大阪市立科学館研究報告 29, 25 - 30 (2019)

Python を利用した天文・気象情報の活用

江 越　航 *

概　要

Python は最近広く使用されている汎用のプログラミング言語である。各種数値計算や画像処理等のライブラリが豊富にそろっており、天文計算専用のライブラリも用意されている。これを用いれば、日の出・日の入り等の暦の計算や、恒星・惑星の位置、国際宇宙ステーションの軌道の計算、星図のプロットなどを容易に行えるようになる。

また気象分野に関しても、観測データをグラフ化しリアルタイムで状況を確認したり、数値予報の各種データを地図と組み合わせて表示したりするなどの応用が可能になる。

本稿では Python を利用して、各種天文・気象情報を活用するための方法について述べる。

1. はじめに

Python は最近広く使用されている汎用のプログラミング言語である。Linux のほか多くの OS 上で使用が可能であり、様々な分野のライブラリが豊富にそろっている。科学技術計算が得意で、線形代数や微積分、フーリエ解析、統計解析などの計算を行うことができる。特に近年は、機械学習のプログラムを書く際の言語として利用されており、ディープラーニングによる人工知能の技術開発のためにも広く用いられている。

天文計算専用のライブラリについても、既に作成されたものがそろっており、日の出・日の入り等の暦の計算や、恒星・惑星の位置、国際宇宙ステーションの軌道の計算を行い、結果を星図にプロットすることも容易に行える。

また気象分野に関しても、観測データを読み込み、処理してグラフ化し、リアルタイムで現在の気象状況を確認したり、数値予報の各種データを解析して、必要データを地図と組み合わせて表示したりするなどの応用が可能になる。

各種解析データは、自動化してホームページ上に掲載できるようにすると、最新情報を随時確認できるようになる。また、展示場で展示と組み合わせてデータを表示すれば、現在の展示をさらに効果的に活用することが可能になる。

*大阪市立科学館学芸課
e-mail:egoshi@sci-museum.jp

そこで本稿では Python を利用して、各種天文・気象情報を活用するための方法について述べる。

2. サーバーの構築

2-1. Linux サーバーの構築

天文・気象に関する情報をリアルタイムで表示するためには、常時起動しているパソコンがあると便利である。そこで今回は、以前使用していた 2003 年発売のノートパソコン Panasonic Let's Note CF-W2 を Linux サーバーとして利用することにした。

ただし手持ちの機種は既にハードディスクドライブが故障して使用できなくなっている。そこで、外部 USB メモリを接続し、Linux (Lubuntu) をインストールした。

しかしこの機種には USB ブートの機能がなく、このままではインストールした Linux を起動することができない。そこで別途、USB メモリ上の Ubuntu に対するブート CD を作成した。これにより、CD-ROM 起動を経由して、USB メモリ上の Linux を起動することが可能になった。

2-2. Python のインストール

インストールした Lubuntu のパッケージには、既に Python が含まれているが、統合開発環境(IDE)があると使いやすい。そこで Python をプログラミングするための IDE としてよく使用される Anaconda をインストールした。Anaconda を使用することで、追加パッケージのインストールや、バージョンの管理も簡単に行えるようになる。

Linux サーバーには、最初からインストールされている Python と、Anaconda の Python が併存する形になる。さらに、Python にはバージョン 2 (Python2) とバージョン 3 (Python3)の 2 系統がある点も注意が必要である。

通常、Linux に最初からインストールされている Python を使用する場合は、プログラムコードの最初に次のように記載する。

```
#!/usr/bin/python で python2.7
#!/usr/bin/python3 で python3.5
```

別途インストールした Anaconda の Python を使用するためには、プログラムコードの 1 行目に

```
#!/usr/bin/env python3
```

と記載する必要がある。

3. 暦の計算

3-1. 日の出・日の入りの計算

Python で天文計算を行うには、既に専用のライブラリがあるため、一から計算式を作らなくても、ライブラリを呼び出すだけで結果を得ることができる。特に Python の天文計算ライブラリの 1 つである PyEphem は、簡単に日の出・日の入り時刻の計算が可能である(ephemeris:天体暦の意)。ライブラリの使用前には、Anaconda から PyEphem をインストールしておく必要がある。

プログラムコードは、以下のように書く。

まず、Pyephem をインポートする。

```
import ephem
```

次に観測者のクラスのインスタンス(ここでは osaka)を作成する。東京と大阪に関しては、次のように記述するだけで設定できる。

```
osaka = ephem.city('Osaka')
```

