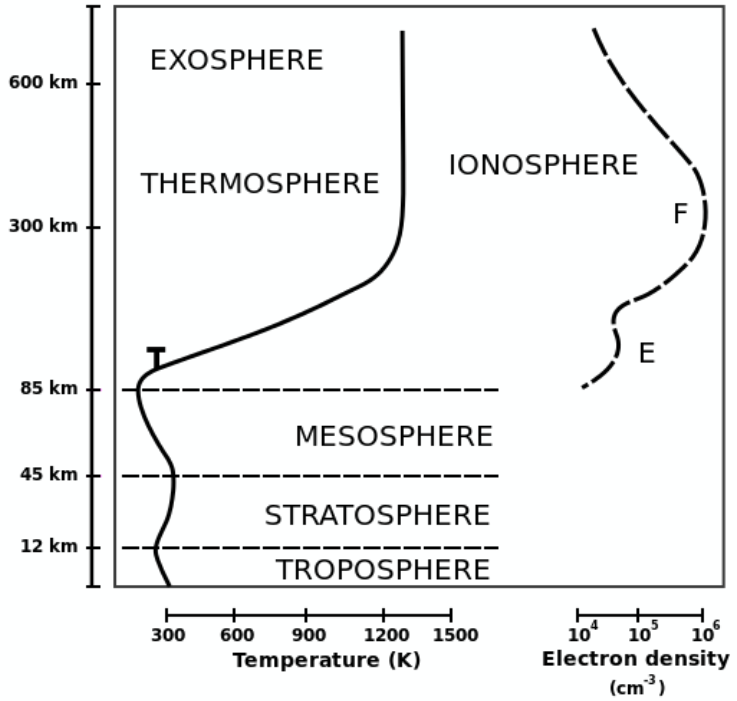
VTEC Swarm - SMOS comparison

Lorenzo Trenchi & Veronica Gonzalez Gambau

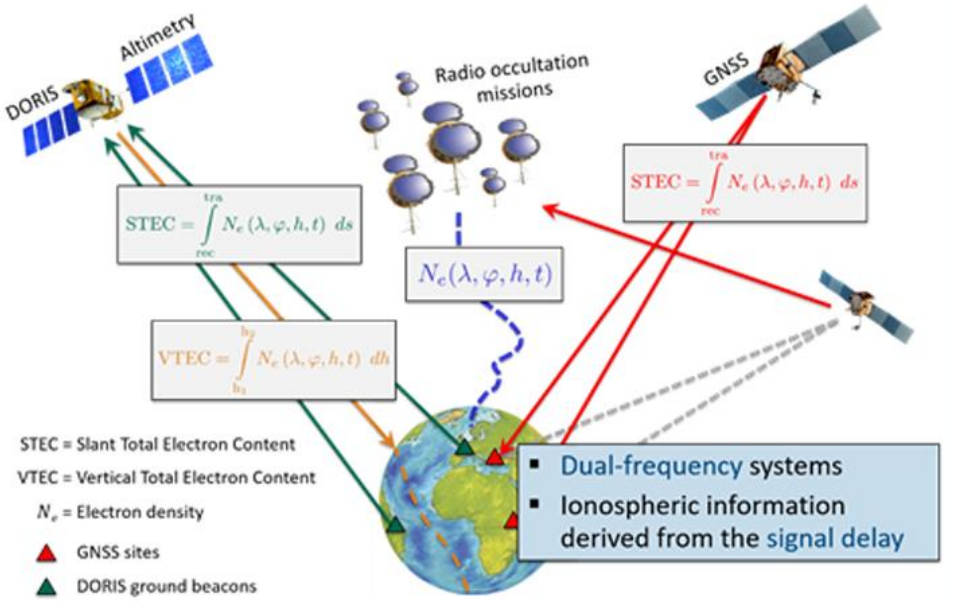
27/10/2023

Earth's ionosphere – Total Electron Content (TEC)



The main mechanism generating charged particles in the ionosphere is UV solar radiation => strong day – night variability.

Other sources are: X rays, energetic particles, cosmic rays.



TEC: integral measure of electron density in the ionosphere: Slanted TEC (along the line of the signal), V-TEC (projected in the radial direction).

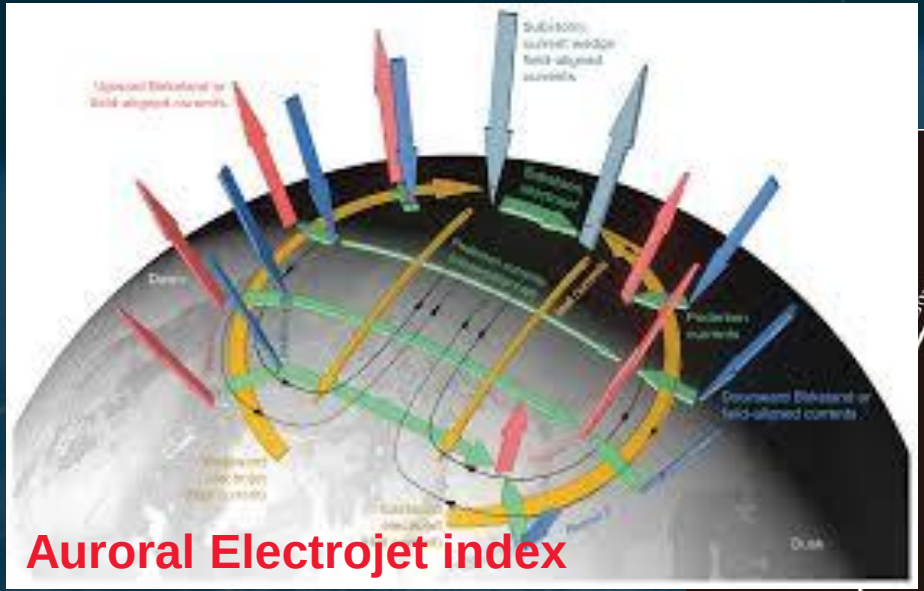
from GNSS signals (from ground and spacecraft receivers – i.e. Swarm), from microwave interferometric radiometer (i.e. SMOS)

Swarm & SMOS explore different ionospheric layers:

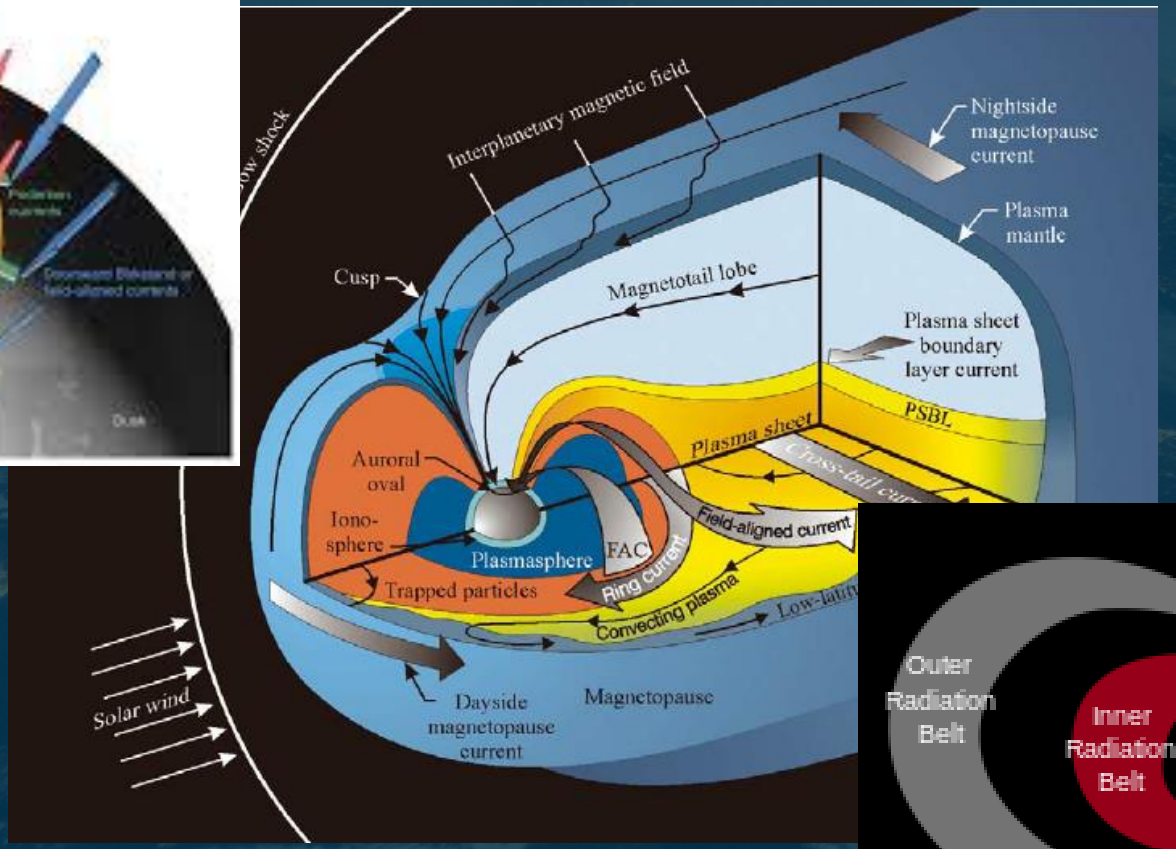
- Swarm from the s/C altitude (~500 km) upward
- SMOS from the s/c altitude (~750 km) downward.

