

DIPContBas

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).

DIPContBas.ipynb

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

PYTHON CODE UNDER DEVELOPMENT, PROVIDED AS IS, NO WARRANTY.

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 DIPContBas.py : DIPCont basic setup and exchange variables

The Python module `DIPContBas.py` 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
```

1.1 General physical constants and planetary parameters

In the first section of `DIPContBas.py`, general physical constants and relevant planetary parameters of the Earth are defined. If not explicitly stated otherwise, physical variables are expressed in SI units.

```
[2]: print(r'Mass of planet Earth [kg]: {:.4e}'.format(DCB.MEarth))
```

Mass of planet Earth [kg]: 5.9722e+24

1.2 LTI region boundaries and regional parameters

The second section of `DIPContBas.py` sets values at LTI region boundaries for the four Daedalus observables T_n (Tn), N_n (Nn), N_e (Ne), T_i (Ti). Acronyms Bot and Top refer to the vertical direction (altitude), Lef and Rig to the horizontal direction (geomagnetically north-south) direction. Electron density parameters are specified at the ionization peak (Ipk) for the horizontally uniform case and a simplified auroral zone crossing scenario. Global reference values for assumingly constant variables (molar masses of neutrals and ions, magnetic field magnitude) are indicated with the acronym LTI.

```
[3]: print('Neutral temperature at the upper left boundary [K]: {:.1e}'.format(DCB.  
      ↪TnTopLef))  
print('Molar mass of neutrals [kg/mol]: {}'.format(DCB.MnLTI))  
print('Default electron density peak value [1/m^3]: {:.1e}'.format(DCB.NeIpk))
```

Neutral temperature at the upper left boundary [K]: 1.0e+03

Molar mass of neutrals [kg/mol]: 0.028

Default electron density peak value [1/m³]: 1.0e+11

1.3 Satellite parameters and instrumental errors

In the third section of `DIPContBas.py`, the orbit parameters of two satellites A and B are specified: common start time (`tBeg`) and stop time (`tEnd`), temporal resolution `ObsPerSec`, perigee altitudes (`zPerA`, `zPerB`), and apogee altitudes (`zApoA`, `zApoB`). Defined are also the relative errors of the four Daedalus observables considered here.

```
[4]: print('Perigee altitude of the lower satellite [m]: {:.1e}'.format(DCB.zPerA))  
print('Relative error of electron density measurements: {}'.format(DCB.  
      ↪RelErrNe))
```

Perigee altitude of the lower satellite [m]: 1.3e+05

Relative error of electron density measurements: 0.1

1.4 Parameter estimation and profile continuation

The fourth section of `DIPContBas.py` is concerned with variables for parameter estimation and profile continuation (extrapolation). Most model parameters refer to local representations with amplitudes and length scales at a reference altitude z_0 (`z0`), and are included here for the purpose of exchange between different DIPCont modules and functions. Concrete values listed here do not matter as the parameters are updated in the course of the modeling procedure. Parameters that

enter a model nonlinearly require an iterative approach and thus initial estimates. To stabilize the estimation of temperature and density scale height parameters, neutral and ion temperature values at the lower boundary are assumed to be available from suitable models. Further parameters defined in this section include the grid of horizontal distances used for constructing two-dimensional distributions of extrapolation horizons.

```
[5]: print('Initial estimate of local neutral temperature [K]: {:.1e}'.format(DCB.
      ↪Tn0Init))
      print('Width of the horizontal selection window [m]: {:.1e}'.format(DCB.xWin))
```

```
Initial estimate of local neutral temperature [K]: 5.0e+02
Width of the horizontal selection window [m]: 1.0e+05
```

1.5 Derived variables and dataframes

The fifth section of `DIPContBas.py` does not contain independent data, only derived variables in convenient representations, e.g., the gravitational acceleration at the Earth's surface and the ion gyrofrequency. Selected boundary values are collected in Pandas dataframes.

