

# First results of an open-source based program for acquiring long time series of ionospheric parameters from the EISCAT Madrigal database.

V. Belyey and C. La Hoz

University of Tromsø, Tromsø, Norway.

## Abstract.

A software package has been designed and implemented to retrieve and process data from the EISCAT Madrigal database. The software package is an important tool to achieve the objectives of the EISCAT\_3D Work Package 10, namely to explore the potential of the new radar to carry out climatic studies based on examination of long-time data series and to construct models of ionospheric electron density relevant to improving the accuracy of navigation and positioning parameters produced by the GPS and Galileo satellite constellations. Included are examples produced by the tool that show daily, seasonal and solar cycle variations of electron density measured by the EISCAT radars over two solar cycles. Work is already under way to use the data acquired and processed by the tool to assess current ionospheric models, including those used to make corrections to positioning parameters produced by the Galileo and GPS satellite constellations.

## 1. Introduction

EISCAT\_3D is an EU supported project devoted to design of a new generation incoherent scatter radar (ISR). The project was launched in 2005. One of its work packages (WP10) is to study the feasibility of using the data obtained by the ISR in new non-traditional ways. Objectives of the work package, as formulated in the description of work, are the use of long and continuous time series of physical parameters in the polar upper atmosphere for:

- Climatic studies, that is, the climatic conditions in the upper parts of the atmosphere
- Corrections for ionospheric disturbances in satellite data especially Galileo and GPS
- Space weather studies
- Correction of SAR remote sensing data

In order to develop a technique for accomplishing the objectives it is necessary to have access to databases that provide information about ionospheric parameters similar to that to be obtained by the new incoherent scatter radar. Most natural for the task at hand is to use the Madrigal database (<http://www.openmadrigal.org>). The Madrigal system is capable of serving archival and real-time data in a variety of formats and from a wide range of instruments including the existing EISCAT incoherent scatter radar systems. The EISCAT data available at Madrigal cover virtually the entire periods of routine observations since the radars were put into operation since 1984, 1990, and 1997 for UHF tristatic system, VHF, and EISCAT Svalbard (ESR) radars, respectively. A software package designed to be a tool to acquire and process the EISCAT data in Madrigal format is described in this paper.

## 2. Software organization

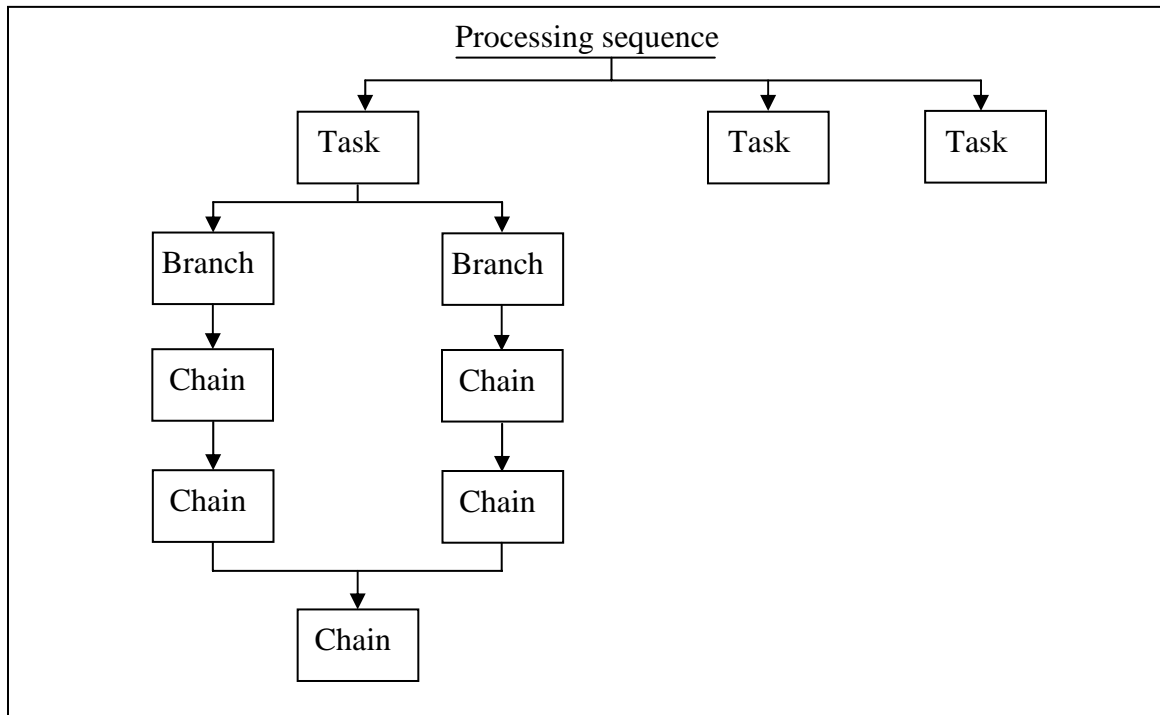
To develop the software as much user-friendly and flexible as possible, a set of requirements was formulated as follows:

