Python を利用した日食図の描画

江 越　航 *

概　要

日食は天文現象の中でも分かりやすく印象深い現象であるため、一般市民の関心も高い。この現象の見られる範囲や時間経過を説明するために、しばしば日食図が用いられる。この図の作成には専門的な天文計算の知識が必要となることから、NASA や国立天文台が作成したものを使用することが多い。しかし自分で計算して描画することができれば、各地の実情に応じた図を描くことが可能になる。

近年、Python のライブラリが充実してきたことから、これを利用して日食図を描くことを試みた。本稿ではそのための手法について述べる。

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。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知られており、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広く用いられている。

Python は機械学習以外にも様々な分野のライブラリが豊富にそろっている。線形代数や微積分、フーリエ解析、統計解析など科学技術計算に関するライブラリも多い。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、筆者はこれを利用して日の出・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例を示した[1]。

天文計算の中でも日食の計算は、天文学に関心のある層の人々にとって興味ある分野である。そこで本稿では Python を利用して、日食図を描く方法について述べる。

2. 日食図

2020 年 6 月 21 日に、日本全国で部分日食を観測することができた。日食は観測する場所によって、見られる時刻や欠け具合が異なる。そこで日食の全体像を示すために、地球上で、いつ、どの範囲で、どのような日食が見られるか、全体像を示した図である日食図がよく用いられる。この図を見れば、皆既日食または金環日食が見られる地域、部分日食が見られる地域、欠け具合などを一目で知ることができる。

日食図としてよく用いられるのは、NASA Eclipse Web Site のページに掲載されている図である（図 1）[2]。このページには、紀元前 2000 年から西暦 3000 年までに起こる、すべての日食の日食図が掲載されている。

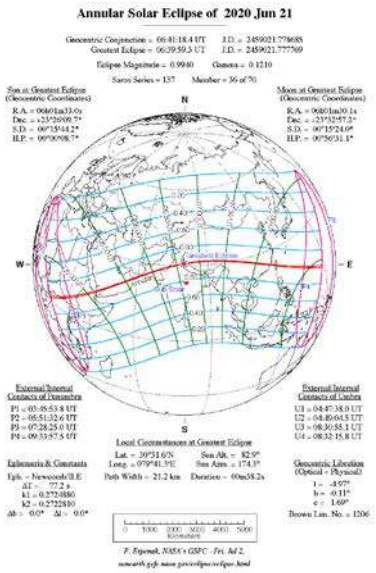

図 1　NASA Eclipse Web Site の日食図

日食図中に示された線の意味を図 2 に示した。日食の進行とともに、月の影が地球上で西から東へと通過していくことに対応して、図の左側の地域では日の出とともに日食が見え始め、右側の地域では日の入りとともに日食が見えなくなる。中心部分は月の本影が

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

地球上にかかる部分であり、皆既日食または金環日食が見られる地域である。中心線を挟んだ上下の部分は部分日食が見られる地域で、中心線から離れるほど、欠け具合が少なくなる。

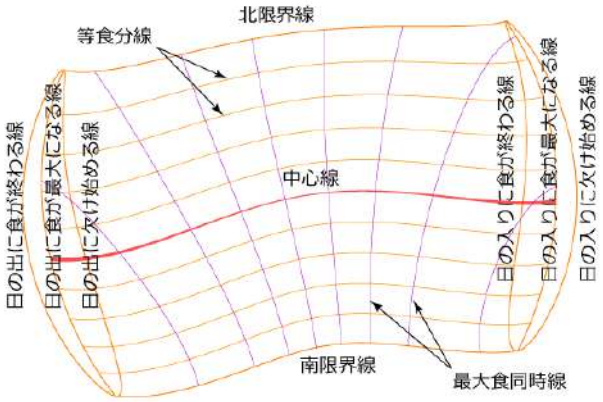

図2　日食図の説明

この日食図は、日食の全体像を見られるように、日の出から日の入りまでの範囲を同時に見られるように描かれている。しかし、日本での様子を知りたいときには、日本を中心に表示したい。また、日本付近のみを拡大して、さらに詳細な図が欲しい。そうした場合は、自分で日食図を描く必要がある。

3. 簡易的な方法による日食図の描画

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計算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、あるいは Skyfield を用いれば、簡単に任意の時刻の太陽・月の座標を得ることができる。ここでは、Skyfield を用いて計算する例を示す。

