

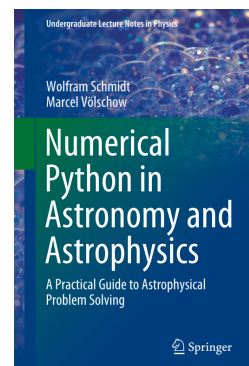
**Numerical Python in stromony and Astrophysics: A
practical guide to astrophysical problem solving
(Schmidt, W.; Völschow, M)**

por

Igo da Costa Andrade

Referência

SCHMIDT, W.; VÖLSCHOW, M. **Numerical Python in stromony and Astrophysics: A practical guide to astrophysical problem solving.** Switzerland, Springer, 2021.



Capítulo 2: Computação e exibição de dados¹

Exercícios

- 2.1** Compute a declinação do Sol para os equinócios e solstícios usando apenas funções trigonométricas do módulo `math` em um loop `for` explícito. Imprima os resultados e cheque se eles concordam com os valores computados usando NumPy nesta seção. Este exercício ajuda a compreender o que está por trás de um loop implícito.

Solução:

Variação anual da declinação do sol:

$$\delta_{\odot} = -\arcsin \left[\sin \epsilon_0 \cos \left(\frac{360^\circ}{365.24} \right) (N + 10) \right]$$

```
import math
from datetime import datetime, timedelta

year = 2024
jan1st = datetime(year,1,1)
eventos = [
    {
        "nome": "Equinócio de outono",
        "data": datetime(year,3,20)
    },
    {
        "nome": "Solstício de inverno",
        "data": datetime(year,6,20)
    },
    {
```

¹Título original: *Computing and Displaying Data*.

```

    "nome": "Equinócio de primavera",
    "data": datetime(year,9,22)
},
{
    "nome": "Solstício de verão",
    "data": datetime(year,12,21)
}
]

omega = 2*math.pi/365.24
ecl = math.radians(23.44)
for evento in eventos:
    diff = evento['data'] - jan1st
    N = diff.days
    delta = -math.asin(math.sin(ecl) * math.cos(omega * (N+10)))
    print(f"{evento['nome']} ({evento['data'].strftime('%d/%m/%Y')})")
    print(f"Declinação = {math.degrees(delta):.2f} deg.")

## Equinócio de outono (20/03/2024)
## Declinação = -0.91 deg.
## Solstício de inverno (20/06/2024)
## Declinação = 23.43 deg.
## Equinócio de primavera (22/09/2024)
## Declinação = -0.42 deg.
## Solstício de verão (21/12/2024)
## Declinação = -23.44 deg.

```



2.2 O contador do dia N na Eq. (2.1) pode ser calculado para uma data com a ajuda do módulo `datetime`. Por exemplo, o dia de equinócio vernal mp ano de 2020 é dado por

```

import datetime
vernal_equinox = datetime.date(2020, 3, 20) - \
    datetime.date(2020, 1, 1)

```

Então `vernal_equinox.days` resulta em 79. Defina um array N (equinócios e solstícios) usando `datetime`.

Solução:

```

import datetime
import numpy as np

N = np.array([
    (datetime.date(2024, m, d)-datetime.date(2024,1,1)).days for m, d \
        in zip([3, 6, 9, 12],[20, 20, 22, 21])
])
print(N)

## [ 79 171 265 355]

```



2.3 Uma fórmula mais precisa para a declinação do Sol leva em consideração a excentricidade $e = 0.0167$ da órbita da Terra:

$$\delta_{\odot} = -\arcsin \left[\sin(\epsilon_0) \cos \left(\frac{360^\circ}{365.24} (N + 10) + e \frac{360^\circ}{\pi} \sin \left(\frac{360^\circ}{365.24} (N - 2) \right) \right) \right]$$

Calcule a declinação assumindo uma órbita circular (Eq. 2.1), a declinação resultante da fórmula acima, a diferença entre esses valores, e o desvio relativo da aproximação circular em % para equinócios e solstícios e liste seus resultados em uma tabela. Certifique-se de que um número adequado de dígitos seja exibido para comparar as fórmulas.

