF TEC MAP
     2                                                            START OF TEC
MAP
  2019     02     14     00     20     00                     EPOCH OF CURRENT
MAP
    26.0 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    32    38    26    33    49    43    33    34    39    40    28    29
31    33    33    43
    45    48    49    48
    26.5 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    36    42    29    45    57    44    34    31    35    40    31    19
17    34    29    38
    49    42    43    47
    27.0 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    39    40    34    44    53    46    34    30    33    38    11    52
45    29    24    32
    43    40    38    39
    27.5 80.0 89.0   0.5 400.0                              LAT/LON1/LON2/DLON/H
    38    36    40    43    48    44    35    31    30    36    25    56
38    23    28    27
    37    47    36    34
```

```
    28.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    37    33    39    42    43    39    35    32    29    34    31    47
30    18    29    23
    31    43    36    32
    28.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    36    33    35    41    38    34    33    32    29    32    36    39
23    23    29    32
    25    37    45    31
    29.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    35    32    32    37    34    29    30    32    29    30    40    30
15    29    30    42
    22    31    44    34
    29.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    35    31    28    33    29    24    27    32    30    28    32    21
8    31    30    44
    32    24    36    37
    30.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    34    30    26    28    23    21    24    28    30    28    23    13
13    32    31    25
    32    17    28    32
    30.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    33    29    25    23    18    22    21    25    27    27    3    33
11    34    34    25
    34    34    27    29
    31.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    32    29    25    15    12    31    28    24    23    31    29    34
37    44    31    26
    30    34    30    26
    31.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    30    28    20    18    31    37    36    33    29    35    36    48
51    40    28    24
    26    30    32    26
     2                                                       END OF TEC MAP

…………………………..
     72                                                        START OF TEC
MAP
  2019     02     14     23     40     00                EPOCH OF CURRENT
MAP
    26.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    34    38    41    33    28    29    29    37    35    30    31    38
44    51    53    52
    52    52    51    51
    26.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    36    35    39    38    30    31    31    32    37    32    28    31
38    44    51    54
    54    53    51    48
    27.0 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    40    38    37    41    35    33    33    34    36    34    30    25
31    37    43    50
    47    43    39    34
    27.5 80.0 89.0   0.5 400.0                          LAT/LON1/LON2/DLON/H
    40    40    40    38    40    35    35    36    36    37    32    27
24    29    32    41
```

```
    42    38    34    24
    28.0 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    40    41    41    41    40    37    37    38    38    38    17    29
44    31    23    33
    38    34    30    38
    28.5 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    41    41    41    41    41    42    39    40    35    39    32    56
44    31    29    29
    33    29    35    47
    29.0 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    38    41    41    41    41    42    41    35    32    36    35    48
45    37    37    32
    31    41    47    46
    29.5 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    35    40    41    42    42    42    40    35    30    32    39    41
42    37    30    25
    31    41    45    46
    30.0 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    32    37    42    42    42    42    39    34    31    29    32    33
31    27    22    21
    22    29    37    40
    30.5 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    29    34    39    42    42    42    38    29    27    23    32    29
24    31    30    21
     9    17    24    32
    31.0 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    25    31    36    41    42    41    33    38    29    48    30    33
40    48    45    35
    24    16    14    19
    31.5 80.0 89.0  0.5 400.0                         LAT/LON1/LON2/DLON/H
    9999    28    33    38    43    35    27    29    30    26    21    30
39    44    44    42
    40    30    24    22
     72                                                  END OF TEC MAP
                                                         END OF FILE
```

## 7.3. ANNEX 3

Python code of the file main.py. It is the file that handles GUI part of tool:

```
import tkinter
from tkcalendar import DateEntry
import datetime
import os
import wget
import subprocess
import glob
from unlzw import unlzw
from tecvalues import driver
from IonexWriter import writeionex
import threading
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from scipy.interpolate import griddata
import csv
import time
import matplotlib.animation as animation
from ftplib import FTP

selected = []
dayyofyear = ''
stringdate = ''

#Providing interface to select the GPS stations
def selectstns():
    def run():
        def on_click():
            for station, intvar in zip(stations, intvars):
                if intvar.get() == 1 and not station in selected:
                    selected.append(station)
                elif intvar.get() == 0 and station in selected:
                    selected.remove(station)
            master.destroy()

        master = tkinter.Tk()
        master.title("Select Stations")
        mydict = {}
        reader = csv.reader(open('stations.csv','r'))
        header = next(reader)
        for row in reader:
            mydict[row[0].lower()]                                          =
{header[1]:row[1],header[2]:row[2],header[3]:row[3],header[4]:row[4],heade
r[5]:row[5]}
        stations = mydict.keys()
        intvars = []
        checkbuttons = []
        for station in stations:
            intvar = tkinter.IntVar(master)
            if station in selected:
```