- Internet-compatible
- modular
- object oriented
- user upgradeable
- open source- and standards based
- platform independent
- non-proprietary
- implementing metadata formats

The python programming language (<http://python.org/>) was found to be satisfactory to accomplish the task with requirements as listed above.

The software is in fact a specialized interpreter of a command file (script) prepared in text format and easily editable by the user. The script determines the set of computational units (CU) that comprise the entire processing program. A computational unit, in this software context, is a class descending from a basic TTask class determined in the package that provides all the necessary member functions for the descendant class to be run in proper succession as well as mechanisms for convenient exchange of data and parameters between different CUs. Thus every computational unit implements an independent processing algorithm that is supplied (if necessary) with input data and produces output data. The execution of the algorithm is controlled by a set of parameters determined by the user. The basic TTask class supplies containers for the parameters and output data which can be accessed by any other computational unit.

The logical order of execution of different CUs comprising a processing sequence is determined by the user by declaring a CU as “Task”, “Branch” or “Chain” (see the functional diagram in Figure 1). All computational units are added to the script using one of the three keywords. The processing sequence (program) described by the script consists of one or more tasks. Tasks are logically independent parts of the processing sequence. In fact, several different tasks can run either as a single script or as several separate scripts. Tasks are executed in parallel and independently of each other, thus increasing processing efficiency when using multi-CPU computers.



**Figure 1.** Functional diagram of a processing sequence.

An example can be a script containing three separate tasks acquiring Madrigal data from different EISCAT radars: UHF, VHF, and ESR.

Each task consists of zero or more branches. Branches are parts of a processing task with relatively weak logical dependence. They also can be run simultaneously as separate independent threads if requested by the user. Usually, branches share the same input data but process them in different ways. For example, the EISCAT Madrigal database supplies records of altitude-dependent ionospheric parameters (electron density, ion and electron temperatures, etc.). One of the two branches shown in Fig. 1 may produce time-integrated altitude profiles of the parameters while the other one may compute altitude-independent parameters such as peak density altitude or total electron content (TEC). The output data from the branches can be merged to form the input data for another computational unit, for instance storing the results in a file.

Finally, a branch (or a task if there are no branches) consists of zero or more chains. Chains are executed sequentially, usually processing portions of data in a cycle. An example of a chain can be CUs for data acquisition, filtering, integration, and saving the results.

A disadvantage of python, which is an interpreter, is that it is rather slow in computationally intensive operations (Fourier analysis, numeric integration algorithms, etc.). On the other hand, there are several python modules developed by members of the open-source community that provide efficient and fast routines for scientific computations (Numeric, <http://people.csail.mit.edu/jrennie/python/numeric/>, ScientificPython, <https://sourcesup.cru.fr/projects/scientific-py/>, and many others). Another solution is to exploit advantages of other programming languages and software packages which have well-developed routines for that kind of computations. So, our software should provide mechanisms for efficient control of non-python applications and fast data exchange between them.