その他の都市で、緯度と経度を直接設定する場合は、次のようにする。この際、緯度、経度は数値でなく、文字列として設定する。

```
osaka = ephem.Observer()
osaka.lat='34.6883'
osaka.lon='135.4933'
```

次に作成した観測者 osaka に、現在の日時を設定する。Pyephem で使用する時刻は、世界時で設定する必要がある。

```
osaka.date = datetime.datetime.utcnow()
```

次に計算対象とする天体、ここでは太陽のインスタンス(sun)を作成する。

```
sun = ephem.Sun()
```

以上の準備をもとに、次のメソッドを実行すると、現在時刻から見て、一番早い日の出の時刻を計算する。

```
print(osaka.next_rising(sun))
```

この結果、次のように日の出の時刻が表示される。なおここで表示される時刻は世界時である。

```
2019/4/30 20:08:19
```

日本標準時で表示するには、ephem.localtime メソッドを使用して、次のように記載する。

```
print(ephem.localtime(osaka.next_rising
 (sun)))
```

この結果は以下のように、日本時間で日の出時刻が表示される。

```
2019-05-01 05:08:18.000002
```

日の入り時刻に関しても、同じ要領で計算することができる。

3-2. 日の出・日の入り時刻線の表示

前項の手法で多くの地点の日の出・日の入り時刻を計算すれば、等時刻線を作成することも可能である。これには等高線図を作成すればよい。等高線図は、グラフ作成のライブラリ Matplotlib を使用して、次のように記述することで作成できる。

```
plt.contour(X, Y, Z)
```

ここで X、Y、Z はそれぞれ、二次元のリストである。各 X、Y で指定される場所に対する Z(ここでは日の出時刻)を与えることで、等高線図を描くことができる。

等高線図の属性は、次のように指定する。

```
cont = ax.contour(lon, lat,
 llocation_time, 10, colors=['black'],
```

```
 linewidths=(1),
 levels=np.arange(0, 86400 + 1, 3600),
 transform=ccrs.PlateCarree())
```

4 番目の 10 は、10 分毎に等高線を引くことを意味する。また、levels でも等高線の本数と間隔を指定できる。

さらに contour が返すインスタンス(cont)を使って、等高線にラベルを付加できる。

```
ax.clabel(cont, cont.levels, fmt=fmt)
```

以上の要領で作成した初日の出の時刻線を図 1 に示す。なお背景の地図は、後に述べる Cartopy という Python モジュールを利用している。

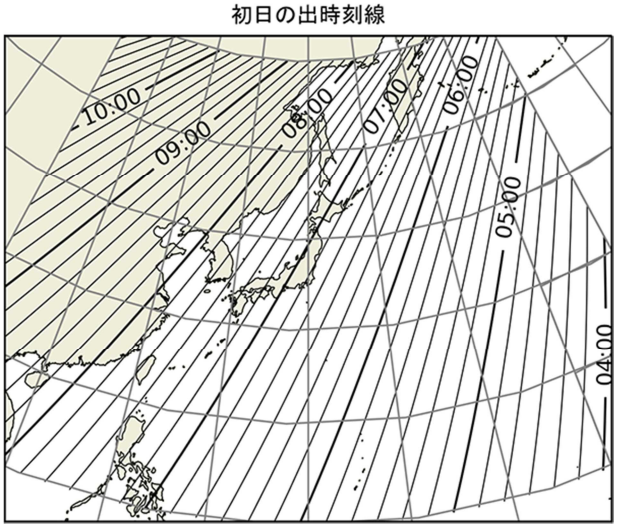

図1　初日の出時刻線

3-3. 星図の描画

Python では、星の位置(赤経・赤緯)を指定すれば、任意の場所・時刻における星の見える方向も、簡単に求めることができる。特に明るい星 94 個については、既にデータベースに登録されており、直接名前を指定するだけで位置を設定できる。

```
st = ephem.star('Betelgeuse')
st.compute(observer)
```

1 行目でベテルギウスのインスタンスを st として作成し、2 行目で観測者を基準にした座標を計算している。

この後、st に関するメソッド st.az、st.alt を実行することで、方位角、地平高度が計算できる。なお、ここで計算される値はラジアン単位である。以下のコードでは、座標を与え、星をプロットするサブルーチンを呼び出している。

```
star_plot(st.az,np.rad2deg(st.alt),
 'o', 2)
```

リストにない星を新たに追加するには、stars フォーマットに沿った次のようなテキストファイルのデータ

```
名前,f(fixed)|S(Star)|スペクトルタイプ,RA|
 固有運動,Dec|固有運動,等級,分点,オプション
Sirius,f|S|A0,6:45:09.3|-546.01,-16:42:
 47|-1223.08,-1.44,2000,0
```