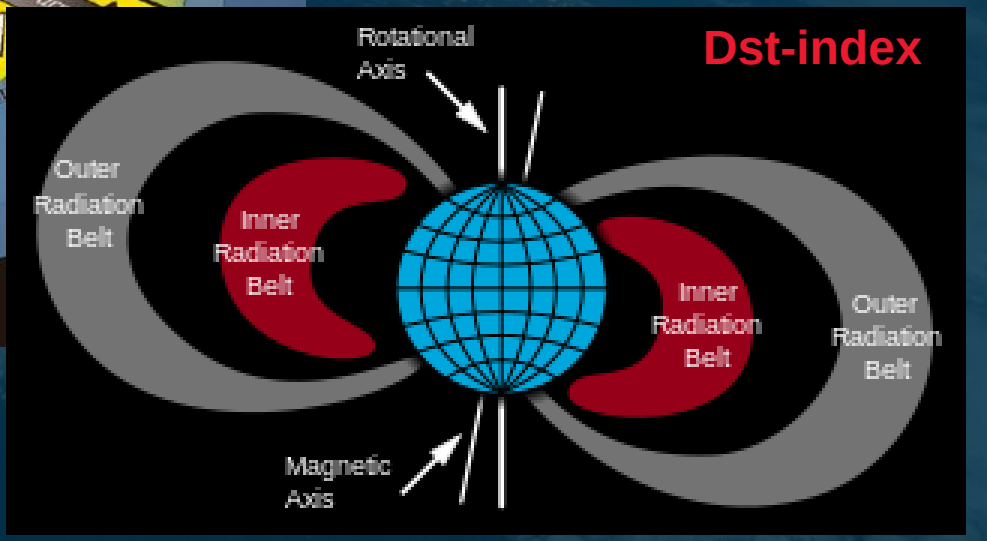
In the Swarm-SMOS comparison, different scales are used.

Geomagnetic activity - Dungey cycle – geomagnetic indices

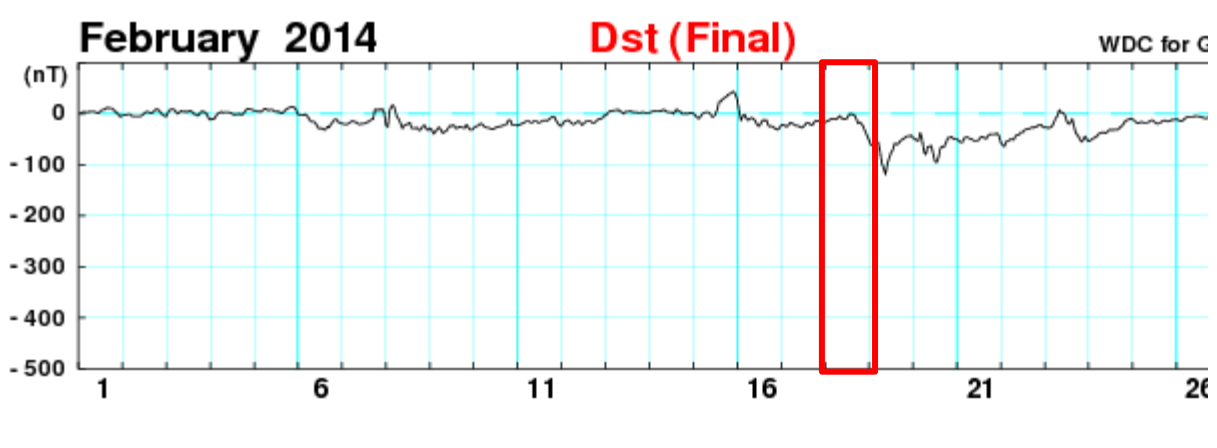


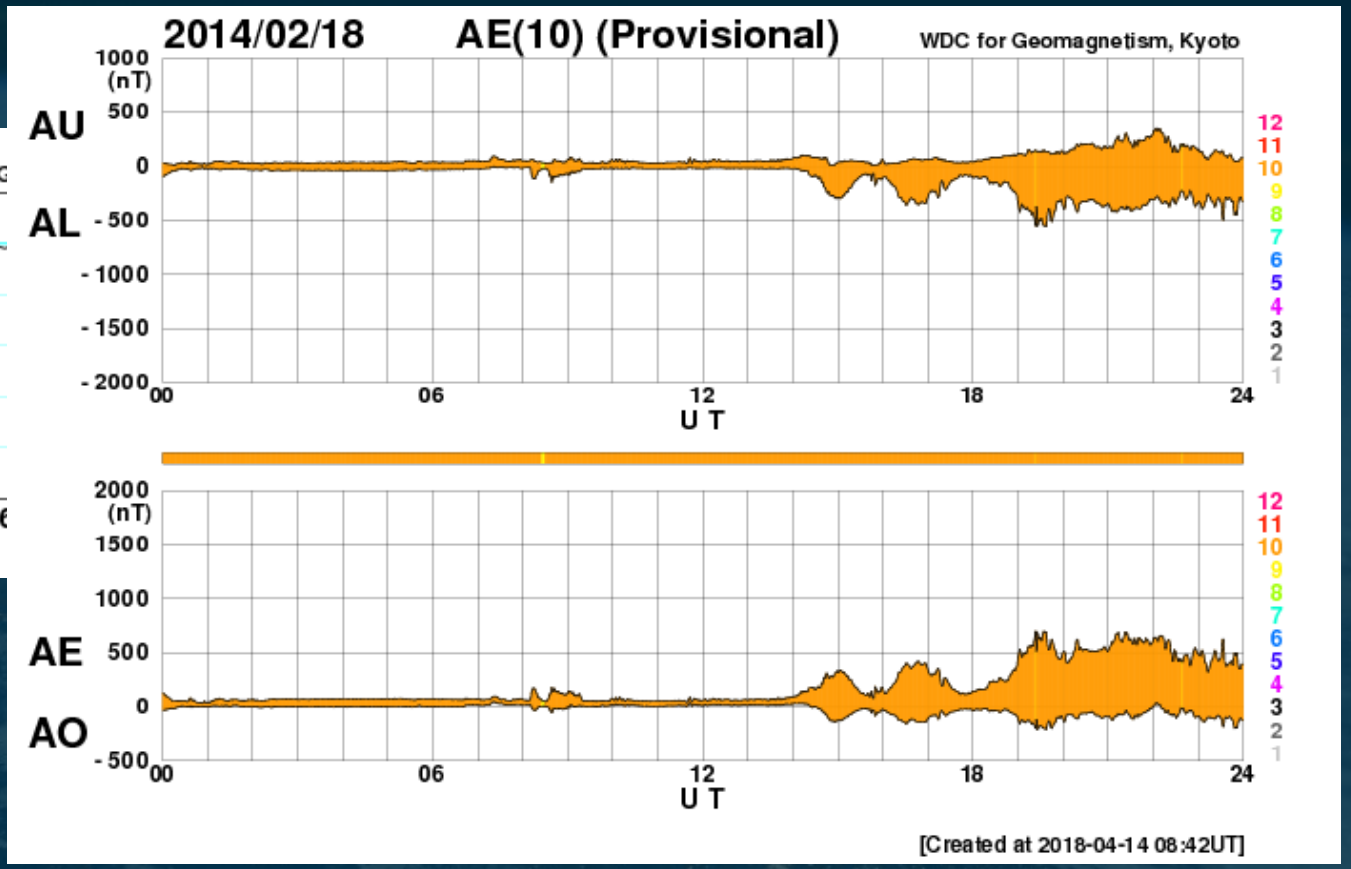
Auroral Electrojet index





Swarm SMOS TEC comparison in quiet conditions





Performing the comparison during quiet geomagnetic conditions, focussing on equatorial regions.

The tool we used…

We developed a new Phyton code, able to download Swarm VTEC data, select Swarm datapoints with a relative distance from SMOS ($\Delta lat^2 + \Delta lon^2$)<threshold, and compare them with SMOS VTEC. (here we used threshold=800)

```
In [1]:
### importa tutti i package necessari ###

import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import datetime as dt
import os
import netCDF4
import xarray as xr
from pathlib import Path
from netCDF4 import Dataset
from scipy.io import netcdf
```

```
In [2]:
### si imposta il nome del file di SMOS da leggere, lo si apre, e si trasferisce il contenuto nella variabile df pandas ###

#file_n = 'VTEC_SMOS_20140218T004931_20140218T013349.nc'
file_n = 'ASC/VTEC_SMOS_20141212T035308_20141212T044309.nc'
ds = xr.open_dataset(file_n,decode_times=False)
df = ds.to_dataframe()
#df
```

```
In [4]:
### Si impostano gli orari dei dati di Swarm da scaricare, sulla base di orario minimo e massimo nel file SMOS,
### aggiungendo 10 minuti da entrambi i lati. Si converte la Julian date in YYYY, MM, DD, HH:MM:SS
# Data di riferimento
reference_date = pd.Timestamp('2000-01-01 00:00:00')

starttime=min(df['time_start']) - (10/1440)
endtime=max(df['time_stop']) + (10/1440)
starttime_date = reference_date + pd.to_timedelta(starttime, unit='D')
starttime_date = starttime_date.round('min')
endtime_date = reference_date + pd.to_timedelta(endtime, unit='D')
endtime_date = endtime_date.round('min')
starttime_date, endtime_date

Out[4]: (Timestamp('2014-12-12 03:43:00'), Timestamp('2014-12-12 04:53:00'))
```

```
In [5]:
### Da ViresClient si scaricano i fati di Swarm corrispondenti all'orario impostato nel precedente ciclo ###
from viresclient import SwarmRequest
r = SwarmRequest("https://vires.services/ows", token="zF_aBX9hmPcYSlaHRErtioM1xpGI6xac")
```

```
In [6]:
### Da ViresClient si scaricano i fati di Swarm corrispondenti all'orario impostato nel precedente ciclo ###
#Swarm_A
r.set_collection('SW_OPER_TECBTMS_2F')
r.available_auxiliaries()
r.set_products(measurements=[p for p in r.available_measurements('SW_OPER_TECATMS_2F')], auxiliaries=['Timestamp','Latitude',
data = r.get_between(start_time=starttime_date, end_time=endtime_date)
df_S_A = data.as_dataframe()
df_SwA_AVG=df_S_A.resample('10s').mean()
```