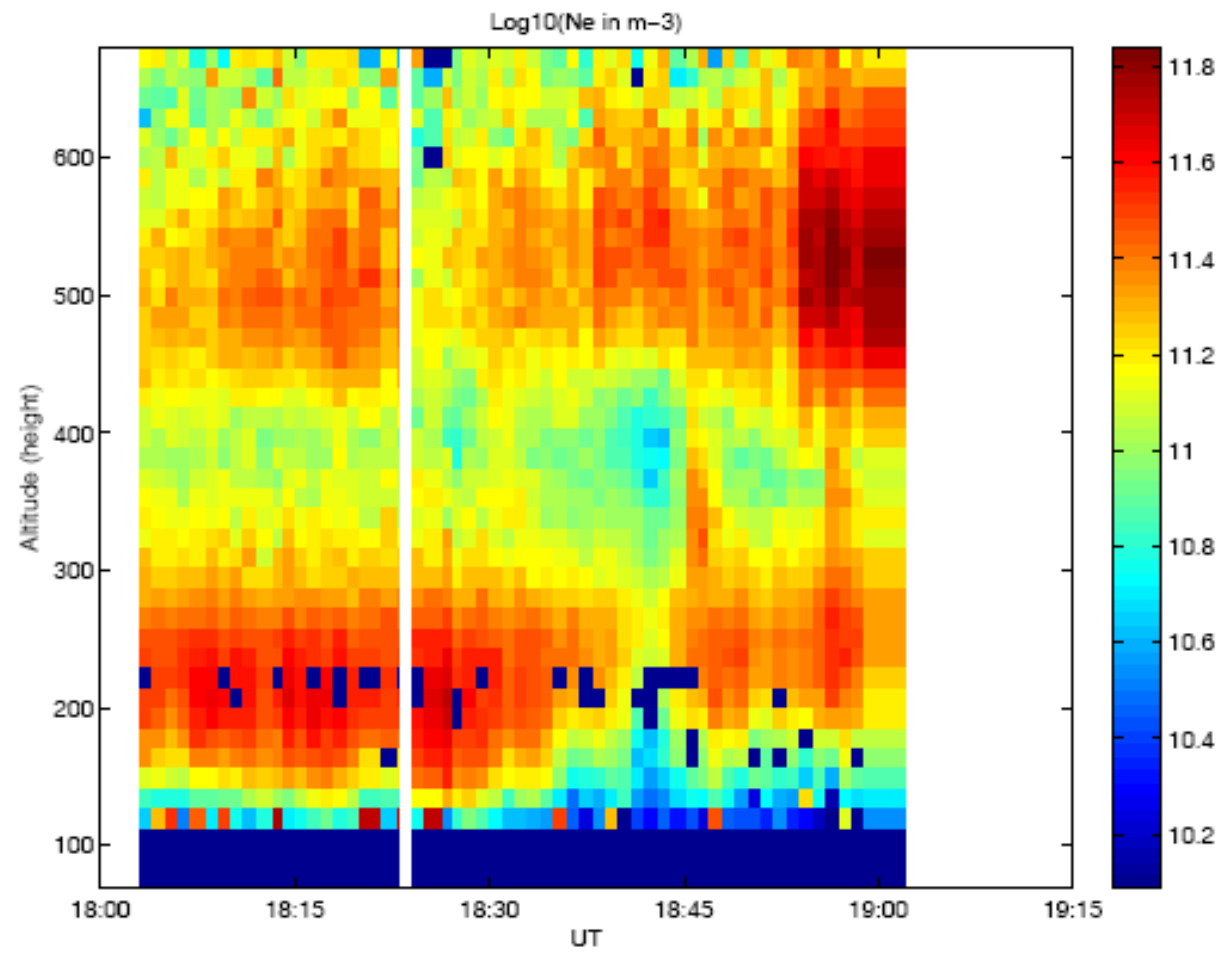
A simple example of a script that acquires raw data from the EISCAT Madrigal web site is shown below.

```

%%%%%%%%%%
Task: Madrigal.GetData
Parameter: 'Site': 'Tromso UHF'
Parameter: 'TimeSpan': [datetime(2002, 11, 03, 18, 00), datetime(2002, 11, 03, 19, 00)]
Parameter: 'DataList': ['Altitude', 'Ne']
%%%
Chain: Generic.SaveData
Parameter: 'File': '_Madrigal-data.txt'
%%%
Chain: Generic.SystemCommand
Parameter: 'Command': 'MATLAB.exe -nosplash -nodesktop -r "PlotMadrigal(\'"+self.Compile(['$File'])+\'\',0)'
%%%
EndChain:
%%%%%%%%%%