```
                intvar.set(1)
            checkbutton            =            tkinter.Checkbutton(master,
text=mydict[station][header[1]], variable=intvar)
            checkbutton.pack(anchor='w')
            intvars.append(intvar)
            checkbuttons.append(checkbutton)

        button = tkinter.Button(master, text="Done", command=on_click)
        button.pack()

        master.mainloop()
    thread1 = threading.Thread(target=run)
    thread1.start()

#For plotting and visualizing the data
def plotter():
    dirpath='data\\'+stringdate[0:4]+'\\'+dayyofyear
    filepath = dirpath+'\\TECValues.csv'
    mydateparser = lambda x: pd.datetime.strptime(x, "%d-%b-%Y (%H:%M:%S)")
    dataframe                  =                    pd.read_csv(filepath,
parse_dates=['Datetime'],date_parser=mydateparser)
    writeionex(dataframe,dirpath)
    timelist = dataframe['Datetime'].tolist()
    timelist = set(timelist)
    timelist = sorted(timelist)
    f = plt.figure()
    def animate(i):
        f.clear()
        ax = plt.subplot(1,1,1)
        subdataframe = dataframe.loc[dataframe['Datetime'] == timelist[i]]
        y = subdataframe['lat'].to_numpy()
        x = subdataframe['lon'].to_numpy()
        z = subdataframe['verticaltec'].to_numpy()
        xi = np.linspace(80,89,20)
        yi = np.linspace(26,31,12)
        zi    =    griddata((x,    y),    z,    (xi[None,:],    yi[:,None]),
method='linear',rescale=True)
        cf                                                                  =
ax.contourf(xi,yi,zi,levels=20,cmap='gist_rainbow',alpha=0.3,antialiased=T
rue)
        ax.set(xlim=(80, 89), ylim=(26, 31))
        ax.set_title('TEC    map    of    '+timelist[i].strftime("%d-%b-%Y
(%H:%M:%S)")+" UTC")
        ax.set_xlabel('Longitude [Degrees]')
        ax.set_ylabel('Latitude [Degrees]')
        plt.imshow(plt.imread(r'map.JPG'), alpha=1, extent=[80,89,26,31])
        cbar = plt.colorbar(cf)
        cbar.ax.set_ylabel('TECU      [10^16      electrons/sq.      m]',
rotation=270,labelpad=10)
        #plt.savefig(timelist[i].strftime("%d-%b-%Y                  (%H-%M-
%S)")+'.png',bbox_inches='tight')
        return ax
```

```
    ani                                                                    =
animation.FuncAnimation(f,animate,len(timelist),interval=1*1e+3,blit=False
)
    #f.colorbar(cf, ax=ax)
    plt.show()

#Download DCB files from CODE
def getdcbfiles(stringdate,dirpath):
    ftp = FTP('ftp.aiub.unibe.ch')
    ftp.login('anonymous','anonymous')
    ftp.cwd('/CODE/'+stringdate[0:4])
    downfilestr1='RETR P1C1'+stringdate[2:4]+stringdate[5:7]+'.DCB.Z'
    downfilestr2='RETR P1P2'+stringdate[2:4]+stringdate[5:7]+'.DCB.Z'
    filestr1 = dirpath+'\\P1C1'+stringdate[2:4]+stringdate[5:7]+'.DCB.Z'
    filestr2 = dirpath+'\\P1P2'+stringdate[2:4]+stringdate[5:7]+'.DCB.Z'
    if   datetime.datetime.now().strftime('%m')   ==   stringdate[5:7]   and
datetime.datetime.now().strftime('%Y') == stringdate[0:4]:
        ftp.cwd('..')
        downfilestr1='RETR P1C1.DCB'
        downfilestr2='RETR P1P2.DCB'
        filestr1 = dirpath+'\\P1C1'+stringdate[2:4]+stringdate[5:7]+'.DCB'
        filestr2 = dirpath+'\\P1P2'+stringdate[2:4]+stringdate[5:7]+'.DCB'
    if not os.path.exists(filestr1):
        p1c1File = open(filestr1, 'wb')
        ftp.retrbinary(downfilestr1, p1c1File.write)
        p1c1File.close()
    if not os.path.exists(filestr2):
        p1p2File = open(filestr2, 'wb')
        ftp.retrbinary(downfilestr2, p1p2File.write)
        p1p2File.close()
    ftp.close()