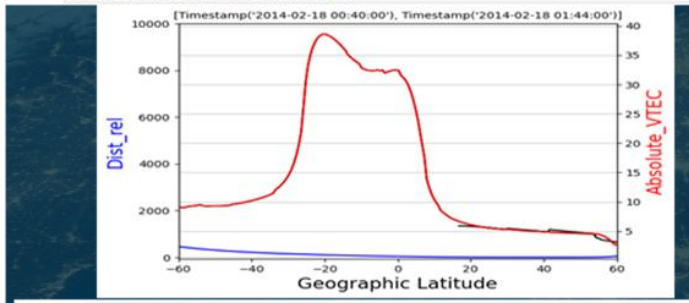
```
In [7]:
### ELIMINA TUTTE LE RIGHE DI SMOS IN CUI VTEC=NaN, E CREA LE DUE VARIABILI dfRes e dfRes_SwA con indici resettati
### per SMOS e SwarmA rispettivamente
df.dropna(subset = ["VTEC"], inplace=True)
dfRes=df.reset_index()
dfRes_SwA=df_SwA_AVG.reset_index()
```

```
In [8]:
### Crea le variabili Dist_rel (con lunghezza pari a dfRes_SwA) e Distanza, che saranno utilizzate nel prossimo blocco
Dist_rel = pd.DataFrame()
Dist_rel.index = np.arange(len(dfRes_SwA))
Dist_rel['COL']=(180**2 + 360**2)
Distanza = pd.DataFrame()
#Dist_rel
```

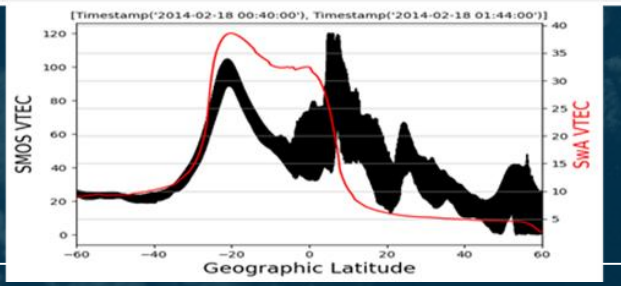
```
In [9]:
### Calcola la distanza minima relativa tra ciascun punto di Swarm ed il più vicino punto di SMOS corrispondente,
### calcolando la distanza relativa come Delta_lat^2 + Delta_long^2, e la inserisce nel vettore Dist_rel
### N.B.: è l'ultimo ciclo IF che seleziona il punto di SMOS più vicino a ciascun punto di Swarm.
for i in range (len(dfRes_SwA)-1):
    Distanza = (180**2 + 360**2)
    if (i/10-int(i/10) == 0): print('Progress report...', i/10)
    for ii in range (len(dfRes)-1):
        lat_diff = dfRes.at[ii, 'latitude'] - dfRes_SwA.at[i, 'Latitude']
        lon_diff = dfRes.at[ii, 'longitude'] - dfRes_SwA.at[i, 'Longitude']
        Distanza = lat_diff*lat_diff + lon_diff*lon_diff
        if(Distanza<Dist_rel.at[i,'COL']):
            ##Dist_rel = Dist_rel.append({'COL': i}, ignore_index=True)
            Dist_rel.at[i,'COL']=Distanza
```

```
In [11]:
### Crea la nuova variabile dfRes_SwA_Filt ed assegna NaN a tutti i valori di Swarm in cui la distanza minima relativa
### dalle posizioni di SMOS è superiore a 800

dfRes_SwA_Filt=dfRes_SwA.copy()
for i in range (len(dfRes_SwA_Filt)-1):
    #if (i/10-int(i/10) == 0): print('Progress report...', i/10)
    if(Dist_rel.at[i,'COL']>800):
        dfRes_SwA_Filt.at[i,'Absolute_VTEC']=np.nan
        #print('rimossa riga',i,dfRes_SwA_Filt.at[i,'Absolute_VTEC'])
```

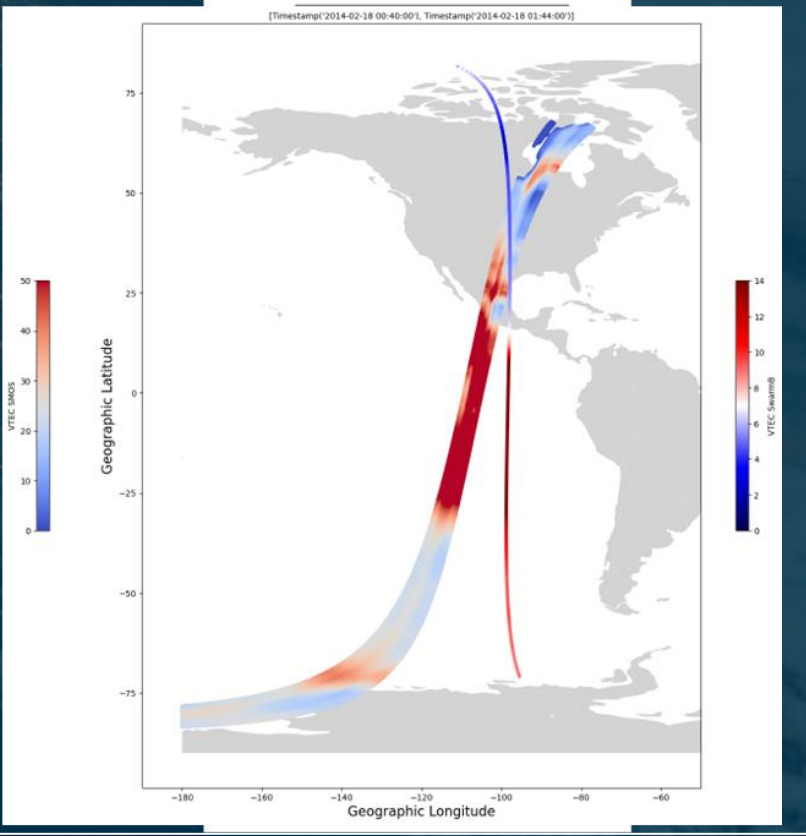
```
In [12]:
### Grafica il VTEC delle variabili dfRes_SwA , dfRes_SwA_Filt, e la minima distanza relativa Dist_rel in funzione della
### latitudine

plt.figure(1)
plt.clf()
plt.figure(figsize=(15,6))
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()

ax1.plot(dfRes_SwA['Latitude'],Dist_rel['COL'],'b-')
ax2.plot(dfRes_SwA['Latitude'],dfRes_SwA['Absolute_VTEC'], 'k-')
ax2.plot(dfRes_SwA_Filt['Latitude'],dfRes_SwA_Filt['Absolute_VTEC'], 'r-')
#ax2.plot(dfRes['Latitude'],dfRes['VTEC'], 'k-')

ax1.set_xlabel('Geographic Latitude', fontsize=16)
ax1.set_ylabel('Dist_rel', fontsize=16, color='b')
ax2.set_ylabel('Absolute_VTEC', fontsize=16, color='r')
ax1.set_ylim([-50, 10000])
ax1.set_xlim([-60, 60])
#plt.xlim(-100,5000)
plt.grid(axis='y', alpha=0.75)
plt.grid(axis='x', alpha=0.75)
plt.title([starttime_date, endtime_date],fontsize=10)
```



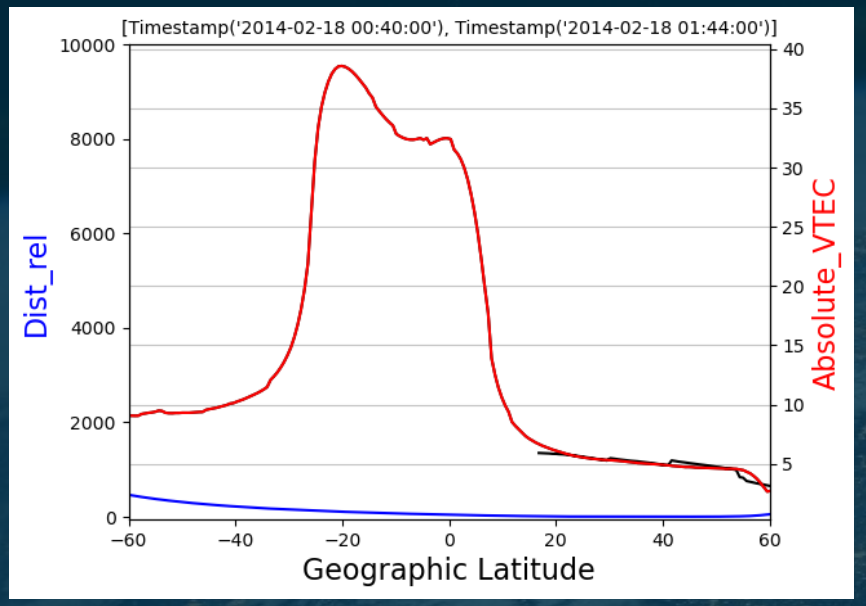

```
In [13]:
### Grafica il VTEC di Swarm e SMOS in funzione della latitudine, tra -60° e +60°

plt.figure(1)
plt.clf()
plt.figure(figsize=(15,6))
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()

ax1.plot(dfRes['latitude'],dfRes['VTEC'], 'k-')
ax2.plot(dfRes_SwA_Filt['Latitude'],dfRes_SwA_Filt['Absolute_VTEC'], 'r-')

ax1.set_xlabel('Geographic Latitude', fontsize=16)
ax1.set_ylabel('SMOS VTEC', fontsize=16, color='k')
ax2.set_ylabel('SwA VTEC', fontsize=16, color='r')
#ax1.set_ylim([-50, 10000])
ax1.set_xlim([-70, 70])
#plt.xlim(-100,5000)
plt.grid(axis='y', alpha=0.75)
plt.grid(axis='x', alpha=0.75)
plt.title([starttime_date, endtime_date],fontsize=10)
```