```

Commands in the script request data from the Tromsø UHF radar for a given interval, select the ionospheric parameters of interest (altitude and electron density). The results are saved in a text file and plotted using a simple matlab program that produces a postscript file.



**Figure 2.** Example of altitude profiles of electron density acquired directly from Madrigal database.

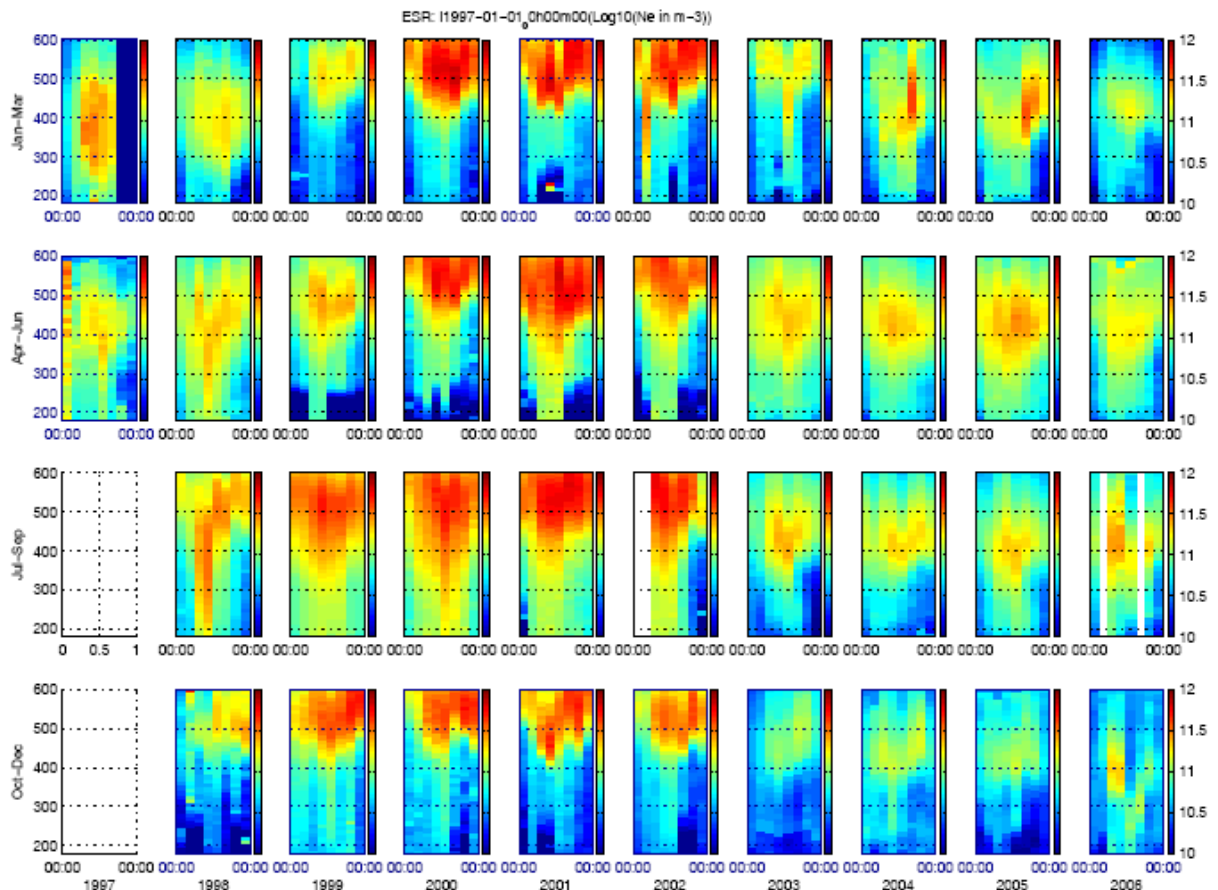
The first computational unit (Madrigal.GetData) is declared as a task. It simulates the process of manual search for the data belonging to the requested time span by downloading the Madrigal web pages for the given date, selecting the appropriate records, and, finally, reading the records in text format. All the subsequent CUs are declared as chains and thus are executed sequentially. The last CU (calling the matlab plotting program) is an example of a non-python procedure used in the processing sequence.