日食の進行状況を知るには、太陽・月の座標からこの 2 つの天体の重なり具合である食分を求めればよい。食分を計算するには図 3 のように、太陽・月の見かけの大きさ、および太陽と月の角距離が分かればよい。

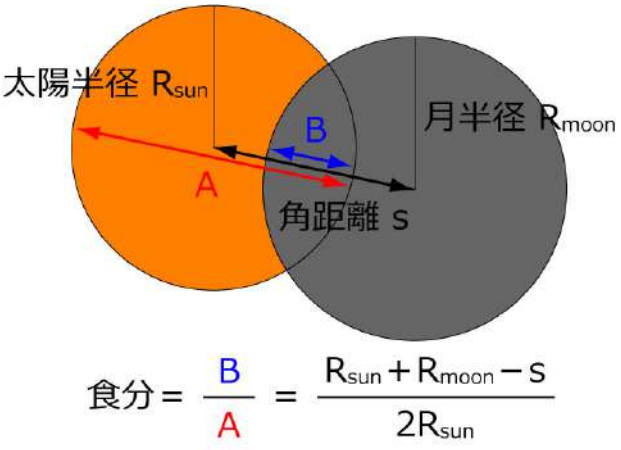

図3　食分の計算

観測地での太陽、月の座標は、あらかじめ定義した観測場所のオブジェクト osaka と、時刻 t を与えることで、以下の通り簡単に得ることができる。

```
sun_app =
 osaka.at(t).observe(sun).apparent()
moon_app =
 osaka.at(t).observe(moon).apparent()
```

太陽・月の見かけの大きさを計算するには、太陽の半径を r_sun で与えておき、太陽までの距離が skyfield のライブラリから km 単位で得られるので、この 2 つの値の比をとればよい。

```
r_sun = 696000
sun_dist = sun_app.distance().km
sun_rad = np.arctan2(r_sun, sun_dist)
```

太陽と月の角距離は、先に求めた太陽、月の座標に対して、separation_from 関数を用いることで、計算される。

```
app_sep =
 sun_app.separation_from(moon_app).radi
 ans
```

以上より食分は

```
percent_eclipse = (sun_rad + moon_rad -
 app_sep) / (sun_rad * 2)
```

で計算できる。

この計算を、日食が起こると見込まれる時間帯において、適当な時間間隔、例えば 1 秒ごとに行い、それぞれの時点での食分の値を求めていく。すると最初に食分が 0 以上になった時刻が欠け始め（第一接触）であり、再び食分が 0 になった時刻が食の終わり（第四接触）となる。その間の食分の最大値が、その場所における食の最大で、その時の時刻が最大食の時刻となる。

なお上記の計算において、Python の天文計算ライブラリで得られる月の座標は、月の重心の位置である。一方、日食の計算を行う場合は、正確には月の円盤の形状中心で計算する必要がある。そこで本来は月の位置を補正する必要があるが、ここではその誤差は無視している。

日食図を描画するためには、前項の手法で多くの地点の最大食分を計算する必要がある。同食分線を描画するには、ある範囲の緯度・経度 1 度ごとのグリッド上の点での最大食分を計算し、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 を使用して作成することができる。

地図の描画は、Python では Cartopy というモジュールを利用することで可能となる。等高線図を地図に重ねて表示すると、図のような同食分線を得られる。

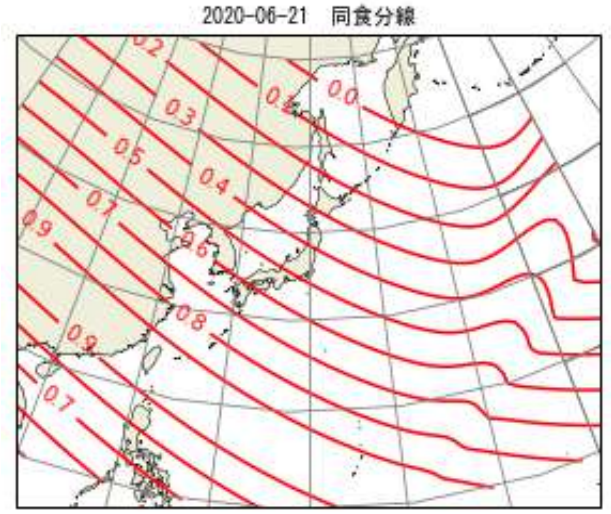

図4 簡易的な方法で計算した日食図

上記の方法で日食図を作成した場合、日本付近での値は比較的正しく計算されているが、端点に近づくにつれ、ずれが大きくなってしまう。これは、端点となる日の入り付近では、ちょうど太陽光が地表面に直角に当たる形になるため、時間当たりの変化量が大きくなるためと考えられる。

