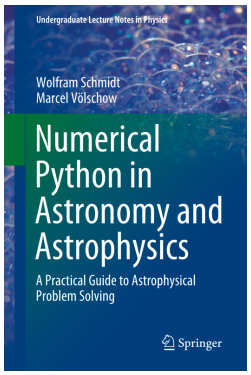
Resolução de Problemas do Livro

Numerical Python in stronomy 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 stronomy and Astrophysics**: A practical guide to astrophysical problem solving. Switzerland, Springer, 2021.

Capítulo 2: Computação e exibição de dados[1]

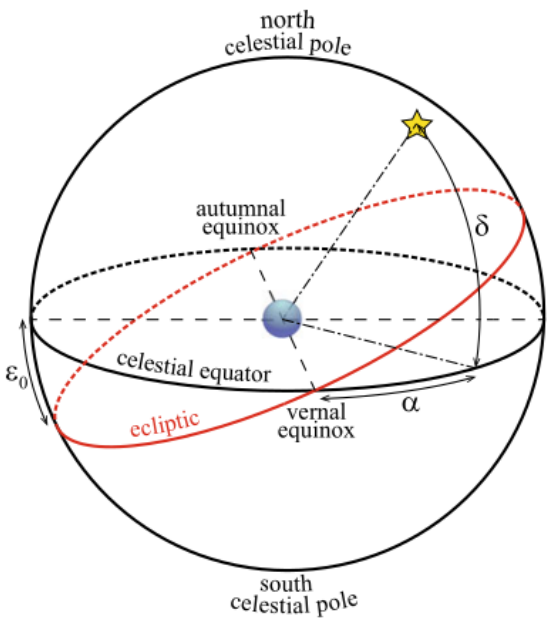
Resumo do capítulo: Os arrays do NumPy são essenciais para cálculos numéricos em Python, estendendo suas capacidades numéricas de forma notável. Por exemplo, eles podem ser usados como variáveis simples para avaliar uma expressão aritmética para muitos valores diferentes sem precisar programar um loop. Na primeira seção, combinamos o poder do NumPy e do Astropy para calcular as posições de objetos na esfera celeste. Além disso, introduzimos o Matplotlib para produzir gráficos a partir de dados de arrays. Outras aplicações são mostradas no contexto das leis de Kepler e forças de maré, como a impressão de tabelas formatadas e a plotagem de mapas vetoriais.

1 Astronomia Esférica

Esfera celeste com coordenadas α (ascensão reta) e δ (declinação) de um objeto estelar. O plano orbital da Terra (eclíptica) intersecta a esfera ao longo do círculo vermelho, que é inclinado pelo ângulo ε (obliquidade) em relação ao equador celeste. O equador celeste é a projeção externa do equador da Terra na esfera celeste. Os pontos de interseção da eclíptica e do equador celeste são os dois equinócios.

[1] Título original: *Computing and Displaying Data*.

Figure 1: Esfera Celeste



1.1 Declinação do Sol

Enquanto a declinação das estrelas é constante, a posição do sol muda no sistema equatorial no decorrer do período de um ano. Isso decorre da inclinação do eixo de rotação da Terra em relação à direção perpendicular da eclípica, a qual é denominada obliquidade e vale $\epsilon_0 = 23,44°$.A variação anual da declinação do Sol é dada por

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

em que N é a quandidade de dias a partir de 1˚ de janeiro.
Exemplo:

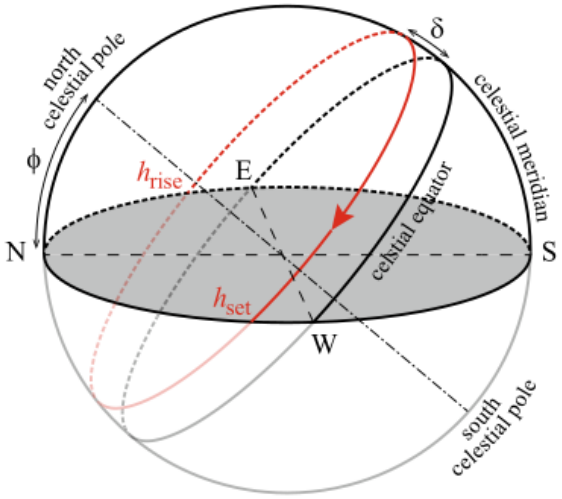
```
# Biblioteca importada
import math

# Dados do exemplo
N = 171 # dia do primeiro solstício
omega = 2*math.pi/365.24 # velocidade angular em rad/dia
ecl = math.radians(obliq) # obliquidade da eclípitica em rad

# Aproximação para a declinação do Sol
delta = -math.asin(math.sin(ecl) * math.cos(omega*(N+10)))
```

Resultado: Declinação do Sol em 20 de junho de 2020: $\delta_\odot = 23,43°$

1.2 Arco Diurnal

Denomina-se **arco diurnal** ao arco descrito na esfera celeste por um objeto em movimento.