### 3. Examples of Madrigal data processing

The software package was used to download and process all the Madrigal data obtained at the three EISCAT radar sites: UHF, VHF, and ESR. First test runs using integration of the data downloaded directly via the Internet proved to be too slow. Up to 98% of the total computation time was spent in data transfers. To provide faster processing, a local Madrigal database was created. The database represents a copy of the raw Madrigal data stored in a compact binary format. It allowed substantial enhancement of the processing speed providing better flexibility of data integration and avoiding possible (and probable) data transfer errors if the data had been acquired through the Internet.

The Madrigal data are collected from many different (common and special) radar programs and virtually every program supplies ionospheric data with different technical parameters such as altitude span, time and altitude resolutions, etc. Also, the data are sparse since the radars were seldom operated continuously. A special integration algorithm was developed to integrate together that kind of temporally and spatially non-uniform data. The algorithm allows integrating the long time series preserving characteristic daily and seasonal variations. Input raw data are integrated for several time intervals (1-3 hours) within a day for all the days belonging to a given seasonal period (3 months). The results of integration were recorded for subsequent visualization and processing.

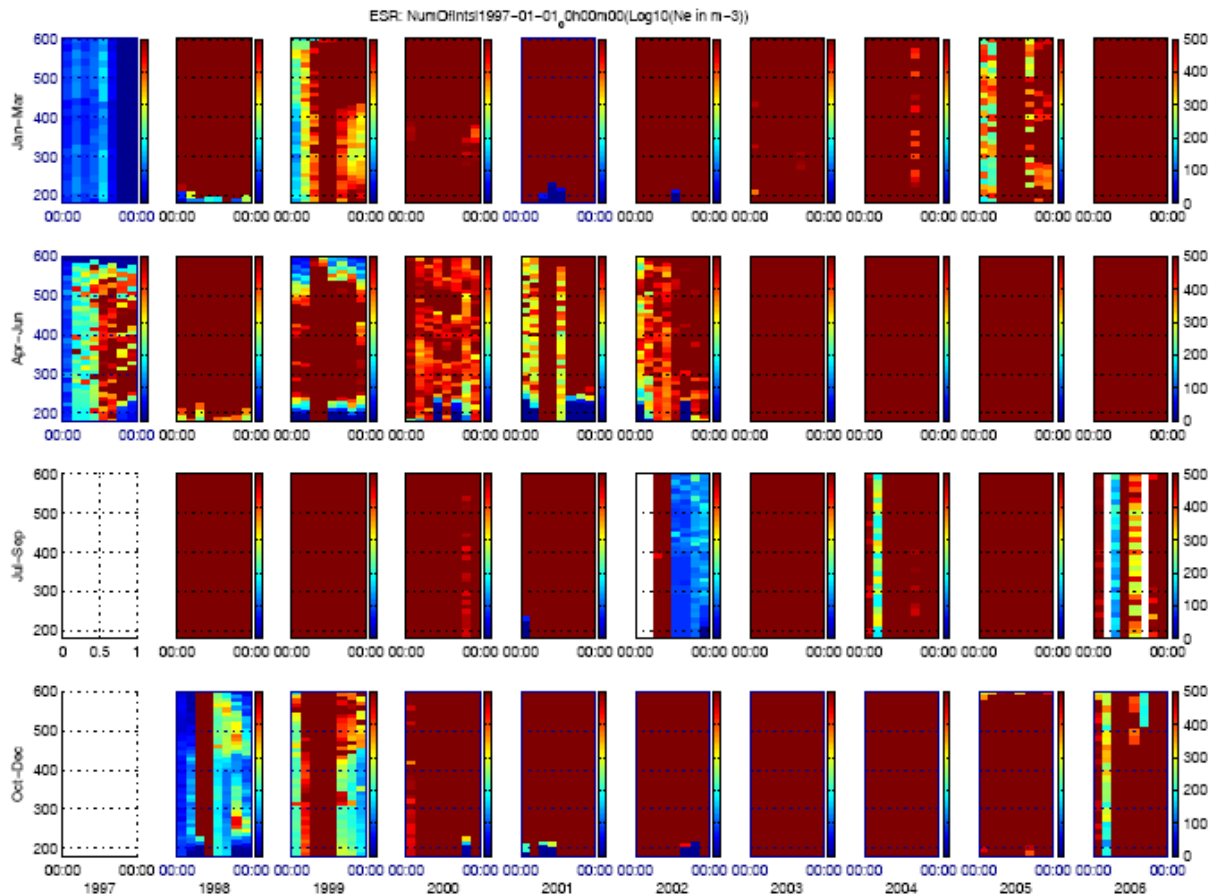
An example of the integration results is shown in Fig. 3. Electron density profiles from the EISCAT Svalbard radar were integrated in 3-hour daily intervals for 3-month seasonal periods (columns and rows in the figure, respectively). Blank panels and regions indicate missing data. Erroneous data that sometimes appear among Madrigal records were filtered out of the integration. The filtering CU excluded, for instance, records with too high electron densities (over  $10^{13} \text{ m}^{-3}$ ) or non-zero density data for altitudes below 50 km. Higher electron densities as well as higher F-layer altitudes during maximum solar activity are evident for 2001. Seasonal dependence is also visible manifested in higher densities during summer months.



**Figure 3.** Altitude profiles of electron density ( $\log_{10} N_e, \text{ m}^{-3}$ ) measured by ESR and integrated over 3 hours in a day for 3 months. Columns represent seasonal variations within a year (1997-2006).