また、この方法で計算を行う場合、各グリッド上の値を計算するのに時間がかかるため、図全体を描画するには、数時間の計算が必要となってしまう。

4. 直接計算による日食図の描画

4-1. ベッセル要素

前項の方法では十分満足な日食図を描けなかったことから、長沢の本[3]を参考に、図 2 に示した日食図の各要素の線を直接計算することとした。

日食計算の際に便利なよう、太陽、月の位置関係に関するパラメータを表した数値をベッセル要素という。ベッセル要素では、ある時刻における月と太陽を結ぶ月影軸の方向、基準面における影の座標、本影・半影の大きさ等の要素の値が与えられている。

日食計算の際には、与えられたベッセル要素の数値をもとに、観測地点における日食の要素を計算することになる。

任意の時刻のベッセル要素の値を計算するには、ベッセル要素を多項式で近似した係数があると便利である。NASA Eclipse Web Site のページ[2]には、あらかじめこのベッセル要素を多項式で近似した係数のデータが掲載されているので、今回はこれを利用した。

4-2. 日出初き線等の計算例

例えば、日の出・日の入りに欠け始める線、または食が終わる線の計算を行うには、次のような手順を踏む。

月の半影は、中心(x_0, y_0)、半径 l_1^2 の円で

$$(x-x_0)^2+(y-y_0)^2=l_1^2$$

と表される。一方、地球外周楕円の方程式は

$$x^2+\frac{y^2}{1-E^2}=1$$

となる。ここで、E は離心率である。地球外周楕円は、長半径 1、離心率 E で描かれる。また、x_0、y_0、l_1、E はすべて時刻の関数である。日の出・日の入りに欠け始める線、または食が終わる線を得るには、この 2 つの曲線の交点の座標を求めればよい。

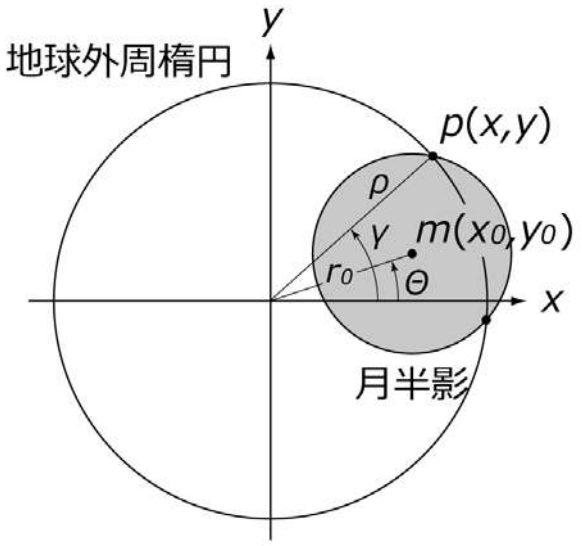

図5 地球外周楕円と月半影の交点の計算

計算の際、実際には極座標で行うのが便利である。先ほどの式を極座標で書き直すと、月の半影および地球外周楕円の方程式は

$$r_0^2+\rho^2-2r_0\rho\cos(\gamma-\theta)=l_1^2$$

$$\rho^2=\frac{1-E^2}{1-E^2\cos^2\gamma}$$

となる。交点の座標は、この 2 式から ρ、γ を求めることに対応する。

月の半影の式は

$$\cos(\gamma-\theta)=\frac{r_0^2+\rho^2-l_1^2}{2r_0\rho}$$

と書き直せる。交点の座標の計算は、最初に適当なρの値を仮定し、逐次近似によって、必要な精度に達するまで計算を繰り返す。

この計算を Python で記載すると、次のようになる。

```
while delta > 1e-7:
 cos_gamma_theta = (r0**2 + rho0**2 -
  l1**2) / (2 * r0 * rho0)
 gamma_theta =
  np.arccos(cos_gamma_theta)
 gamma = gamma_theta + theta

 rho2 = (1 - e2) / (1 - e2 *
  np.cos(gamma)**2)
 rho = np.sqrt(rho2)

 delta = abs((rho - rho0) /rho0)
 rho0 = rho
```