```
[6]: print('Ion gyrofrequency [rad/s]: {:.1f}'.format(DCB.FGiLTI))
```

```
Ion gyrofrequency [rad/s]: 172.3
```

1.6 Satellite orbits around perigee

The sixth section of `DIPContBas.py` provides two functions for computing altitudes and horizontal distances along the orbits of satellites around perigee. Numerical integration by means of the Stoermer-Verlet method is implemented in `OrbitAtPerSV()` while `OrbitAtPerPA()` offers the polynomial approximation

$$\begin{aligned} z(t) &= z_{\text{per}} + \frac{a_{\text{per}}}{2} t^2, \\ x(t) &= \frac{R_E V_{\text{per}}}{R_{\text{per}}} \left(t - \frac{a_{\text{per}}}{3 R_{\text{per}}} t^3 \right), \end{aligned}$$

with the acceleration a_{per} at perigee given by

$$a_{\text{per}} = \frac{GM_E}{R_{\text{per}}^2} \frac{R_{\text{apo}} - R_{\text{per}}}{R_{\text{apo}} + R_{\text{per}}}.$$

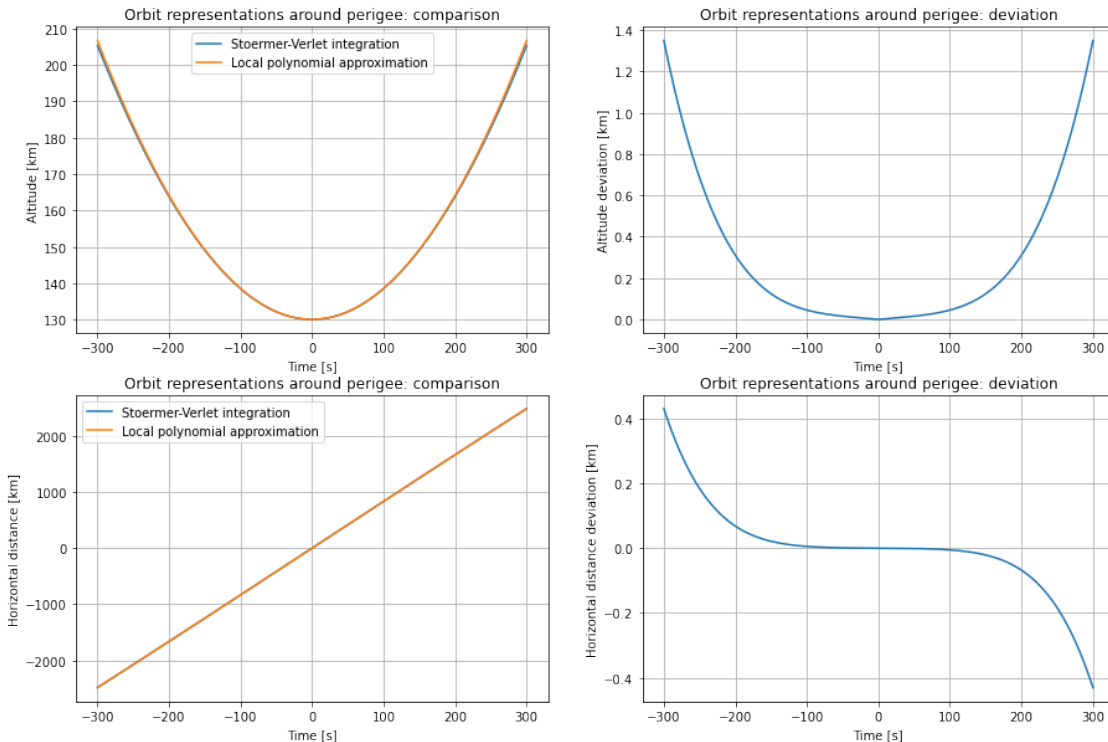
Here R_E is the Earth's radius, z_{per} , R_{per} , and V_{per} are the altitude, geocentric distance, and satellite velocity at perigee, R_{apo} is the geocentric distance at apogee, and G is the gravitational constant, and M_E is the mass of planet Earth. The two orbit representations around perigee are compared below.

```
[7]: import matplotlib.pyplot as plt
      zSatA_SV, xSatA_SV, tSatA_SV = DCB.OrbitAtPerSV(DCB.zPerA, DCB.zApoA)
      zSatA_PA, xSatA_PA, tSatA_PA = DCB.OrbitAtPerPA(DCB.zPerA, DCB.zApoA)
      plt.figure(figsize=(15, 10))
      plt.subplot(221)
      plt.plot(tSatA_SV, zSatA_SV/1e3, label='Stoermer-Verlet integration')
```

```

plt.plot(tSatA_PA,zSatA_PA/1e3,label='Local polynomial approximation')
plt.title('Orbit representations around perigee: comparison')
plt.xlabel('Time [s]')
plt.ylabel('Altitude [km]')
plt.legend()
plt.grid()
plt.subplot(222)
plt.title('Orbit representations around perigee: deviation')
plt.plot(tSatA_SV,(zSatA_PA-zSatA_SV)/1e3)
plt.ylabel('Altitude deviation [km]')
plt.xlabel('Time [s]')
plt.grid()
plt.subplot(223)
plt.plot(tSatA_SV,xSatA_SV/1e3,label='Stoermer-Verlet integration')
plt.plot(tSatA_PA,xSatA_PA/1e3,label='Local polynomial approximation')
plt.title('Orbit representations around perigee: comparison')
plt.xlabel('Time [s]')
plt.ylabel('Horizontal distance [km]')
plt.legend()
plt.grid()
plt.subplot(224)
plt.plot(tSatA_SV,(xSatA_PA-xSatA_SV)/1e3)
plt.title('Orbit representations around perigee: deviation')
plt.xlabel('Time [s]')
plt.ylabel('Horizontal distance deviation [km]')
plt.grid()

```



1.7 Supplementary variables and arrays

The seventh section of `DIPContBas.py` contains supplementary variables and arrays, e.g., allowing for more compact plot functions.

```
[8]: print('Default color palette for vertical profiles:',DCB.cGrdM)
```

```
Default color palette for vertical profiles: ['blue', 'green']
```

1.8 Length scales expressed in kilometers

In the eighth section of `DIPContBas.py`, length parameters and arrays are provided also in kilometers.

```
[9]: print('Base of the LTI region [km]:',DCB.zBot_km)
```

```
Base of the LTI region [km]: 100.0
```

End of DIPContBas.ipynb