を作成する。これを、一行ずつ読み込み、

```
star = ephem.readdb(line)
stars_plus[star.name] = star
```

とすれば、新しい星のオブジェクトを作成できる。

星座線を描画するには、例えば北斗七星の場合は次のように星の組み合わせをリストとして用意しておく。

```
star_lines =
 [['HIP54061','HIP53910'],['HIP53910','
 HIP58001'],['HIP58001','HIP59774'],['H
 IP59774','HIP62956'],['HIP62956','HIP6
 5378'],['HIP65378','HIP67301'],['HIP59
 774','HIP54061'], # 北斗七星
```

これを読み込んで、各星の座標を計算し、線で結ぶ処理を行う。

実際に星図を描画するには、グラフライブラリである Matplotlib を利用して、極座標のグラフとしてプロットする

```
import matplotlib.pyplot as plt
ax = fig.add_subplot(111,
 projection='polar')
ax.plot(azimuth, 90 - altitude, marker =
 mk, markersize=mksize, color = clr)
```

星図の場合は、中心が高度 90 度のため、動径方向の長さは 90 度から引いた値とする。こうして作成した星図が図 2 である。

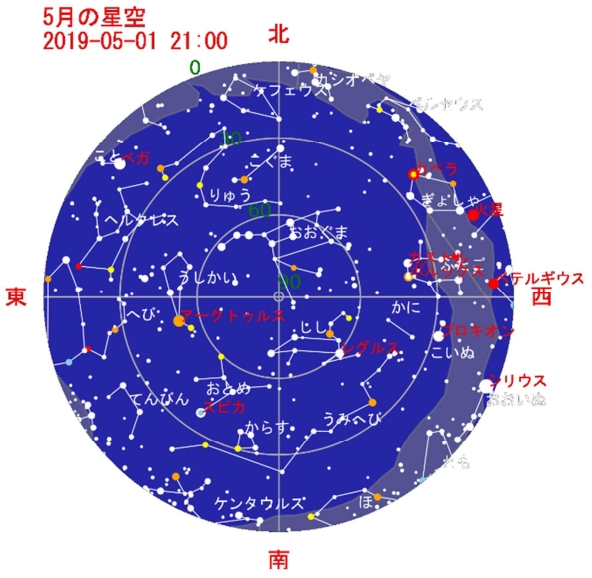

図2 星図の描画

3-4. 国際宇宙ステーション

Pyephem では人工衛星に関しても、軌道要素を設定することで、位置を計算することが可能である。特に地球を周回する人工衛星に関しては、NORAD(北アメリカ航空宇宙防衛司令部)が提供する「2 行軌道要素形式(TLE)」をそのまま読み込むことができる。[1]

```
iss = ephem.readtle("ISS (ZARYA)",
    "1 25544U 98067A ... ",
    "2 25544  51.6361 ... ")
```

この軌道要素は定期的に更新されてホームページで公開されているので、プログラムコード中に要素を記載せず、直接ホームページから読み込んで設定してもよい。

以上の人工衛星のオブジェクトを設定し、計算する場所 osaka と、開始時刻 t をあらかじめ設定の上

```
osaka.date = t
iss.compute(osaka)
```

とすれば、軌道計算が行われる。結果については、

```
rise_t, az_rise, max_t, alt_max, set_t,
 az_set = osaka.next_pass(iss)
```

とすることで、各変数に、設定時刻から一番近い次回のパスの出没時刻、出没方位、最大迎角、最大迎角時刻が読み込まれる。

上記で計算されるのは、可視か不可視かにかかわらず、すべてのパスである。例えば夜空に国際宇宙ステーションを見たい場合は、太陽が沈んで辺りが暗くなっているが、上空を飛行する国際宇宙ステーションには太陽光が当たっているという条件を選び出す必要がある。これは、以下のようにすればよい。

```
if iss.eclipsed is False and
 np.rad2deg(sun.alt) < -6 :
```