ρの値が変化しないようになれば、この時得られたρ、γ の組が、求める交点の座標になる。

日食図に描かれるほかの線についても順次計算を行うことで、最終的に図 6 のような日食図を描くことができた。

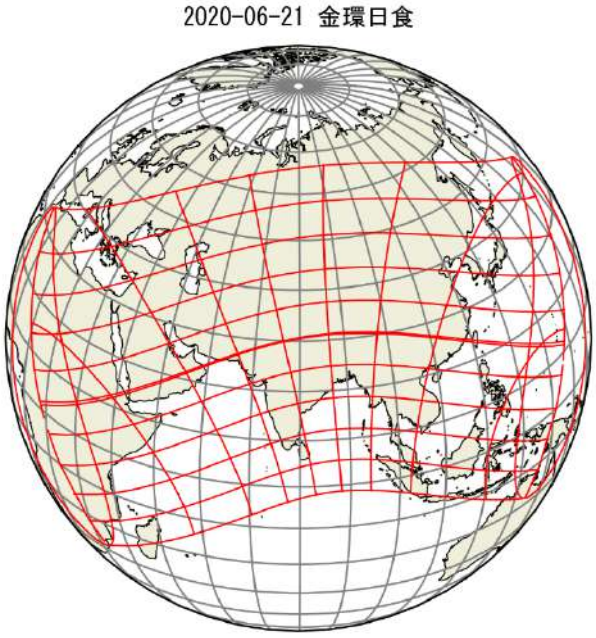

図6 Python で描いた日食図

Python で計算して得られた日食図に描かれる線の座標は、csvファイルに出力することで、地図描画ソフト QGIS に取り込むことができる。QGIS を使って正射図法(orthographic projection)で日本を中心にした世界図に、日食図の線を取り込んで描画した図を作成し、さらに Illustrator で装飾したものが図 7 である。

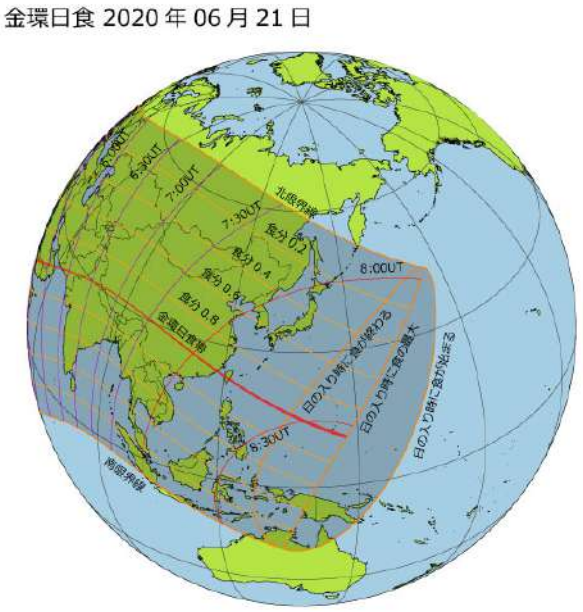

図7 日本付近を中心に描いた日食図

5. おわりに

今回、プログラミング言語 Python を用いて日食図を描画する方法について述べた。日食の際にその日食の経過を説明するためには、NASA が作成した日食図を引用するケースが多かったが、今回の日食では日本は日食図の端に位置するため、日本での様子を知るには必ずしも分かりやすいものでなかった。

ベッセル要素をもとに、自分自身で日食図を描けることができれば、さらに日本付近の詳細な日食図を描くことが可能になる。また遠い将来や過去の日食で、適当な日食図がないものであっても、自分で解説図を作ることが可能になる。

さらには、地球自転の遅れなどを考慮すると、日食が見られる範囲がどのように変化するかといった図も描くことが可能となる。

Python はほかにも天文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]江越 航, 大阪市立科学館研究報告, 29, p25-30(2019)

[2]NASA Eclipse Web Site, https://eclipse.gsfc.nasa.gov/eclipse.html

[3]長沢 工,「日食計算の基礎」, 地人書館, (2011)