#Download the GNSS Data from UNAVCO site and DCB from CODE
def downloaddata():
    global dayyofyear,stringdate
    dayyofyear = cal.get_date().strftime('%j')
    stringdate = str(cal.get_date()) #2019-09-02
    i = 0
    dirpath = 'data\\'+stringdate[0:4]+'\\'+dayyofyear
    for each in selected:
        statuslabel.config (text="Dowloading data of:"+each)
        progressbar['value']=((i/len(selected))*100)
        root.update()
        i = i+1
        urlstringobs                    =                     'ftp://data-
out.unavco.org/pub/rinex/obs/'+stringdate[0:4]+'/'+dayyofyear+'/'+each+day
yofyear+'0.'+stringdate[2:4]+'d.Z'
        urlstringnav                    =                     'ftp://data-
out.unavco.org/pub/rinex/nav/'+stringdate[0:4]+'/'+dayyofyear+'/'+each+day
yofyear+'0.'+stringdate[2:4]+'n.Z'
        if not os.path.exists(dirpath):
            os.makedirs(dirpath)
        try:
```

```
            if                                                      not
os.path.exists(dirpath+'\\'+each+dayyofyear+'0.'+stringdate[2:4]+'d.Z'):
                wget.download(urlstringobs,out=dirpath)
            if                                                      not
os.path.exists(dirpath+'\\'+each+dayyofyear+'0.'+stringdate[2:4]+'n.Z'):
                wget.download(urlstringnav,out=dirpath)
        except:
            continue
    statuslabel.config (text="Downloading DCB files")
    getdcbfiles(stringdate,dirpath)
    progressbar['value']=100
    statuslabel.config (text="Download Complete")

#Processing the GPS data and getting the CSV file out of it
def process():
    def subfunction():
        starttime                                                     =
stringdate+"T"+str(starthourvar.get())+':'+str(startminvar.get())
        stoptime                                                      =
stringdate+"T"+str(stophourvar.get())+':'+str(stopminvar.get())+':45'
        timegap = gaptimevar.get()
        dirpath = 'data\\'+stringdate[0:4]+'\\'+dayyofyear
        navfiles = glob.glob(dirpath+'\\*.'+stringdate[2:4]+'n.Z')
        obsfiles = glob.glob(dirpath+'\\*.'+stringdate[2:4]+'d.Z')
        dcbfiles = glob.glob(dirpath+'\\*.DCB.Z')
        progressbar['value']=0
        statuslabel.config (text='Decompressing Files')
        root.update()
        decompressnav(navfiles)
        decompressobs(obsfiles)
        decompressdcb(dcbfiles)
        statuslabel.config (text='Decompressing Done')
        root.update()
        satdcbsintecu = {}
        p1c1dcbfile                                                   =
open(dirpath+'\\P1C1'+stringdate[2:4]+stringdate[5:7]+'.DCB','r').readline
s()[7:39]
        p1p2dcbfile                                                   =
open(dirpath+'\\P1P2'+stringdate[2:4]+stringdate[5:7]+'.DCB','r').readline
s()[7:39]
        for line1,line2 in zip(p1c1dcbfile,p1p2dcbfile):

satdcbsintecu[line1.split()[0]]=(float(line1.split()[1])+float(line2.split
()[1]))*2.85
        dotofiles = glob.glob(dirpath+'\\'+'*.'+stringdate[2:4]+'o')
        dotnfiles = glob.glob(dirpath+'\\'+'*.'+stringdate[2:4]+'n')
        csvfile = open(dirpath+'\\TECValues.csv','w')

csvfile.write('Datetime,Satnum,slanttec,elevationangle,verticaltec,lat,lon
\n')
        i = 0
        for (obs,nav) in zip(dotofiles,dotnfiles):
            progressbar['value']=((i/len(dotnfiles))*100)
            i=i+1
```

```
            statuslabel.config(text="Processing file:"+obs)
            root.update()
            csvdata                                                   =
driver(obs,nav,starttime,stoptime,timegap,satdcbsintecu)
            csvfile.write(csvdata)
        csvfile.close()
        statuslabel.config(text="Processing Done")
        progressbar['value']=100
        root.update()
    thread2 = threading.Thread(target=subfunction)
    thread2.start()

#Converting .xxn.z files to .xxn files
def decompressnav(navfiles):
    for each in navfiles:
        infile = open(each,'rb+')
        incontent = infile.read()
        outcontent = unlzw(incontent)
        outfile = open(each[0:-2],'wb')
        outfile.write(outcontent)
        outfile.close()

