

DIPContMod

July 6, 2022

Daedalus Ionospheric Profile Continuation (DIPCont) Project

The *DIPCont* project is concerned with the accuracy and the robustness of parameter estimation and (downward) continuation in the lower thermosphere-ionosphere (LTI) region based on Daedalus in situ data. Using synthetic measurements of neutral temperature T_n , neutral density N_n , electron density N_e , and ion temperature T_i along anticipated Daedalus satellite orbit sections around perigee, the *DIPCont* approach proceeds to

- estimate *ensembles of model parameters* through Monte Carlo runs,
- construct *ensembles of altitude profiles* $T_n = T_n(z)$, $N_n = N_n(z)$, $N_e = N_e(z)$, $T_i = T_i(z)$ from the model parameter ensembles,
- compute *ensembles of derived variables* such as ion-neutral collision frequency $\nu_{in} = \nu_{in}(z)$ and *Pedersen conductivity* $\sigma_P = \sigma_P(z)$,
- determine *relative deviations of variables* as functions of altitude z ,
- obtain *extrapolation horizons* for suitable error thresholds.

When using DIPCont code, please acknowledge the DIPCont publication (Joachim Vogt et al., under review).

DIPContMod.ipynb

This Jupyter Notebook is offered as supplementary documentation of the module `DIPContMod.py`.

Written by Joachim Vogt, Jacobs University Bremen (JUB). Tested also by Octav Marghitu, ISS Bucharest (ISS), Adrian Blagau (ISS, JUB), Leonie Pick (DLR Neustrelitz, JUB), and Nele Stachlys (U Potsdam, JUB); in collaboration with Stephan Buchert, Swedish Institute of Space Physics in Uppsala, Theodoros Sarris, Democritus University of Thrace in Xanthi (DUTH), Stelios Tourgaidis (DUTH), Thanasis Balafoutis (DUTH), Dimitrios Baloukidis (DUTH), and Panagiotis Pirnaris (DUTH).

This version: v0.2, 2022-07-06.

1 DIPContMod.py : DIPCont parametric models of LTI region variables

The Python module `DIPContMod.py` builds on `DIPContBas.py` that defines the basic setup of the DIPCont project, initializes parameters and variables that are exchanged between different modules and functions, and provides supplementary information. After the following import, a module description is available through `help(DCB)`.

```
[1]: import DIPContBas as DCB
```

In a similar fashion, the module `DIPContMod.py` is imported. Since no exchange variables (global model parameters) are affected, the abbreviation is in lower case letters.

```
[2]: import DIPContMod as dcm
```

1.1 Horizontal profiles of boundary values and peak parameters

The functions provided in the first section of `DIPContMod.py` facilitate the construction of horizontal profiles of model parameters. In the functions `VarBotModx()` and `VarTopModx()`, neutral and ion temperatures at the upper and lower boundaries as well as neutral density at the lower boundary are linearly interpolated using configurational parameters from `DIPContBas.py`. The function `ScaleHeightParReg()` computes density scale height parameters from temperature values at the upper and lower boundaries. For the specific model setup of a simplified auroral zone crossing with two ionization peaks (inbound and outbound orbital sections of the auroral oval), electron density peak parameters are defined in the function `IpkParAZC()`.

```
[3]: x = 0
TnTop = dcm.VarTopModx('Tn',x)
TnBot = dcm.VarBotModx('Tn',x)
IGHNn,HNnTop,HNnBot = dcm.ScaleHeightParReg('n',TnTop,TnBot)
print('Inverse gradient of the density scale height : IGHNn = {:.2f}'.
      ↪format(IGHNn))
print('Density scale height at the upper boundary : HNnTop = {:.2f} km'.
      ↪format(HNnTop/1e3))
print('Density scale height at the lower boundary : HNnBot = {:.2f} km'.
      ↪format(HNnBot/1e3))
```

```
Inverse gradient of the density scale height : IGHNn = 5.13
Density scale height at the upper boundary : HNnTop = 24.35 km
Density scale height at the lower boundary : HNnBot = 4.87 km
```