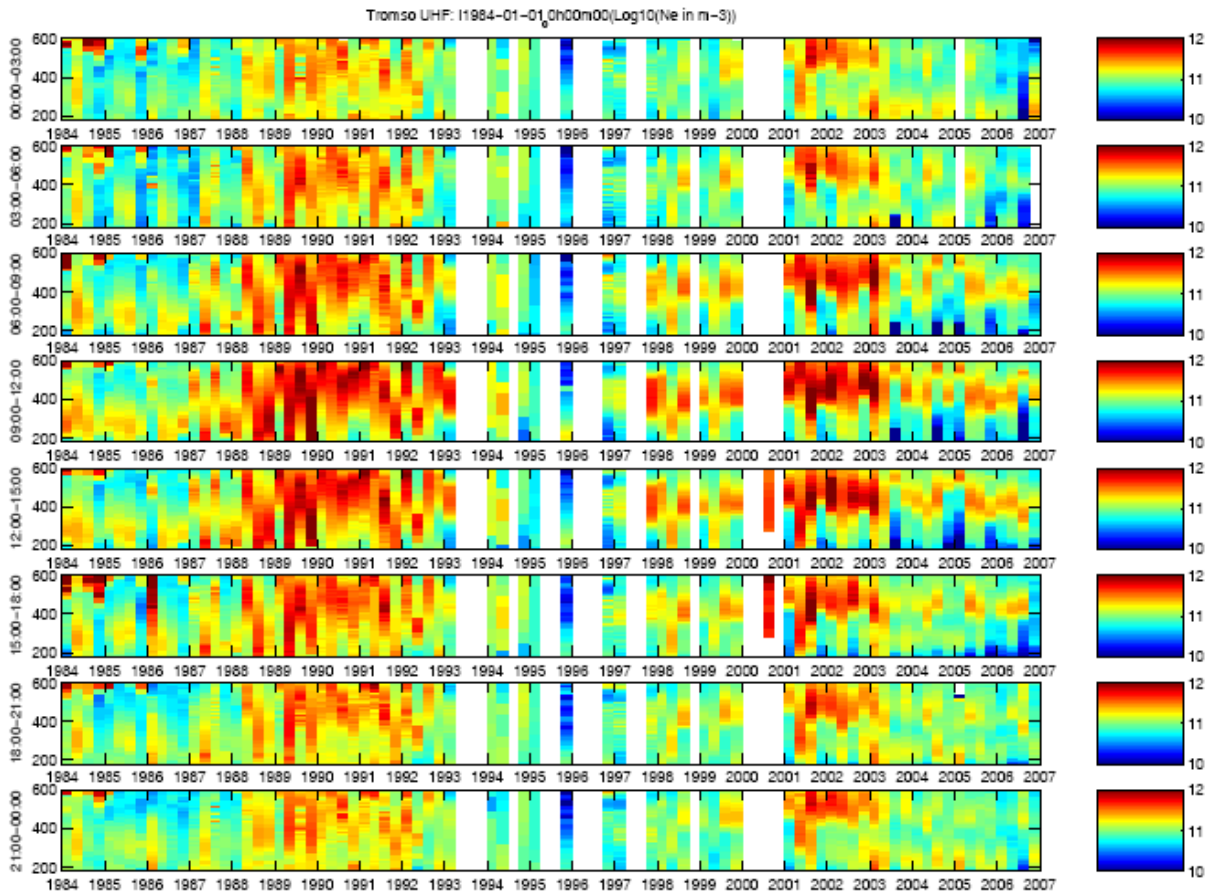
Figure 4 shows the number of integrations for every time interval and altitude range corresponding to those in figure 3. Although the maximum number of integrations in the plot is about 12000, the color scales were intentionally set to 0-500 so that the numbers exceeding 500 (approximate number of 10-minute intervals

present in a 3-hour interval during 3 months) appear in the same color and correspond to conditionally satisfactory integration.