#Converting .DCB.z files to .DCB files
def decompressdcb(dcbfiles):
    for each in dcbfiles:
        infile = open(each,'rb+')
        incontent = infile.read()
        outcontent = unlzw(incontent)
        outfile = open(each[0:-2],'wb')
        outfile.write(outcontent)
        outfile.close()

#Converting .xxd.z files to .xxo files
def decompressobs(obsfiles):
    for each in obsfiles:
        infile = open(each,'rb+')
        incontent = infile.read()
        outcontent = unlzw(incontent)
        outfile = open(each[0:-2],'wb')
        outfile.write(outcontent)
        outfile.close()
        subprocess.call('tools\\crx2rnx -f '+each[0:-2],shell=True)

#GUI part of the program
root = tkinter.Tk()
root.title('Ionospheric map preparation software')
upperframe = tkinter.Frame(root)
upperframe.pack()
cal   =   DateEntry(upperframe,firstweekday='sunday',showweeknumbers=False,
width=12,   background='darkblue',   foreground='white',   borderwidth=2,
mindate= datetime.date(2018,1,1), maxdate= datetime.date.today())
cal.pack(padx=5, pady=10,side=tkinter.LEFT)
selectstationbutton  =  tkinter.Button(upperframe,  text="Select  Stations",
command=selectstns)
```

```
selectstationbutton.pack(padx=5, pady=10,side=tkinter.LEFT)
getdata = tkinter.Button(upperframe, text="Get Data", command=downloaddata)
getdata.pack(padx=5, pady=10,side=tkinter.LEFT)
tkinter.Label(upperframe,text="Start Time
(HH:MM)").pack(padx=5,pady=10,side=tkinter.LEFT)
hourlist = tuple(range(0,24))
starthourvar = tkinter.IntVar(root)
starthourvar.set(hourlist[0])
startHour = tkinter.OptionMenu(upperframe,starthourvar,*hourlist)
startHour.pack(padx=1, pady=10,side=tkinter.LEFT)
tkinter.Label(upperframe,text=":").pack(side=tkinter.LEFT)
minlist = tuple(range(0,60))
startminvar = tkinter.IntVar(root)
startminvar.set(minlist[0])
startMin = tkinter.OptionMenu(upperframe,startminvar,*minlist)
startMin.pack(padx=1, pady=10,side=tkinter.LEFT)
tkinter.Label(upperframe,text="Stop Time
(HH:MM)").pack(padx=5,pady=10,side=tkinter.LEFT)
stophourvar = tkinter.IntVar(root)
stophourvar.set(hourlist[-1])
stopHour = tkinter.OptionMenu(upperframe,stophourvar,*hourlist)
stopHour.pack(padx=1, pady=10,side=tkinter.LEFT)
tkinter.Label(upperframe,text=":").pack(side=tkinter.LEFT)
stopminvar = tkinter.DoubleVar(root)
stopminvar.set(minlist[-1])
stopMin = tkinter.OptionMenu(upperframe,stopminvar,*minlist)
stopMin.pack(padx=1, pady=10,side=tkinter.LEFT)
tkinter.Label(upperframe,text="Time
Interval").pack(padx=5,pady=10,side=tkinter.LEFT)
gaptimetist = ('0.25','0.5','1','2','5','10','20','30','60','120')
gaptimevar = tkinter.StringVar(root)
gaptimevar.set(gaptimetist[0])
gapTime = tkinter.OptionMenu(upperframe,gaptimevar,*gaptimetist)
gapTime.pack(padx=1, pady=10,side=tkinter.LEFT)
tkinter.Label(upperframe,text="Minutes").pack(padx=5,pady=10,side=tkinter.
LEFT)
processdata = tkinter.Button(upperframe, text="Process Data",
command=process)
processdata.pack(padx=5, pady=10,side=tkinter.LEFT)
plotdata = tkinter.Button(upperframe, text="Generate TEC Maps",
command=plotter)
plotdata.pack(padx=5, pady=10,side=tkinter.LEFT)
lowerframe = tkinter.Frame(root)
lowerframe.pack(fill='x')
progressbar =
tkinter.ttk.Progressbar(lowerframe,orient='horizontal',length=100,mode='de
terminate')
progressbar.pack(padx=5, pady=5, fill='x')
statuslabel = tkinter.Label(lowerframe,text="Ready")
statuslabel.pack(padx=5, pady=5, fill='x')
root.mainloop()
```

## 7.4. ANNEX 4

Python code of tecvalues.py file. It is the code that handles the calculation part:

```
import georinex as gr
import math
import numpy as np
import pymap3d as pm
import glob

def getsatElev(recposgeo,satpos):
      slat,slon,shei=  pm.ecef2geodetic(satpos[0],  satpos[1],  satpos[2],
deg=True)
      az,el,r   =   pm.geodetic2aer(slat,   slon,   shei,   recposgeo[0],
recposgeo[1], recposgeo[2], deg=True)
      return (el,az)

def getVTEC(stec,elev):
      rofearth = 6371000
      hofip = 400000
      mapfunc                        =                        math.sqrt(1-
((rofearth*math.cos(math.radians(elev)))/(rofearth+hofip))**2)
      vtec = stec*mapfunc
      return vtec

def getIPPLattLon(recvpos,eleaz):
      rofearth = 6371000
      hofip = 400000
      p               =               (math.pi/2)-math.radians(eleaz[0])-
math.asin((rofearth*math.cos(math.radians(eleaz[0])))/(rofearth+hofip))
      ipplat                                                             =
math.degrees(math.asin(math.sin(math.radians(recvpos[0]))*math.cos(p)+math
.cos(math.radians(recvpos[0]))*math.sin(p)*math.cos(math.radians(eleaz[1])
)))
      ipplon                                                             =
recvpos[1]+math.degrees(math.asin(math.sin(p)*math.sin(math.radians(eleaz[
1])/math.cos(math.radians(recvpos[0])))))
      ipplatlon = (ipplat,ipplon)
      return ipplatlon


def getSatXYZ(nav,obssv,obstime):
      xyz = tuple()

      #Constants
      GM = 3.986004418e14
      EMAV = 7.2921151467e-5
      svdata = nav.sel(sv=obssv).dropna(dim='time')
      timedifferences         =         [abs((t-obstime.to_datetime64())/
np.timedelta64(1,'s')) for t in svdata.coords['time'].values]
      epochtime                                                          =
svdata.coords['time'].values[timedifferences.index(min(timedifferences))]
      finaldata = svdata.sel(time=epochtime)
      timeeph = finaldata['Toe']
```

```
    t = getGpsTime(obstime)-timeeph

    #Keplerian Elements
    M0 = finaldata['M0']
    sqrtA = finaldata['sqrtA']
    deltaN = finaldata['DeltaN']
    ecc = finaldata['Eccentricity']
    incli = finaldata['Io']
    rateofIncli = finaldata['IDOT']
    argofperigee = finaldata['omega']
    rightacc = finaldata['Omega0']
    rateofRightAcc = finaldata['OmegaDot']

    #coefficients for correction
    cuc = finaldata['Cuc']
    cus = finaldata['Cus']
    crc = finaldata['Crc']
    crs = finaldata['Crs']
    cic = finaldata['Cic']
    cis = finaldata['Cis']

    #computation for anomalies
    meanAmomaly = M0 + t*(deltaN+ math.sqrt(GM/sqrtA**6))
    ecentricAnomaly = solveIter(meanAmomaly,ecc)
    trueAnomaly               =               math.atan((math.sqrt(1-
ecc**2)*math.sin(ecentricAnomaly))/(math.cos(ecentricAnomaly)-ecc))

    #computation for pertubrations
    phik = argofperigee+trueAnomaly
    argofperigee_comp                                               =
argofperigee+cuc*math.cos(2*phik)+cus*math.sin(2*phik)
    radialDistance                         =                        (1-
ecc*math.cos(ecentricAnomaly))*(sqrtA**2)+crc*math.cos(2*phik)+crs*math.si
n(2*phik)
    inclination                                                     =
incli+rateofIncli*t+cic*math.cos(2*phik)+cis*math.sin(2*phik)

    #computation for right accension
    rightacc_comp = rightacc+t*(rateofRightAcc-EMAV)-(EMAV*timeeph)
    cosra = math.cos(rightacc_comp)
    sinra = math.sin(rightacc_comp)
    cosaop = math.cos(argofperigee_comp)
    sinaop = math.sin(argofperigee_comp)
    cosi = math.cos(inclination)
    sini = math.sin(inclination)
    cosVk = math.cos(meanAmomaly)
    sinVk = math.sin(meanAmomaly)
    smallr = np.array([radialDistance*cosVk,radialDistance*sinVk,0])
    capitalR = np.array([[cosra*cosaop-sinra*sinaop*sini,-1*cosra*sinaop-
sinra*cosaop*cosi,sinaop*sini],
                      [sinra*cosaop+cosra*sinaop*cosi,-
1*sinra*sinaop+cosra*cosaop*cosi,-1*cosra*sini],
                      [sinaop*sini,cosaop*sini,cosi]])
    coordsmatrix = np.matmul(capitalR,smallr)
```