Figure 2: Arco Diurnal



Arco diurno de uma estrela movendo-se ao redor da esfera celeste (círculo vermelho espesso) no sistema horizontal de um observador na latitude ϕ (o plano horizontal é mostrado em cinza). Como o plano equatorial está inclinado pelo ângulo $90° - \phi$ em relação ao plano horizontal, a culminação superior da estrela no meridiano é dada por $a_{\max} = 90° - \phi + \delta$, onde δ é a declinação. No sistema corotacional, a estrela nasce no ângulo horário $h_{\rm rise}$, atinge sua maior altitude quando cruza o meridiano em $h = 0$, e se põe no horizonte em $h_{\rm set} = -h_{\rm rise}$.

$$\cos h_{\rm set} = -\tan\delta \tan\phi,$$

em que δ é a declinação do objeto e ϕ é a latitude da posição do observador na Terra.

Exemplo:

Consideremos a estrela Betelgeuse na constelação de Orion. É uma gigante vermelha que está entre as estrelas mais brilhantes do céu. Sua declinação pode ser obtida com a ajuda de `astropy.coordinates`, o qual oferece uma função que pesquisa pelo nome de um objeto em bancos de dados online:

```
from astropy.coordinates import SkyCoord, EarthLocation

betelgeuse = SkyCoord.from_name('Betelgeuse')
table = ""
table += 46 * "="
table += "\n"
table += f"{'Name':>12s} | {'RA':>16s} | {'dec':>12s}\n"
table += 46 * "-"
table += "\n"
table += f"{'Betelgeuse':>12s} | {betelgeuse.ra:>16s} | {betelgeuse.dec:>12s}\n"
table += 46 * "="
print(table)
```

```
## ==============================================
##         Name |                RA |          dec
## ----------------------------------------------
##   Betelgeuse |  88.79293899 deg | 7.407064 deg
## ==============================================
```

```
delta = betelgeuse.dec
print(delta)
```

```
## 7d24m25.4304s
```

Suponhamos que desejamos determinar o comprimento do arco diurnal de Betelgeuse visto do Observatório de Hamburg ($\phi \approx +53°28'49''$).

```
import math
import astropy.units as u

# posição geográfica do observador
obs = EarthLocation(
  lat=53*u.deg + 28*u.arcmin + 49*u.arcsec,
  lon=10*u.deg + 14*u.arcmin + 23*u.arcsec,
)

# Latitude
phi = obs.lat

# h
h = math.acos(-math.tan(delta.radian) * math.tan(phi.radian))

# T
T = (math.degrees(2*h)/360) * u.sday

print(f"T = {T.to(u.h):.2f} (em dias siderais).")
```

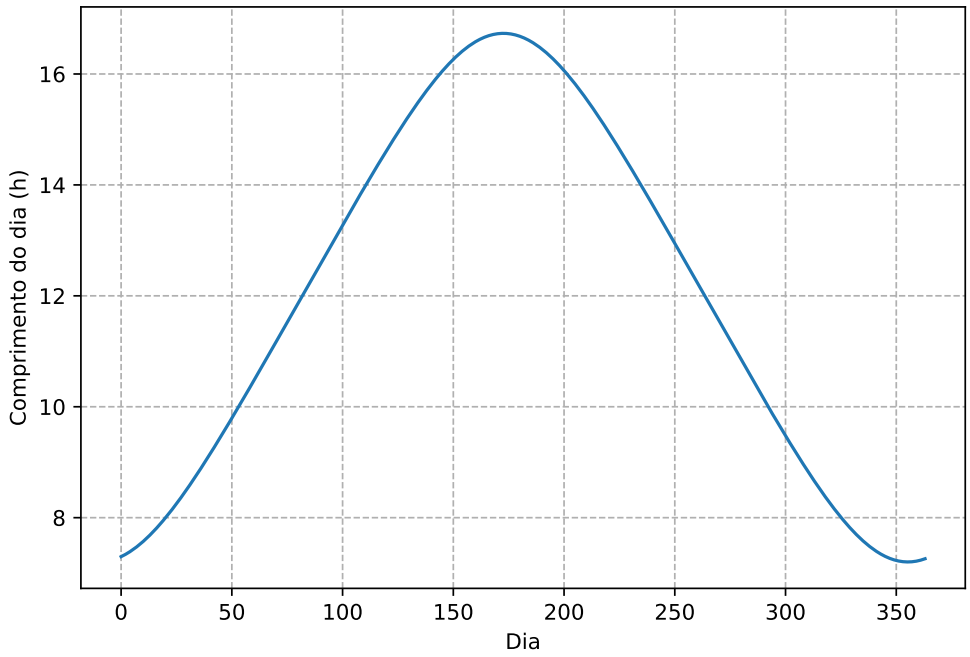
```
## T = 13.31 h (em dias siderais).
```