上記は、国際宇宙ステーションが食となっておらず、太陽高度が地平線下 6 度以下であるという条件を選んでいる。

この結果、選ばれたパスを、前項の星図上に表示する処理を行ったのが図 3 である。

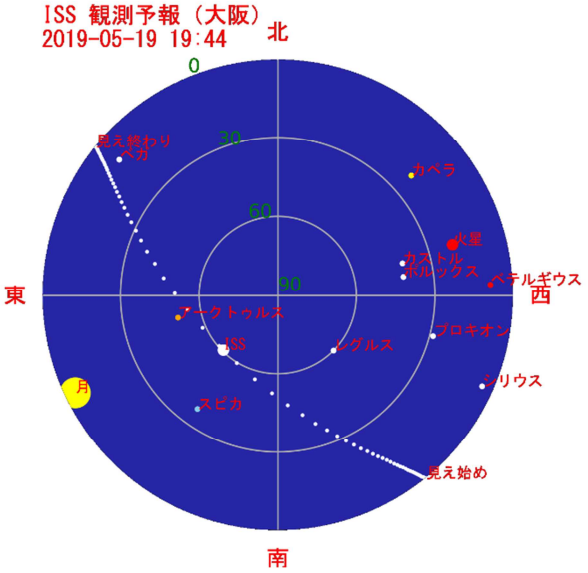

図3 国際宇宙ステーションの軌道の描画

4. 気象分野への応用

4-1. グラフの作成

科学館屋上には気象測器を設置している。この機器で測定したデータをグラフ化し、リアルタイムで表示することができれば、気象展示の活用の幅を広げることができる。機器に付属した制御プログラムでグラフを表示することはできるが、表示をカスタマイズすることはできず、思った通りのグラフにはなっていない。そこで、Python のグラフライブラリ Matplotlib を利用して、グラフを描くことを試みた。

測定データは次のように、何らかの形でテキストデータとしても提供されている。

```
  Date     Time      Temp    Bar
19/05/18    0:01    20.8    1018.1  .../
```

これを読み込んで、リアルタイムでグラフとしてプロットする。

データがテキストとして用意されている場合、NumPy のライブラリ genfromtxt を利用して、csv ファイルを読み

込んで、配列を作ることが可能である。

```
array_a2 = np.genfromtxt(fname,
 delimiter=',', dtype=str)
```

データを読み込むことができれば、次のように必要列を配列として変数に読み込む。

```
dates = list(array_b[0])
times = list(array_b[1])
temp = list(array_b[2])
bar = list(array_b[16])
```

あとは次のように、Matplotlib のグラフ表示の設定を行う。この結果、図 4 のようなグラフを描くことができる。

```
ax3 = fig.add_subplot(423, sharex=ax1)
ax3.tick_params(axis = 'both',
 labelsize=7)
ax3.set_ylabel('hPa')
ax3.set_title(u"気圧",
 fontproperties=fontprop)
ax3.plot(dtime, bar, color = "gray",
 linewidth = 0.7)
```

図4 気圧の変化のグラフ

4-2. 数値解析データの利用

現在の天気予報は、大型計算機による数値予報をもとに作成されている。数値予報の結果は格子点データとして、国際気象機関（WMO）が定める GRIB 共通フォーマットに従って提供されている。このデータは地球大気をメッシュ状に区切った各格子点の様々な気象要素が含まれることから、必要なデータを選択して地図上に表示できれば、現況の気象状況をより幅広く理解できるようになる。そこで、数値解析データを描画することを試みた。

気象庁の解析データ（GPV : Grid Point Value 格子点値）は、気象業務支援センターから購入することができるが、今回はアメリカ海洋大気庁（NOAA）が提供している NOMADAS（NOAA 運用モデルアーカイブ配布システム）を利用することとした。[2]

NOMADAS のホームページから、データセットとして GDAS の grib filter と進み、必要な日付と時刻を選択する。 続いて現れる GRIB Filter の画面で、必要なデータを選択する。例えば 850hPa 面の温度のデータを取得するには、850mb と TMP の項目をチェックする。以上の手順で必要な数値予報データをダウンロードすることができる。またデータのダウンロードは、URL パラメータとして必要データを指定することもできる。そのため、プログラム上でデータを自動でダウンロードすることも可能である。

ダウンロードしたファイルは、GRIB2 フォーマットに準じたバイナリ形式となっている。このファイルをデコードするツールも、アメリカ海洋大気庁（NOAA）が wgrib2 というツールを提供していることから、これをインストールして利用した。[3]

wgrib2 は、Linux のターミナルから

```
wgrib2 filename -undefine out-box 110:170
 10:60 -csv outfile.csv
```

とすることで、csv ファイルとして、各経度・緯度での気温の値を得ることができる。

今回は処理の自動化のために、wgrib2 の上記のコマンドを、次のように Python のプログラム中からシェルスクリプトとして呼び出して利用した。

```
subprocess.call(['./wgrib2.sh'],
 shell=True)
```

以上の手順により、必要とする格子点データが含まれるテキストファイルを作成することができる。

4-3. 数値解析データの描画

前項の手順により気象データを準備できれば、グラフライブラリを利用して描画することが可能になる。気温分布は、pcolor メソッドによる 2 次元カラー等高線図を用いて表示すると分かりやすい。

```
ax.pcolor(lon2, lat2, t, cmap='jet',
 alpha=0.5, edgecolors='none')
```