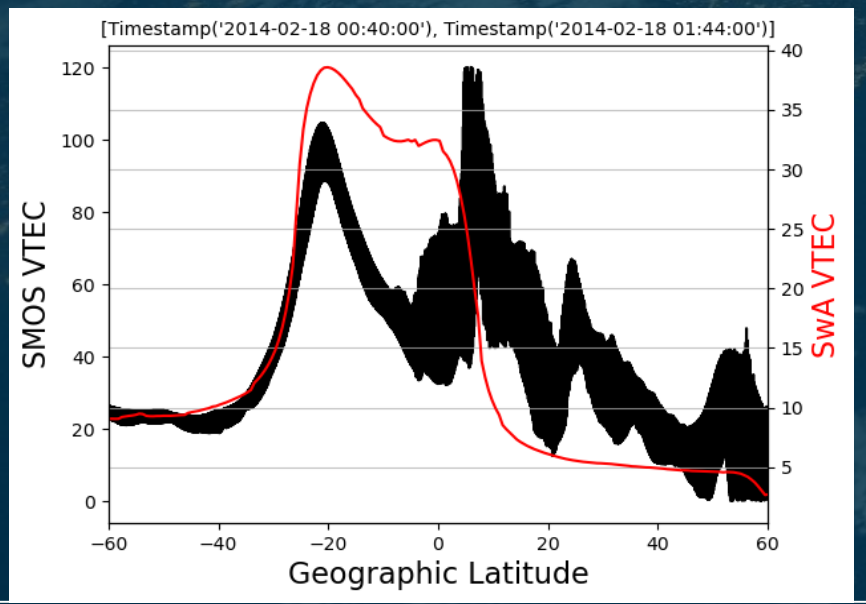


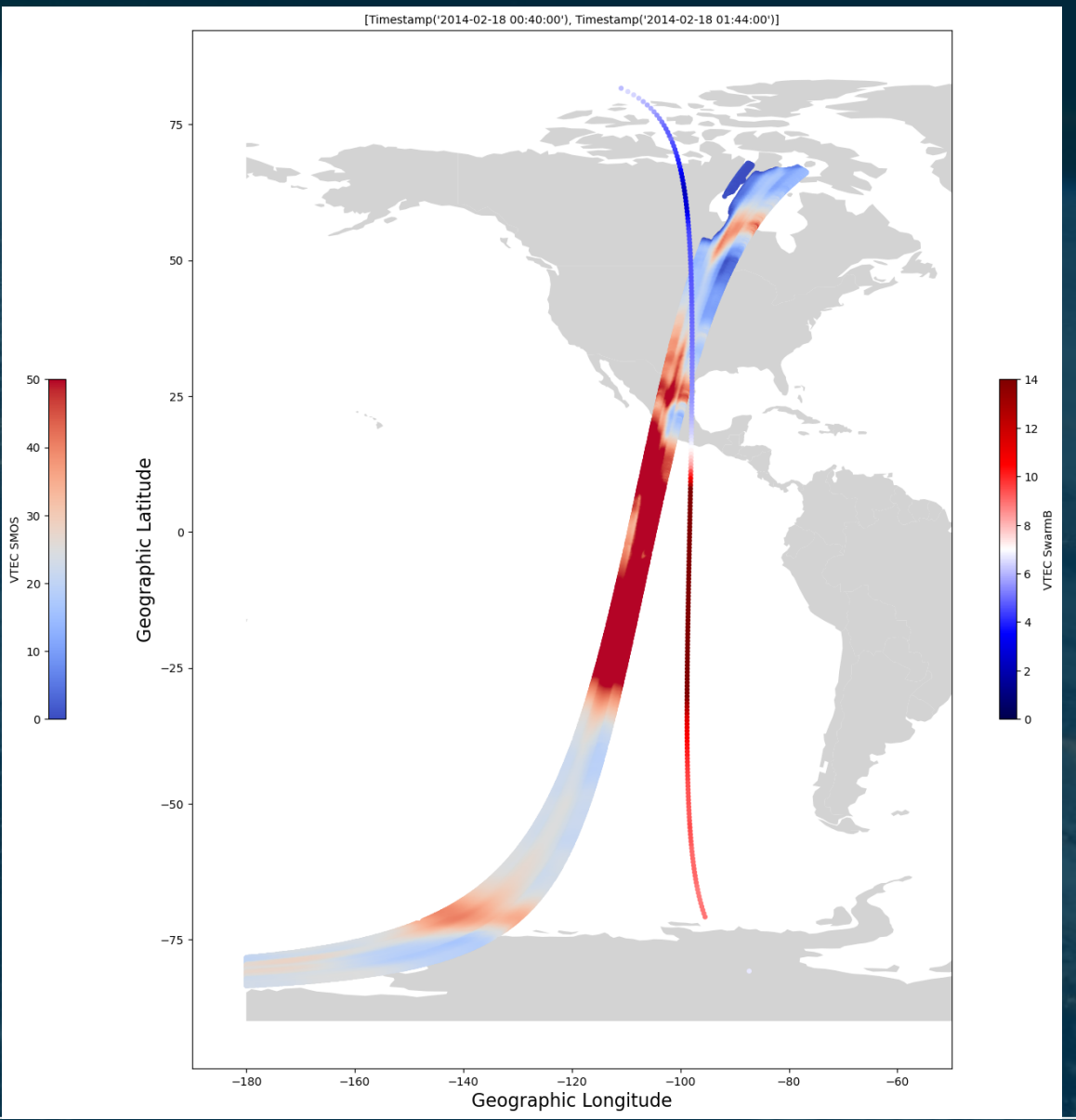

```
In [16]:
### Grafica il VTEC di Swarm e SMOS in funzione della Latitudine e Longitudine
plt.figure(1)
plt.clf()
plt.figure(figsize=(9,12))
plt.scatter(dfRes['longitude'],dfRes['latitude'],c=(dfRes['VTEC']), vmin=0, vmax=50, s=12, cmap='coolwarm')
char =plt.colorbar(orientation='vertical', shrink=0.3, location = 'left')
char.set_label('VTEC SMOS')

#plt.scatter(dfRes_SwA_Filt['Longitude'],dfRes_SwA_Filt['Latitude'],c=(dfRes_SwA_Filt['Absolute_VTEC']), vmin=0, vmax=10, s=1
plt.scatter(dfRes_SwA['Longitude'],dfRes_SwA['Latitude'],c=(dfRes_SwA['Absolute_VTEC']), vmin=0, vmax=10, s=12, cmap='seismic
char_SwA =plt.colorbar(orientation='vertical', shrink=0.3, location = 'right')
char_SwA.set_label('VTEC SwarmB')

plt.xlim(-190,190)

plt.ylabel('Geographic Latitude', fontsize=16)
plt.xlabel('Geographic Longitude', fontsize=16)
plt.title([starttime_date, endtime_date],fontsize=10)

#df_SwA_AVG['Latitude'],df_SwA_AVG['Absolute_VTEC']
```