Solução:

```
# Bibliotecas necessárias
import math
import numpy as np
import datetime

# Função auxiliar
def sun_declination(N, ecl, omega, e=0.0):
    delta = -np.arcsin(
        math.sin(ecl) * np.cos(omega*(N+10) + 2*e*np.sin(omega*(N-2)))
    )
    return delta

# Dados do problema
ecl = math.radians(23.44)
omega = 2*math.pi/365.24
e=0.0167

year = 2024
jan1st = datetime.date(year,1,1)
eventos = [
    {"nome": "Equinócio de outono", "data": datetime.date(year,3,20)},
    {"nome": "Solstício de inverno", "data": datetime.date(year,6,20)},
    {"nome": "Equinócio de primavera", "data": datetime.date(year,9,22)},
    {"nome": "Solstício de verão", "data": datetime.date(year,12,21)}
]

tbl_header = f"{'Evento ({year})':<30s} {'Decl. (Órb. Circular)':<20s} {'Decl. (Órb. Elíptica)':<20s}"

tbl = "="*len(tbl_header) + "\n"
tbl += tbl_header + "\n"
tbl += "-"*len(tbl_header) + "\n"

for evento in eventos:
    N = (evento['data'] - jan1st).days
    delta_circ = sun_declination(N=N, ecl=ecl, omega=omega, e=0.0)
    delta_elipse = sun_declination(N=N, ecl=ecl, omega=omega, e=e)
    erro = (delta_circ-delta_elipse)/delta_elipse
    row = f"{'{' + evento['nome'] + '':<30s} {'{' + evento['data'].strftime('%d/%m') + '':<30s} {'{' + math.degrees(delta_circ) + '':<20s} {'{' + math.degrees(delta_elipse) + '':<20s} {'{' + erro + '':<20s}"
```

```

        f"{math.degrees(delta_elipse):>20.2f}° " + f"{erro:>10.2%}"
    tbl += row + "\n"
tbl += "="*len(tbl_header)

print(tbl)

```

```

## =====
## Evento (2024)          Decl. (Órb. Circular) Decl. (Órb. Elíptica) Erro (%)
## -----
## Equinócio de outono (20/03)          -0.91°          -0.17°          440.87%
## Solstício de inverno (20/06)         23.43°          23.43°           -0.02%
## Equinócio de primavera (22/09)       -0.42°           0.33°        -227.79%
## Solstício de verão (21/12)          -23.44°         -23.44°           0.01%
## =====

```



2.4 A maior altitude a_{max} (também conhecida como culminação superior) de uma estrela medida a partir do plano horizontal de um observador na Terra é dada por:

$$a_{max} = \begin{cases} 90^\circ - \phi + \delta; & \text{se } \phi \geq \delta, \\ 90^\circ + \phi - \delta; & \text{se } \phi < \delta \end{cases}$$

onde ϕ é a latitude do observador e δ é a declinação da estrela. Calcule a_{max} na sua localização atual para as estrelas: Polaris ($\delta = 89^\circ 15' 51''$), Betelgeuse ($\delta = +07^\circ 24' 25''$) e Rigel ($\delta = -08^\circ 12' 15''$) e Sirius A ($\delta = -16^\circ 42' 58''$). Para conseguir os dois casos na (Eq. 2.3), use a função `where()` de numpy. Por exemplo,

```
np.where(phi <= delta, phi-delta, phi+delta)
```

compara `phi` e `delta` elemento por elemento e retorna um *array* com elementos de `phi-delta` se a condição `phi <= delta` é verdadeira e elementos de `phi+delta` caso contrário. Imprima os resultados com o número apropriado de dígitos significativos junto com as declinações.