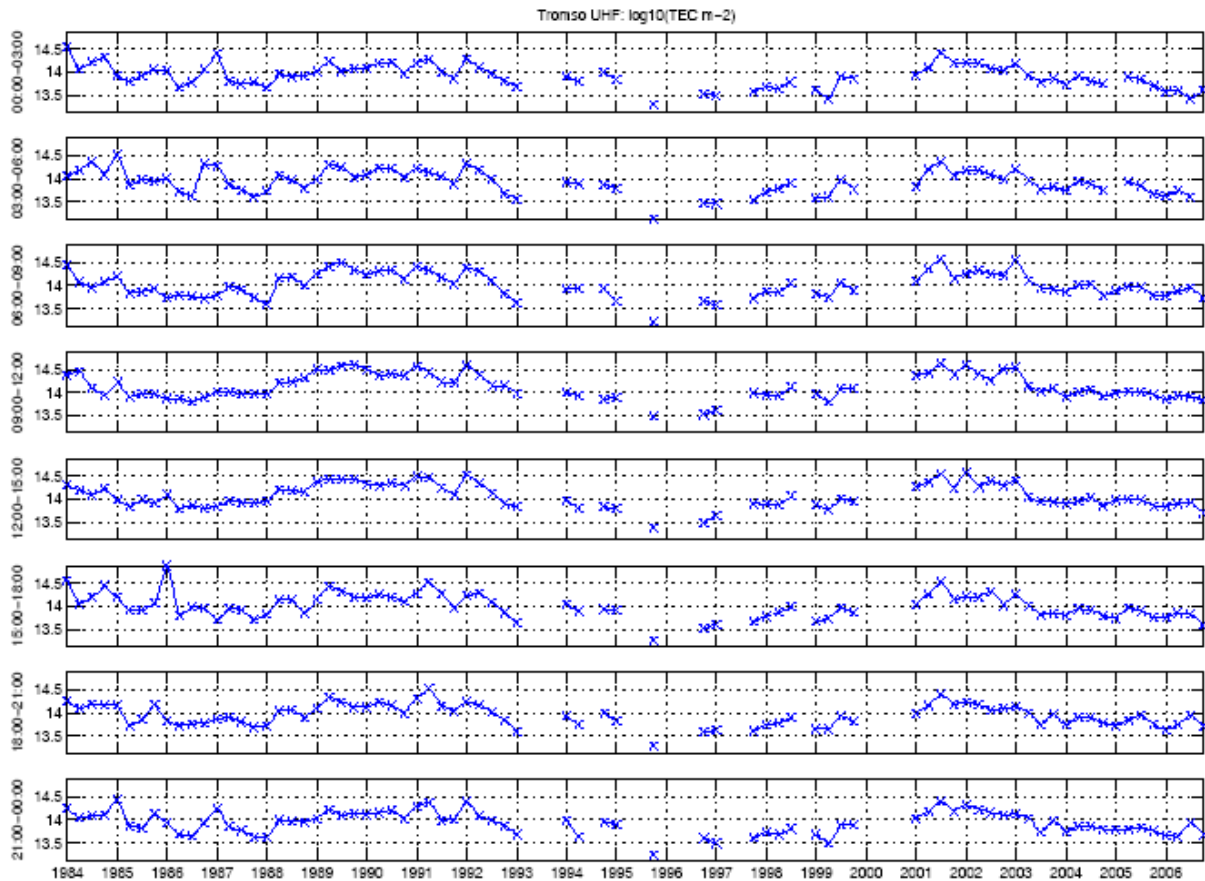
**Figure 4.** Numbers of integrations of the data presented in figure 3.

Integrated electron densities from the Tromsø UHF radar are shown in figure 5. Each of the 8 panels represent one 3-hour integration interval for the period 1984-2007 and contains two 11-year solar cycles. Again, enhanced density and higher F-layer altitudes are clearly seen during the periods of maximum solar activity (1990 and 2001). Diurnal variations are also evident. Higher electron densities correspond to near-noon integration intervals.



**Figure 5.** Altitude profiles of electron density ( $\log_{10} N_e, \text{m}^{-3}$ ) measured by Tromsø UHF radar and integrated over 3 hours in a day for 3 months. Panels correspond to the 3-hour integration intervals.

Figure 6 shows total electron content computed from the data shown in figure 5. The variations due to the solar activity cycle are also visible.



**Figure 6.** Total electron content ( $\log_{10} \text{TEC}, \text{m}^{-2}$ ) computed from the data presented in figure 5.

#### 4. Outlook.

The examples described in this report demonstrate the potential of the software tool to acquire and process incoherent scatter data from large databases. Efforts are underway to utilise the tool to assess the accuracy of ionospheric models, such as the International Reference Ionosphere (IRI) and other models used to make corrections to positioning parameters produced by the GPS and Galileo satellite constellations.