1.2 Prediction models in terms of coordinates (z, x)

The second section of `DIPContMod.py` contains two functions. Two-dimensional distributions of model variables are produced by `VarModzx(Var,z,x)`, intended to be used, e.g., in contour diagrams of LTI region variables. The input coordinate arrays `z` and `x` do not have to be two-dimensional arrays, they are only required to follow general broadcasting rules. The function `SatPrd()` computes model predictions along satellite orbits as provided with `DIPContBas` (DCB), and collects the results in a Pandas dataframe.

```
[4]: import numpy as np
z = np.linspace(DCB.zBot,DCB.zTop,6)
print('Sample altitude profile [km]: ',z/1e3)
x = 0
Ne = dcm.VarModzx('Ne',z,x)
print('Electron density profile [1/cm^3]:')
with np.printoptions(precision=1, suppress=True): print(Ne/1e6)
dfPrd = dcm.SatPrd()
```

```
display(dfPrd)
```

Sample altitude profile [km]: [100. 120. 140. 160. 180. 200.]

Electron density profile [1/cm³]:

[36391.1 78356.8 36196.5 18774.6 11015.4 7064.8]

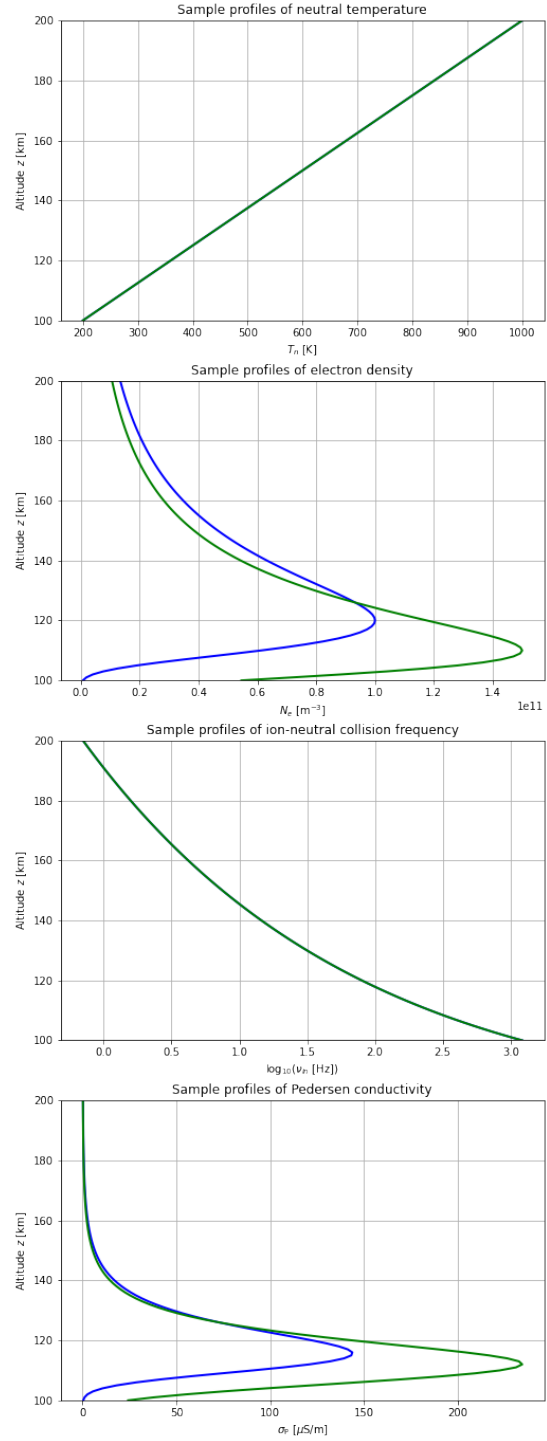
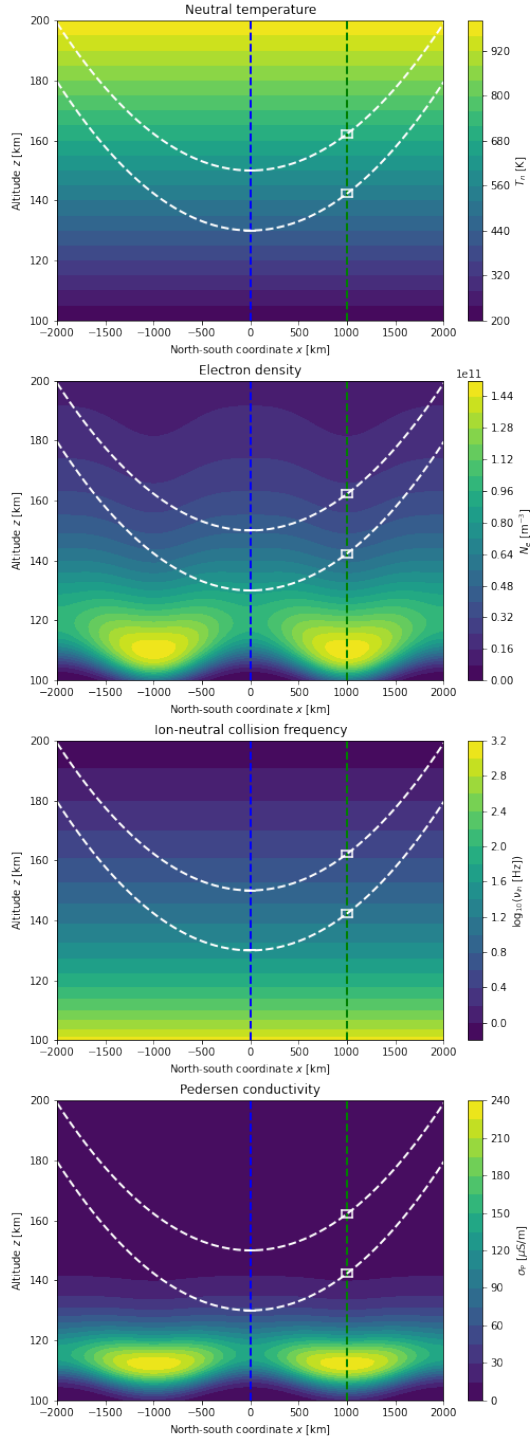
	z	x	LogTn	LogNn	LogNe	LogTi
0	206743.657004	-2.481929e+06	6.960300	35.216304	22.544129	6.960300
1	206711.683811	-2.481420e+06	6.960057	35.217550	22.544749	6.960057
2	206679.717280	-2.480911e+06	6.959814	35.218796	22.545370	6.959814