Solução:

```

import pandas as pd
stars = [
    {
        "name": "Polaris", "sgn": "+", "D": 89, "M": 15, "S": 51
    },
    {
        "name": "Betelgeuse", "sgn": "+", "D": 7, "M": 24, "S": 25
    },
    {
        "name": "Rigel", "sgn": "-", "D": 8, "M": 12, "S": 15
    },
    {
        "name": "Sirius A", "sgn": "-", "D": 16, "M": 42, "S": 58
    }
]

```

```

for i, star in enumerate(stars):
    star["DMS"] = 0
    for j, label in enumerate(["D", "M", "S"]):
        star["DMS"] += star[label]/60**j
        if star["sgn"] == "-":
            star["DMS"] *= -1
    stars[i] = star

stars_df = pd.DataFrame(stars)

location = {
    "name": "Palmas",
    "lat": {
        "sgn": "-", "D": 10, "M":10, "S":2
    },
    "lon": {
        "sgn": "-", "D": 48, "M":19, "S":2
    }
}

delta = stars_df["DMS"]
phi = sum(location["lat"][label]/60**i for i, label in enumerate(["D", "M", "S"]))
phi = phi if location["lat"]["sgn"]=="+" else -phi
phi

## -10.167222222222222

upper_culmination = 90 + np.where(phi>=delta, -phi+delta, phi-delta)

stars_df["upper_culmination"] = upper_culmination
stars_df

##          name sgn   D   M   S      DMS  upper_culmination
## 0    Polaris   +   89  15  51   89.264167          -9.431389
## 1  Betelgeuse   +    7  24  25    7.406944          72.425833
## 2    Rigel     -    8  12  15   -7.804167          87.636944
## 3   Sirius A   -   16  42  58  -15.316111          84.851111

```

2.5 Um dia sideral é aproximadamente 3 min 56 s mais curto que um dia solar (24 h). Mostre que isso implica em $1^h \approx 0,9973$ h. Como você poderia modificar a definição de T na Sec. 2.1.2 para fazer uso desse fator sem utilizar Astropy units?

Solução:

$$1^h = \frac{24\text{h} - (3\text{min}56\text{s})}{24\text{h}} = \frac{24\text{h} - \left(\frac{3}{60}\text{h} + \frac{56}{3600}\text{h}\right)}{24\text{h}} \approx 0.9973\text{h}$$

- 2.6** Calcule e faça o gráfico da variação anual da duração do dia na sua localização geográfica, utilizando tanto a fórmula com correção de excentricidade do Exercício 2.3 quanto a função `get_sun()` de `SkyCoord`. Qual é o tamanho do desvio? Como seu resultado se compara com outros lugares mostrados na Fig. 2.5?

Solução:

```
import math
import numpy as np
import matplotlib.pyplot as plt
from astropy.coordinates import SkyCoord, EarthLocation
import astropy.units as u

# Posição geográfica Palmas (Tocantins)
obs = EarthLocation(
    lat=-10*u.deg-10*u.arcmin-8*u.arcsec,
    lon=-48*u.deg-19*u.arcmin-54*u.arcsec
)

phi = obs.lat

N = np.arange(365) # array com elementos 0, 1, 2, ..., 364
omega = 2*math.pi/365.24 # Velocidade angular da Terra
ecl = math.radians(23.44) # Obliquidade da Eclíptica

# cálculo da declinação do Sol para cada dia do ano
delta = -np.arcsin(math.sin(ecl) * np.cos(omega*(N+10)))
# Cálculo da duração do dia em horas solares
h = np.arccos(-np.tan(delta) * math.tan(phi.radian))
T = (np.degrees(2*h)/360) * u.sday.to(u.h)

fig, ax = plt.subplots(figsize=(6, 4))
ax.plot(N, T)
ax.set_xlabel("Dia")
ax.set_ylabel("Duração do Dia (h)")
ax.set_title("Variação anual da duração do dia em Palmas (TO)")
plt.grid(ls="dashed")
plt.show()
```