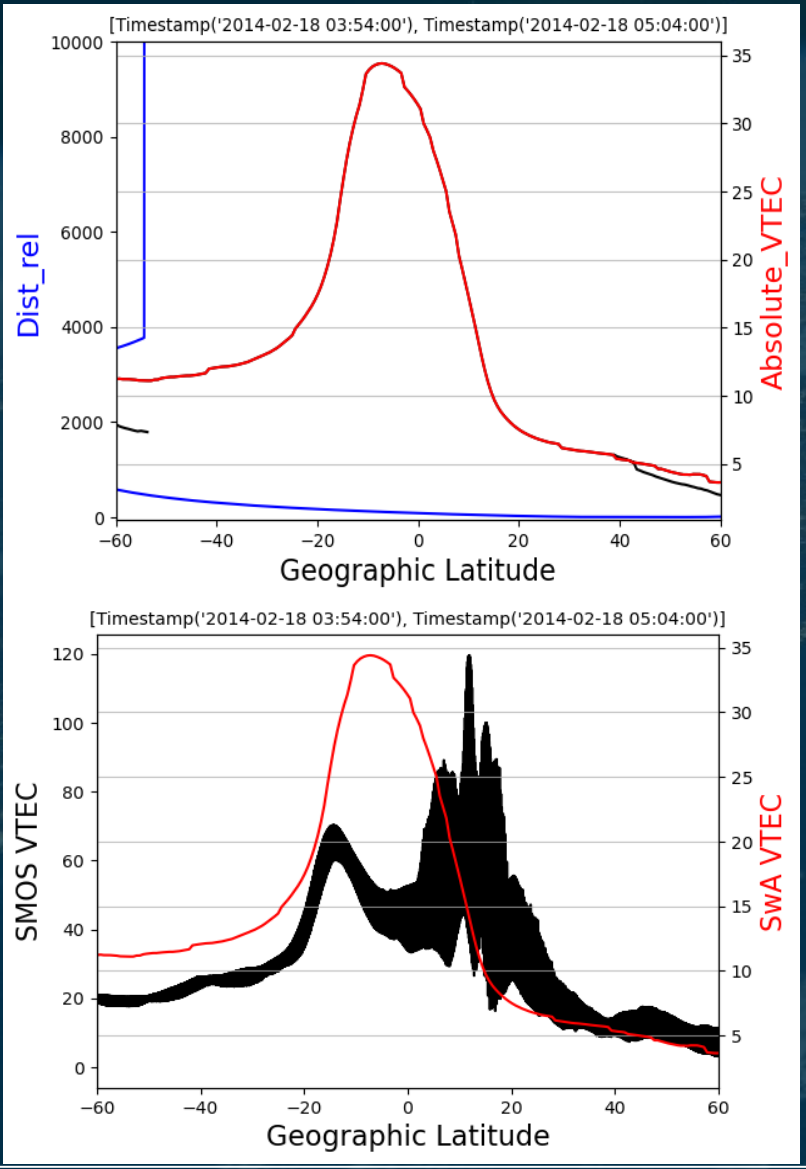
2014-02-18 00:40:00 – 01:44:00

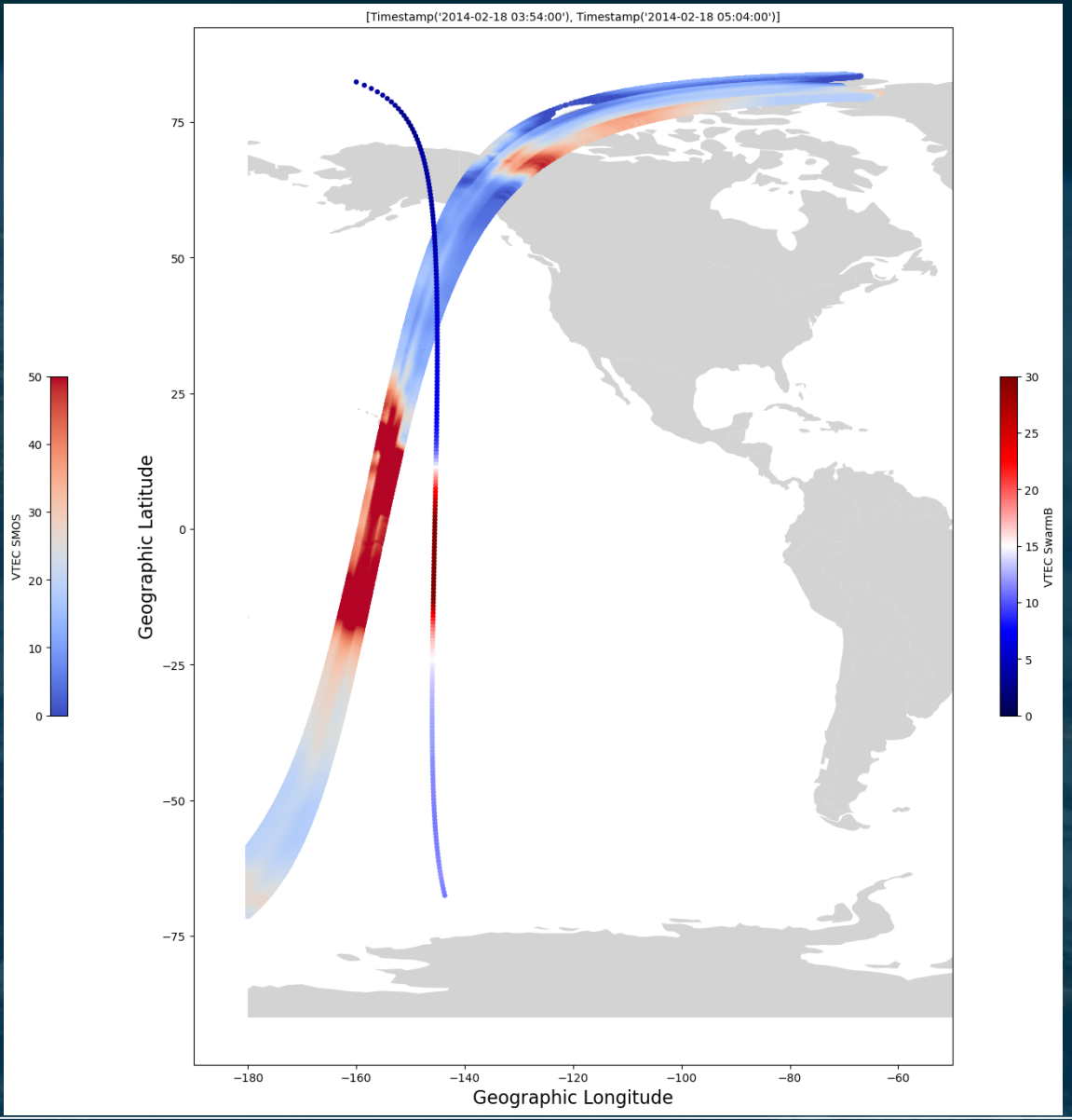




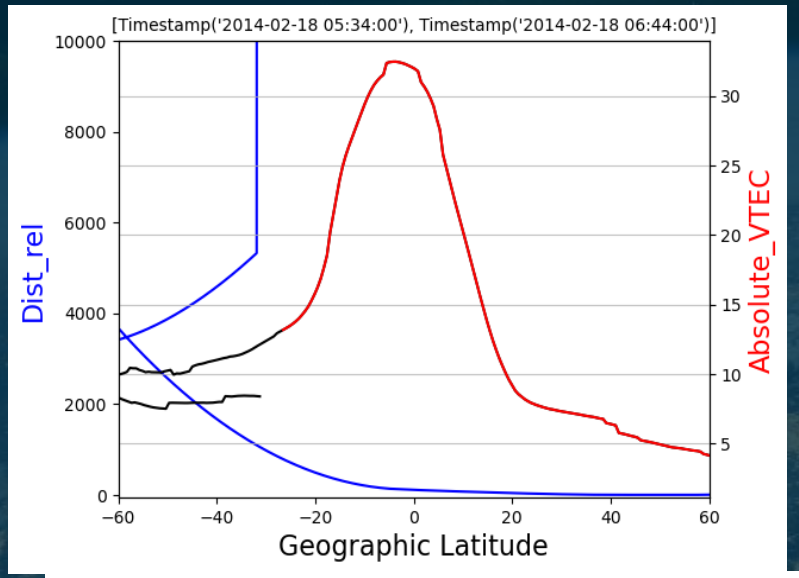


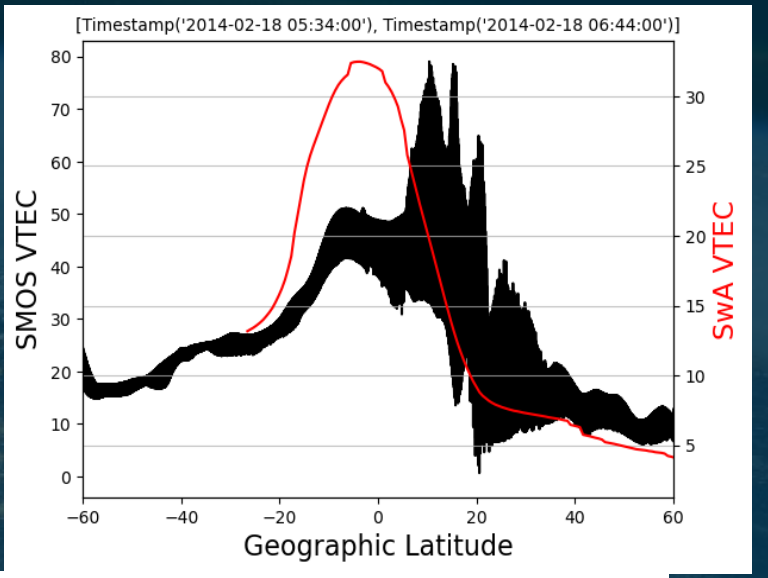
2014-02-18 03:54:00 – 05:04:00

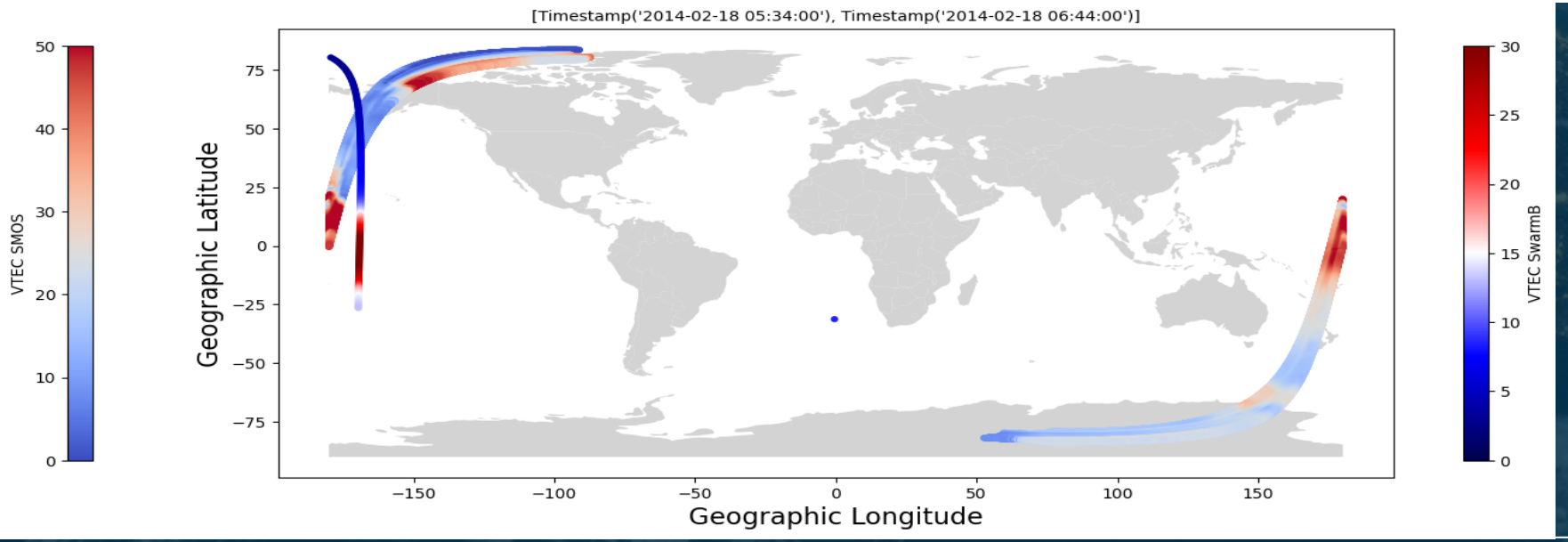




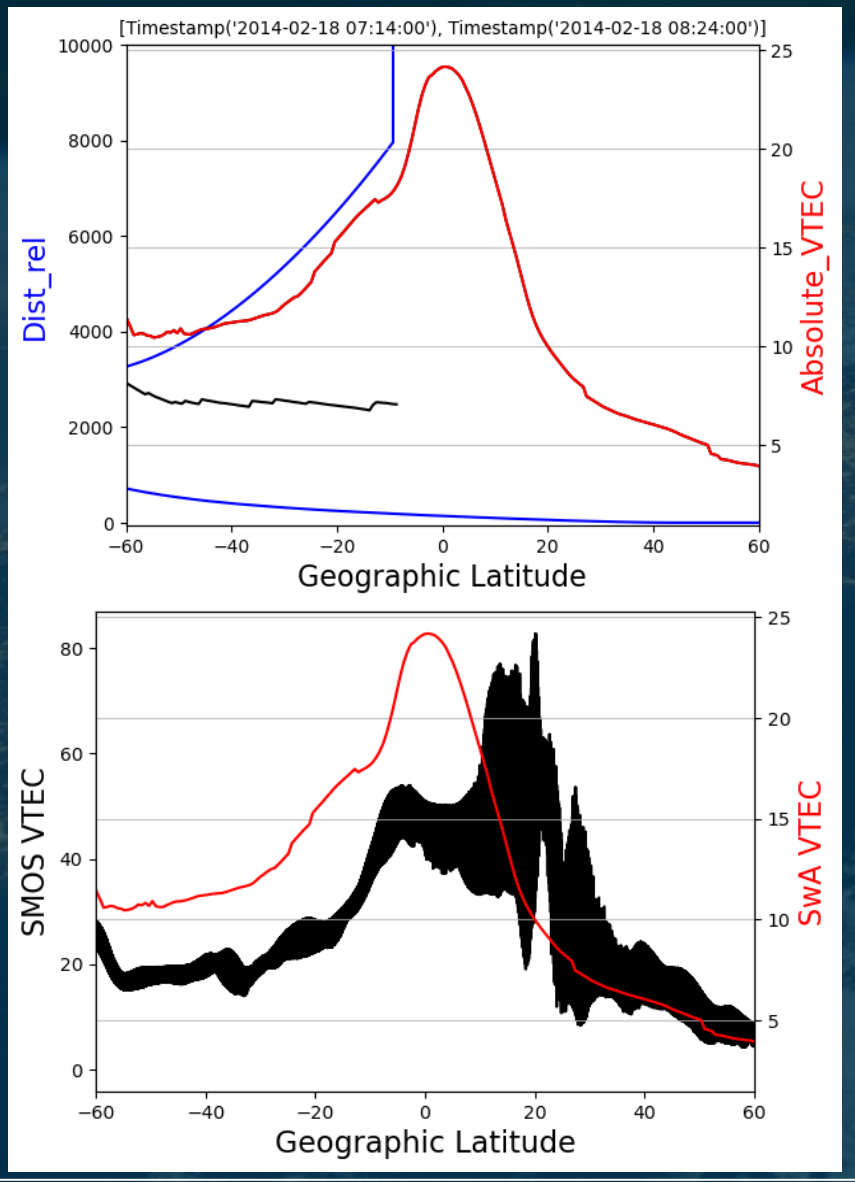
2014-02-18 05:34:00 – 06:44:00

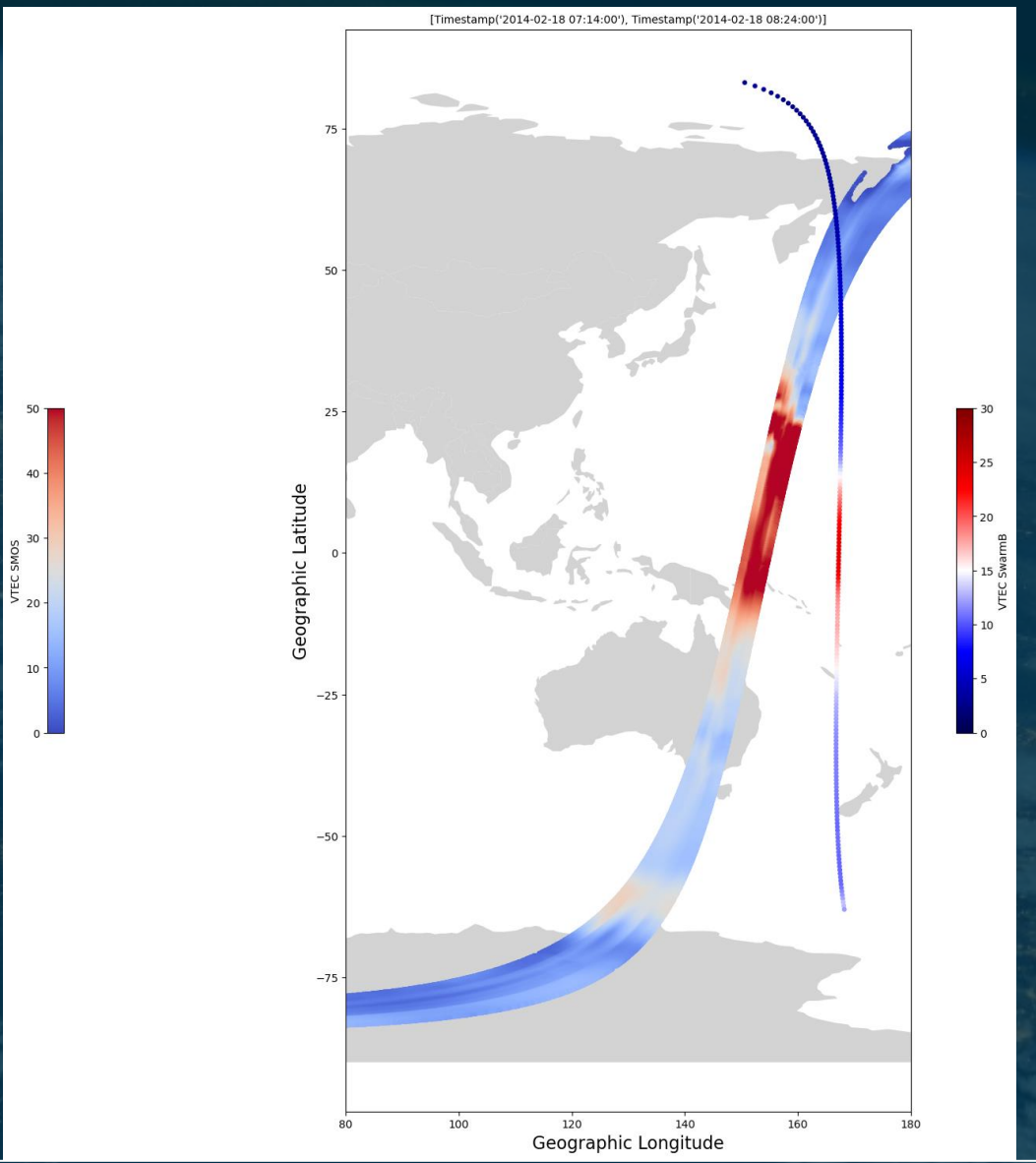




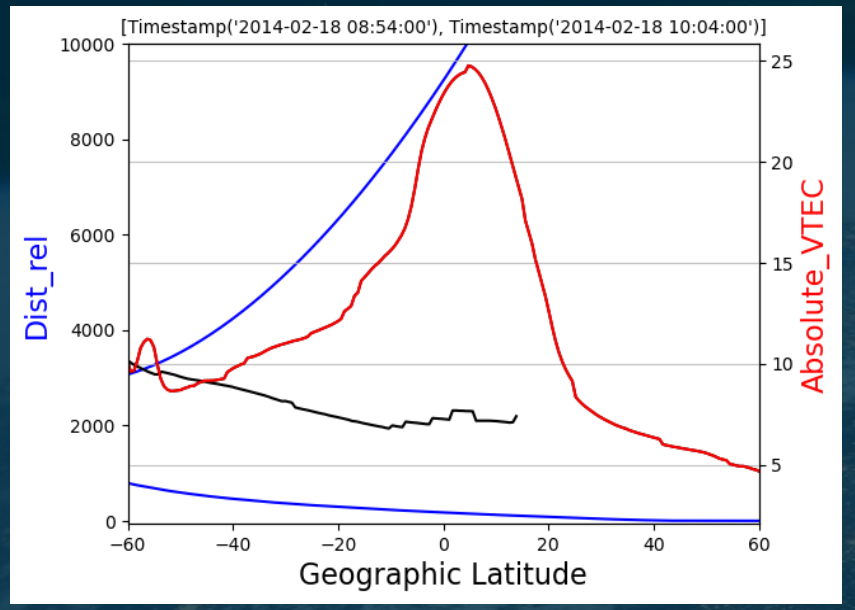


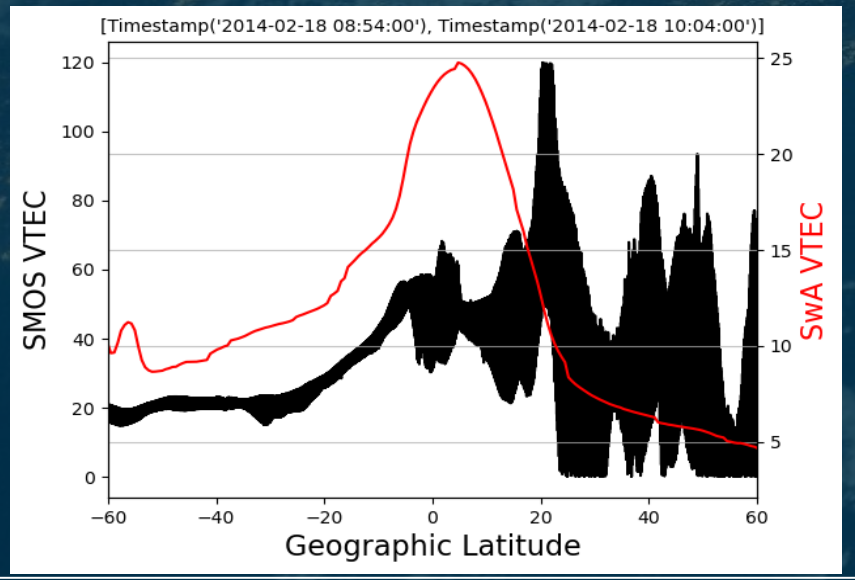
2014-02-18 07:14:00 – 08:24:00

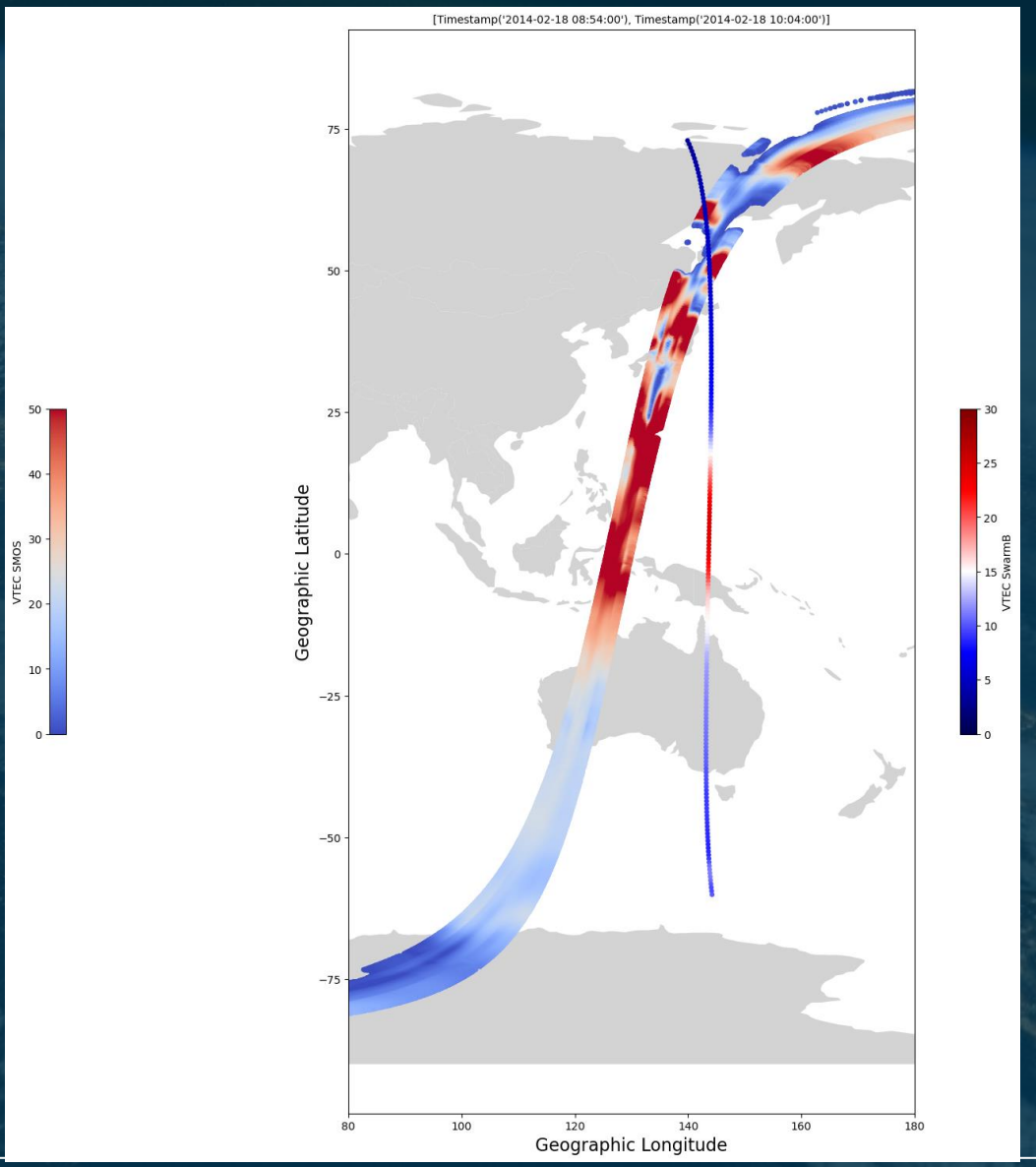




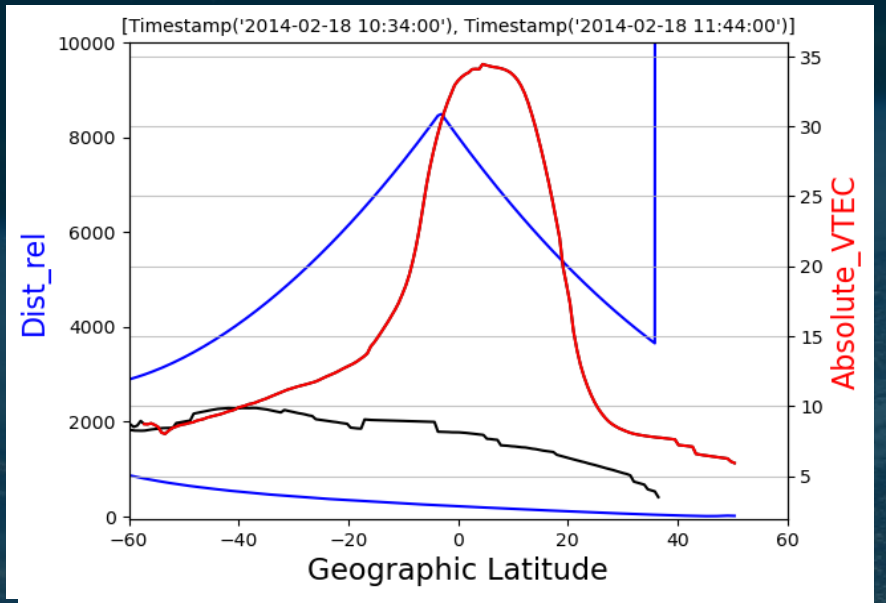