上記で jet はカラーマップの定義の 1 つで、値の変化に応じて、青から黄色、赤へと変化をつけるものである。こうして表示した気温分布が図 5 である。

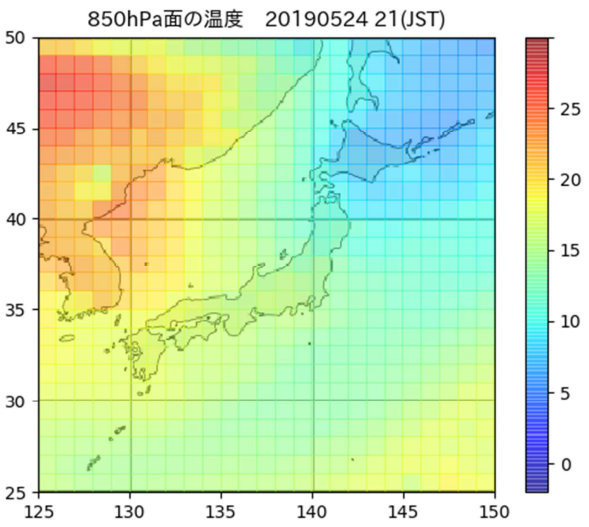

図5　850hPa での気温の図

風のデータは、streamplot メソッドを使用して流線による表現にすると分かりやすい。このためには、次のように各緯度、経度に対して、u、v、2 つの値を指定する。

```
ax.streamplot(lon2, lat2, u, v,
 linewidth=1, density=3,
 color=magnitude)
```

こうして表示した風のデータが図 6 である。

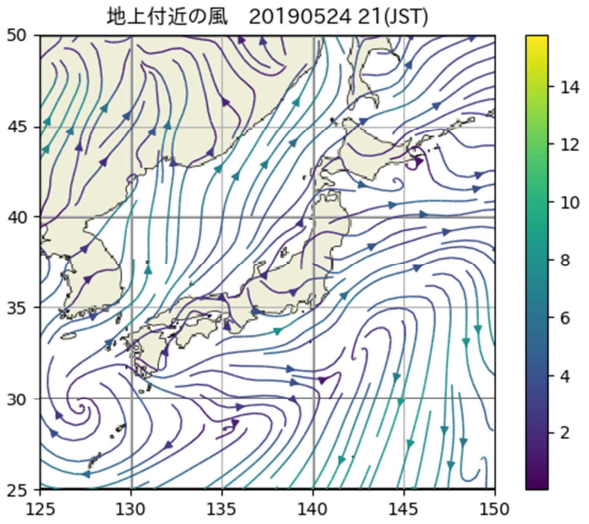

図6　地上付近の風

4-4. 地図の描画

数値解析データの結果は、地図と重ねて表示できると効果的である。Python では Cartopy というモジュールを利用することで、地図を表示することができる。Cartopy はイギリス気象庁（UK Met Office）によって開発されたパッケージで、地理空間データ処理を行い、処理結果を地図とともに表示できるものである。

例えば正距円筒図法で書かれた地図を描くためには以下のようにする。

```
import cartopy.crs as ccrs
ax =
plt.axes(projection=ccrs.PlateCarree())
ax.set_extent([110,170,10,60],ccrs.Plat
 eCarree())
ax.coastlines()
```

このようにに地図座標を定義しておけば、あとはグラフライブラリ Matplotlib を利用して、数値解析データを地図に重ねて表示することが可能となる。

5. おわりに

プログラミング言語 Python を用いると、様々な天文・気象情報を取り扱うことが可能になる。あらかじめ種々の天文計算やグラフライブラリが用意されているので、計算方法に精通していなくても、必要な図を描くことが可能になる。天文現象は各地域で異なることも多く、時々刻々変化することから、自分自身で自由にデータが扱えるのは有用である。

また、当館では気象データの収集を行っているが、こうしたデータをリアルタイムでグラフ化して表示するとともに、各種数値予報のデータと組み合わせて表示することで、より現況を深く理解できるようになると期待される。

Python の利用は天文・気象情報を取り扱う上で、多くの可能性を秘めている。今後も更なる活用方法を検討していきたい。

参考文献

[1] NORAD Two-Line Element Sets Current Data
http://www.celestrak.com/NORAD/elements/stations.txt
[2]NOAA Operational Model Archive and Distribution System
http://nomads.ncep.noaa.gov/
[3]wgrib2: wgrib for GRIB-2
https://www.cpc.ncep.noaa.gov/products/wesley/wgrib2/