3	206647.757411	-2.480403e+06	6.959571	35.220042	22.545990	6.959571
4	206615.804203	-2.479894e+06	6.959329	35.221289	22.546611	6.959329
...
19197	225085.285251	2.466307e+06	7.090645	34.547094	22.210603	7.090645
19198	225116.600144	2.466813e+06	7.090854	34.546023	22.210068	7.090854
19199	225147.921565	2.467319e+06	7.091062	34.544952	22.209534	7.091062
19200	225179.249516	2.467826e+06	7.091271	34.543880	22.209000	7.091271
19201	225210.583995	2.468332e+06	7.091480	34.542809	22.208466	7.091480

[19202 rows x 6 columns]

1.3 Plot functions for individual LTI region model variables

The third section of `DIPContMod.py` provides two functions for displaying model variables in the LTI region: `PltMod2d()` produces contour diagrams, and `PltMod1d()` plots a set of vertical profiles at horizontal positions defined by the array `xGrd`.

```
[5]: import matplotlib.pyplot as plt
DCB.LTIModelType = 'NeAuroralZoneCrossing'
xGrd = np.array([0.5*(DCB.xLef+DCB.xRig),0.5*DCB.xRig])
DCB.xWin = 100e3
xWin = DCB.xWin
PltVarStr = ['Tn','Ne','FCin','Cped']
plt.figure(figsize=(18,24))
for iVar in range(len(PltVarStr)):
    Var = PltVarStr[iVar]
    plt.subplot(len(PltVarStr),2,1+2*iVar)
    dcm.PltMod2d(Var,xGrd=xGrd,xWin=xWin)
    plt.subplot(len(PltVarStr),2,2+2*iVar)
    dcm.PltMod1d(Var,xGrd)
```



End of DIPContMod.ipynb