2014-02-18 08:54:00 – 10:04:00

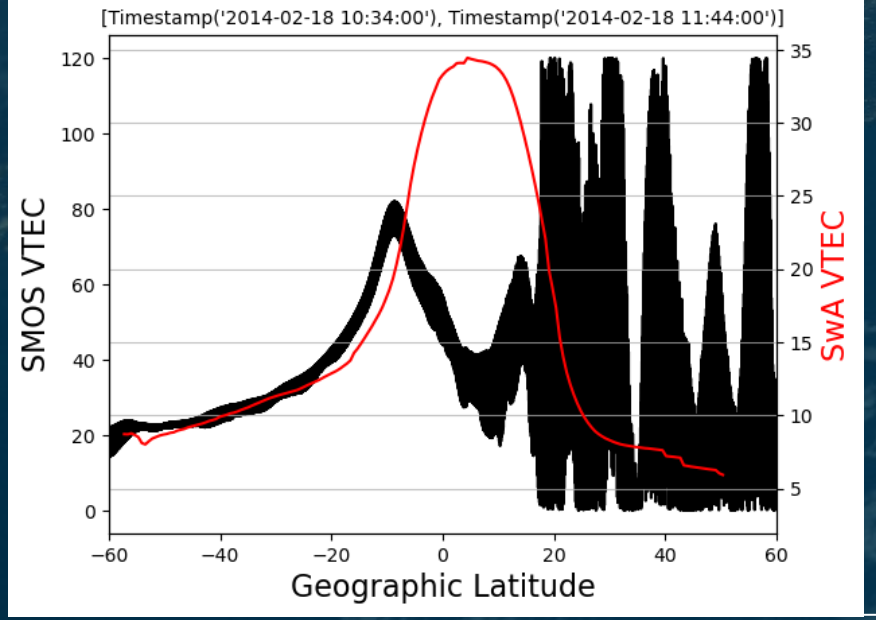


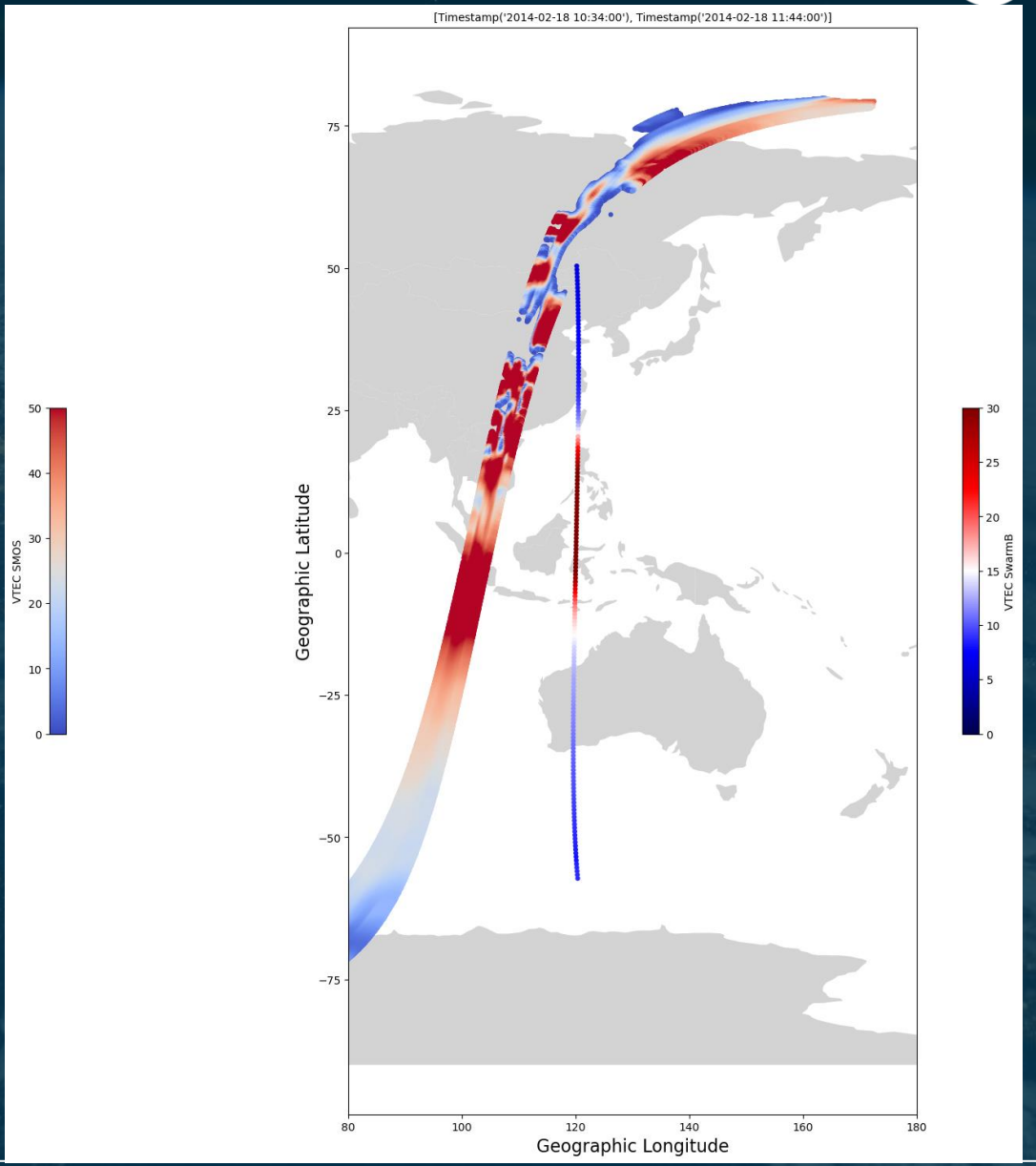




2014-02-18 10:34:00 – 11:44:00







The equatorial fountain effect

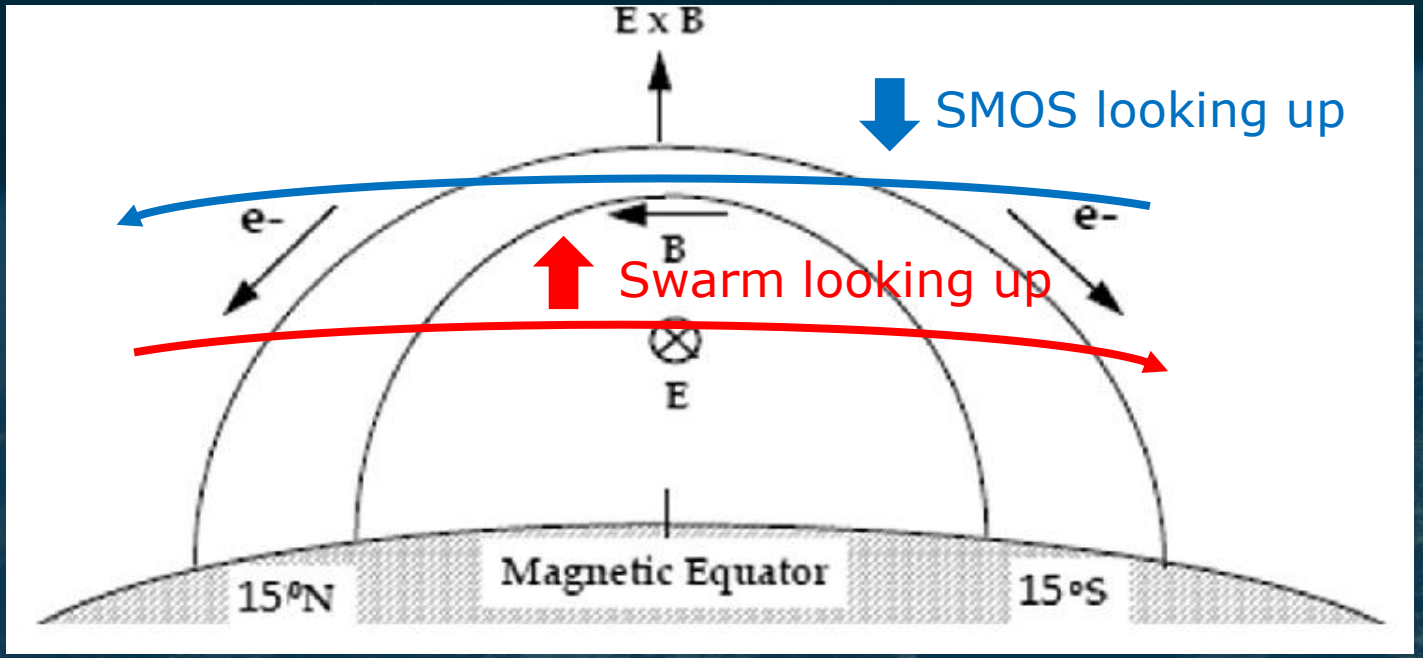
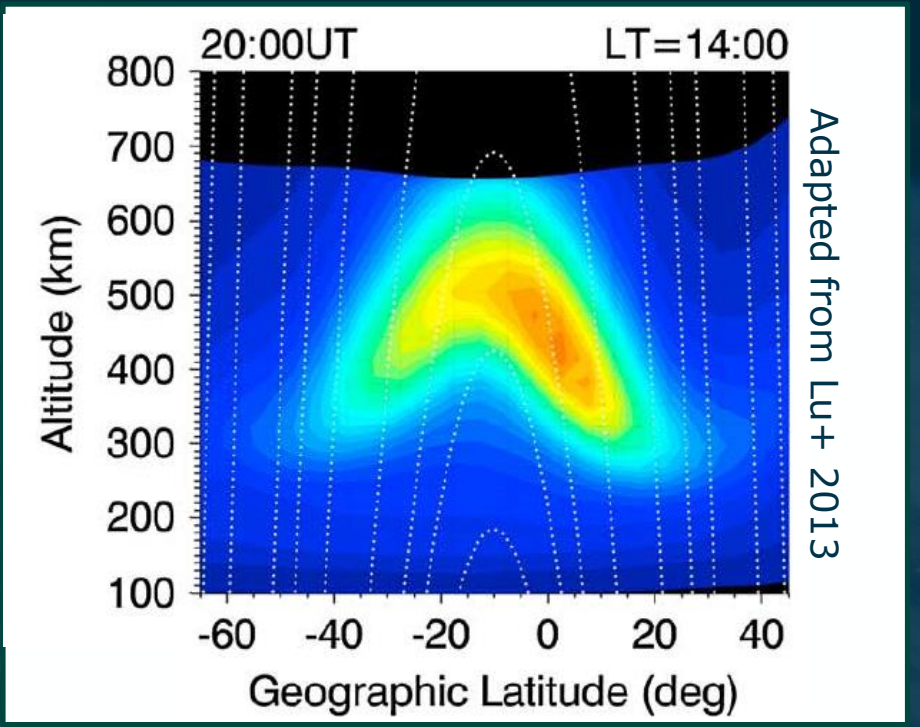


Fig. adapted from Bosco Oryema et al. (2015)



The fountain effect is a lift from ionospheric lower layers of dense plasma around the geomagnetic equator, under the effect of E x B drift.

E horizontal, from east to west along the equator, due to ionospheric wind dynamo (tidal effects + neutral winds); B horizontal is south to north (Earht's dipole field). The result is an upward drift.

When the upward drift loses momentum, plasma diffuses down under gravity along the geomagnetic field lines (SMOS double peaks)

SUMMARY

- Good agreement among the trend of SMOS – Swarm VTEC, when SMOS data are not contaminated by sea / land interfaces. The larger VTEC values measured by SMOS are expected, since it samples the lower and denser ionosphere.
- Both spacecraft clearly measure the fountain effect intensification of density around the magnetic equator, with a single peak for Swarm and a double peak for SMOS.
- Some latitudinal shift is also present among the VTEC from the two s/c => could be related to the inclination of geomagnetic equator when displayed in geographical coordinates

NEXT STEPS

- Add data from other Swarm s/c (during these 2014 events all the 3 s/c along the same orbit) + add predictions from IRI model to conclude this quiet geomagnetic conditions comparison
- Set up a statistical comparison, to monitor the average behavior of the two signals: Super-epoch analysis, taking as a reference time the crossing of magnetic equator, separating the events according to geomagnetic activity level ; Local Time; etc seems a feasible approach