```
      xyz = (coordsmatrix[0],coordsmatrix[1],coordsmatrix[2])
      return xyz

def getGpsTime(dt):
      """_getGpsTime returns gps time (seconds since midnight Sat/Sun) for
a datetime
      """
      total = 0
      days = (dt.weekday()+ 1) % 7 # this makes Sunday = 0, Monday = 1, etc.
      total += days*3600*24
      total += dt.hour * 3600
      total += dt.minute * 60
      total += dt.second
      return(total)

def solveIter(mu,e):
      """
      __solvIter returns an iterative solution for Ek
      Mk = Ek - e sin(Ek)
      adapted to accept vectors instead of single values
      from Bill Rideout's tec.py
      """
      thisStart = mu-1.01*e
      thisEnd = mu + 1.01*e
      bestGuess = 0

      for i in range(5):
            minErr = 10000
            for j in range(5):
                  thisGuess = thisStart + j*(thisEnd-thisStart)/10.0
                  thisErr = abs(mu - thisGuess + e*np.sin(thisGuess))
                  if (thisErr<minErr):
                        minErr = thisErr
                        bestGuess = thisGuess

            # reset for next loop
            thisRange = thisEnd - thisStart
            thisStart = bestGuess - thisRange/10.0
            thisEnd = bestGuess + thisRange/10.0

      return(bestGuess)

def driver(obsfile,navfile,start,stop,timegap,satdcb):
      obs = gr.load(obsfile, meas=['L1','L2','C1','P2'], tlim=[start,stop])
      nav = gr.load(navfile)
      finalresult = ''
      testing = gr.load(obsfile)
      testingpoints = testing.coords['time'].values
      if not len(testingpoints) == 5760:
            return finalresult
      points = obs.coords['time'].values
      GPSsats = ('G01', 'G02', 'G03', 'G05', 'G06', 'G07', 'G08', 'G09',
'G10', 'G11',
       'G12', 'G13', 'G14', 'G15', 'G16', 'G17', 'G18', 'G19', 'G20', 'G21',
```

```
        'G22', 'G23', 'G24', 'G25', 'G26', 'G27', 'G28', 'G29', 'G30', 'G31',
        'G32')
      for eachepoch in points[::int(float(timegap)*4)]:
            oneepochl1 = obs['L1'].sel(time = eachepoch).dropna(dim='sv')
            oneepochl2 = obs['L2'].sel(time = eachepoch).dropna(dim='sv')
            oneepochc1 = obs['C1'].sel(time = eachepoch).dropna(dim='sv')
            oneepochp2 = obs['P2'].sel(time = eachepoch).dropna(dim='sv')

            l1satset = set(oneepochl1.coords['sv'].values)
            l2satset = set(oneepochl2.coords['sv'].values)
            c1satset = set(oneepochc1.coords['sv'].values)
            p2satset = set(oneepochp2.coords['sv'].values)
            commonsats = l1satset.intersection(l2satset)
            commonsats = commonsats.intersection(c1satset)
            commonsats = commonsats.intersection(p2satset)
            commonsats = commonsats.intersection(GPSsats)

            for eachsv in commonsats:
                  stamp = eachepoch.strftime("%d-%b-%Y (%H:%M:%S)")
                  satelliteno = eachsv
                  l1value                                                    =
oneepochl1.values[oneepochl1.coords['sv'].values.tolist().index(eachsv)]
                  l2value                                                    =
oneepochl2.values[oneepochl2.coords['sv'].values.tolist().index(eachsv)]
                  c1value                                                    =
oneepochc1.values[oneepochc1.coords['sv'].values.tolist().index(eachsv)]
                  p2value                                                    =
oneepochp2.values[oneepochp2.coords['sv'].values.tolist().index(eachsv)]
                  stec   =   9.5172816799473472   *(p2value   -   c1value)   +
satdcb[eachsv]
                  satCoord = getSatXYZ(nav,eachsv,eachepoch)
                  satelevaz = getsatElev(obs.position_geodetic,satCoord)
                  vtec = getVTEC(stec,satelevaz[0])
                  if vtec < 0:
                        continue
                  elif vtec >70:
                        continue
                  ipplatlon                                                  =
getIPPLattLon(obs.position_geodetic,satelevaz)
                  result                                                     =
(stamp+","+satelliteno+","+str(stec)+","+str(satelevaz[0])+","+str(vtec)+"
,"+str(ipplatlon[0])+","+str(ipplatlon[1])+"\n")
                  finalresult = finalresult+result
      return finalresult
```

## 7.5. ANNEX 5

Python code of IonexWriter.py file. It handles the IONEX generation part of the project:

```
import numpy as np
from scipy.interpolate import griddata
from datetime import datetime

def writeionex(dataframe,dirpath):
    timelist = dataframe['Datetime'].tolist()
    timelist = set(timelist)
    timelist = sorted(timelist)
    ionexfile                                                           =
open(dirpath+'\\nepal'+timelist[0].strftime('%j')+'0.'+timelist[0].strftim
e('%y')+'i','w+')
    header = """    1.0          IONOSPHERE MAPS    GNSS             IONEX
VERSION / TYPE
FinalProjV1         KU                   {today}     PGM / RUN BY / DATE
Ionospheric Model generated using CORS data of Nepal        COMMENT
Regional Ionospheric Model of Nepal. This is the product    DESCRIPTION
of final year project of Undergraduate at Kathmandu         DESCRIPTION
University                                                  DESCRIPTION
Contact address: binabhdevkota@gmail.com                    DESCRIPTION
  {start}                  EPOCH OF FIRST MAP
  {end}                  EPOCH OF LAST MAP
  {interval}                                                      INTERVAL
    {nofmaps}                                                         # OF
MAPS IN FILE
    COSZ                                                  MAPPING FUNCTION
     0.0                                                  ELEVATION CUTOFF
Pseudorange values with DCB correction                         OBSERVABLES
USED
  6371.0                                                    BASE RADIUS
     2                                                      MAP DIMENSION
   400.0 400.0   0.0                                         HGT1 / HGT2 /
DHGT
    80.0 89.0  0.5                                           LAT1 / LAT2 /
DLAT
    26.0 31.0  0.5                                           LON1 / LON2 /
DLON
    -1                                                      EXPONENT
TEC/RMS values in 0.1 TECU; 9999, if no value available     COMMENT
                                                            END         OF
HEADER\n""".format(today        =        datetime.now().strftime("%d-%b-%y
%H:%M"),start=timelist[0].strftime("%Y       %m       %d       %H       %M
%S"),end=timelist[-1].strftime("%Y        %m        %d        %H        %M
%S"),interval=(timelist[1]-
timelist[0]).total_seconds(),nofmaps=len(timelist))
    ionexfile.write(header)
    for each in timelist:
        subdataframe = dataframe.loc[dataframe['Datetime'] == each]
        y = subdataframe['lat'].to_numpy()
        x = subdataframe['lon'].to_numpy()
        z = subdataframe['verticaltec'].to_numpy()
```

```
        xi = np.linspace(80,89,20)
        yi = np.linspace(26,31,12)
        zi    =    griddata((x,    y),    z,    (xi[None,:],    yi[:,None]),
method='linear',rescale=True)
        np.nan_to_num(zi,copy=False)
        zi[zi == 0] = 9999
        mapstart      =      """                                   {mapnum}
START OF TEC MAP
  {epoch}                                           EPOCH   OF   CURRENT
MAP\n""".format(epoch=each.strftime("%Y       %m       %d       %H       %M
%S"),mapnum=timelist.index(each)+1)
        ionexfile.write(mapstart)
        i=0
        for row in zi:
            datasection  =  """         {lat}  80.0  89.0       0.5  400.0
LAT/LON1/LON2/DLON/H
    {v1}    {v2}    {v3}    {v4}    {v5}    {v6}    {v7}    {v8}    {v9}
{v10}    {v11}    {v12}    {v13}    {v14}    {v15}    {v16}
    {v17}                           {v18}                           {v19}
{v20}\n""".format(lat=26+0.5*i,v1=int(row[0]),v2=int(row[1]),v3=int(row[2]
),v4=int(row[3]),v5=int(row[4]),v6=int(row[5]),v7=int(row[6]),v8=int(row[7
]),v9=int(row[8]),v10=int(row[9]),v11=int(row[10]),v12=int(row[11]),v13=in
t(row[12]),v14=int(row[13]),v15=int(row[14]),v16=int(row[15]),v17=int(row[
16]),v18=int(row[17]),v19=int(row[18]),v20=int(row[19]))
            i=i+1
            ionexfile.write(datasection)
        mapend      =      """                                     {mapnum}
END OF TEC MAP      \n""".format(mapnum=timelist.index(each)+1)
        ionexfile.write(mapend)
    fileend = """                                                          END
OF FILE         """
    ionexfile.write(fileend)
    ionexfile.close()
```

## 7.6. ANNEX 6

Screen captures of using the tool:

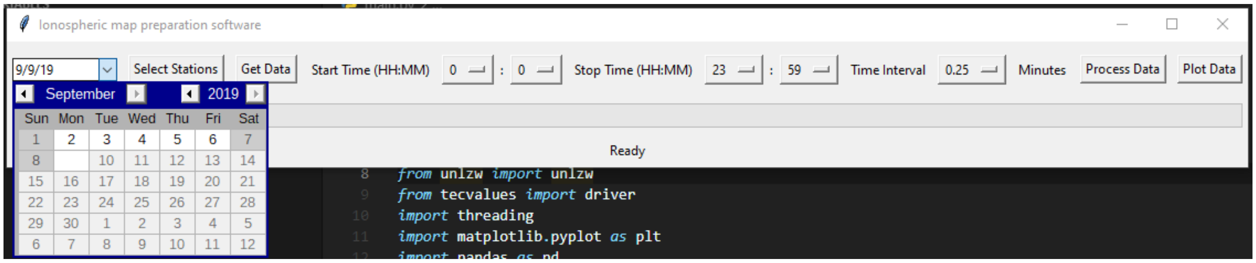



Figure 16: Date selection for Ionospheric map generation

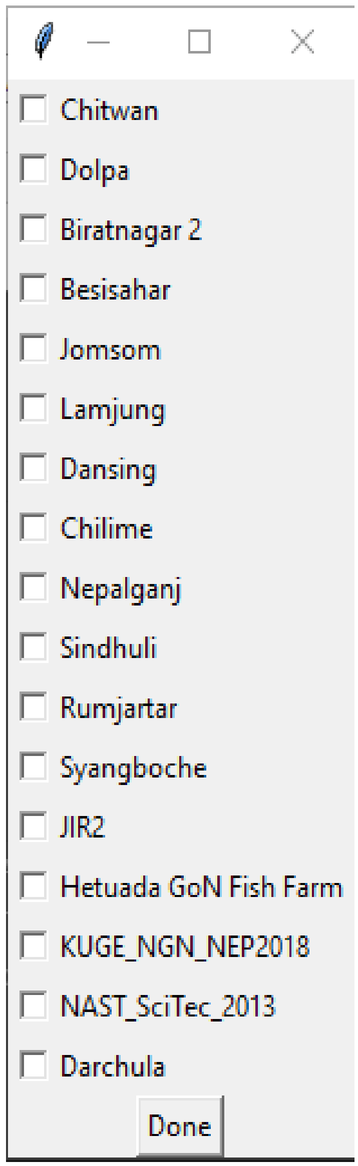



Figure 15: Station selection for Ionospheric map generation

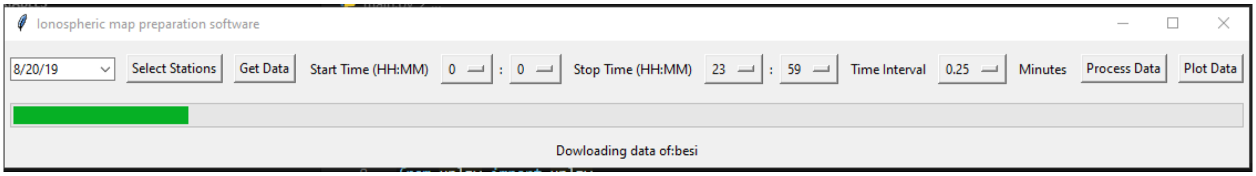



Figure 17: Data Downloading from internet

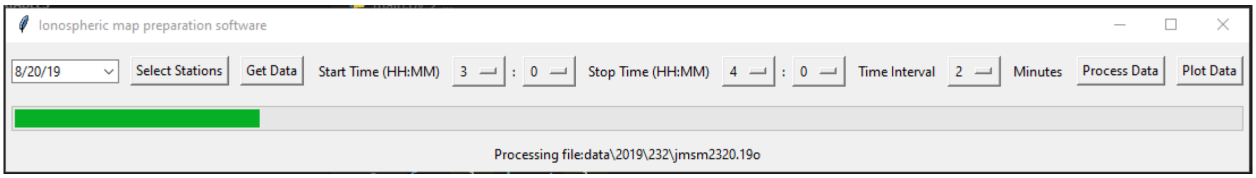



Figure 18: Data processing for TEC map generation