Exemplo: Variação do arco diurnal do Sol durante o ano

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

N = np.arange(364) # Array com elementos 0, 1, 2, ...,364
omega = 2*math.pi/365.24 # Velocidade angular da Terra em rad/dia
ecl = math.radians(23.44) # obliquidade da eclíptica em radianos

# Cálculo da declinação do Sol para todos os dias do ano
delta = -np.arcsin(math.sin(ecl) * np.cos(omega*(N+10)))

# Cálculo do comprimento 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)

# Gráfico
plt.plot(N, T)
plt.xlabel("Dia")
```

```
plt.ylabel("Comprimento do dia (h)")
plt.grid(ls='dashed')
plt.tight_layout()
plt.savefig('figure/chap-02/day-length.pdf')
plt.close()
```

Figure 3: Variação do arco diurnal Solar durante o ano



1.2.1 Arco diurnal para três cidades brasileiras

Vamos determinar a variação do arco diurnal do Sol para três cidades brasileiras: Teresina-PI, Palmas-TO e São Paulo-SP:

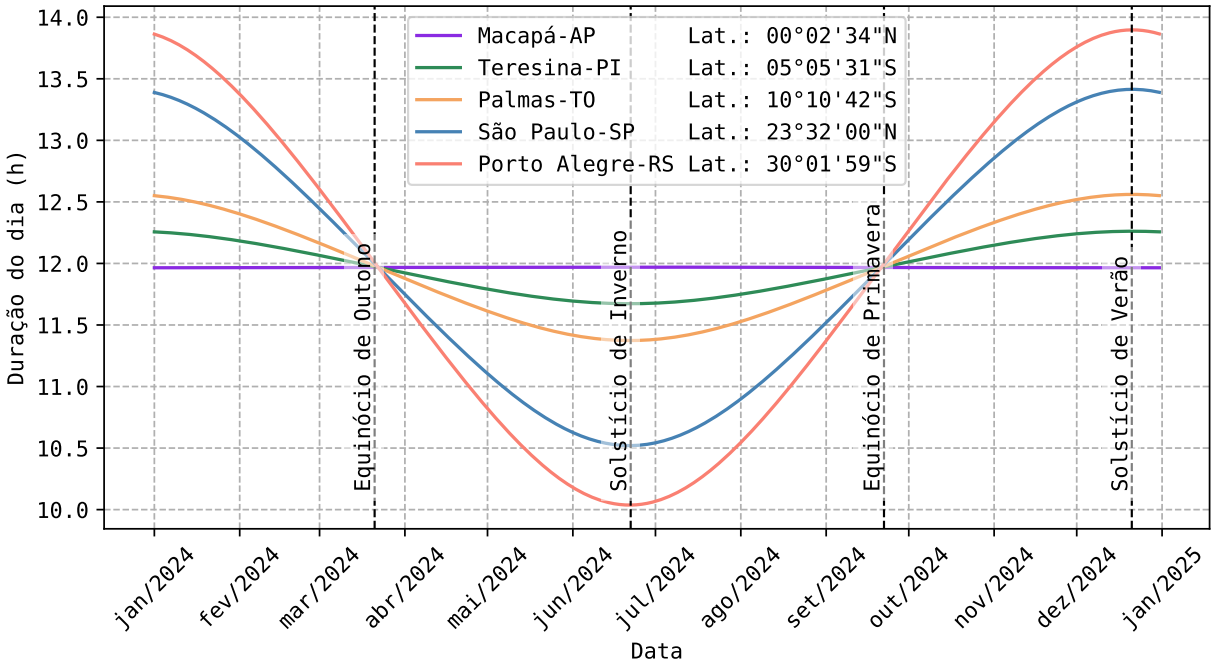
Cidade	Latitude (DMS)	Longitude (DMS)
Teresina-PI	5° 5' 31" S	42° 48' 13" W
Palmas-TO	10° 10' 42" S	48° 19' 47" W
São Paulo-SP	23° 32' 0" S	46° 37' 59" W

```
import numpy as np
from datetime import datetime, timedelta
import matplotlib.dates as mdates
plt.rcParams['font.family'] = 'monospace'

cidades = [
  {
    "nome": "Macapá-AP",
    "lat": (0,2,34),
    "lon": (51,3,37),
  },
  {
```

```
    "nome": "Teresina-PI",
    "lat": (-5,-5,-31),
    "lon": (-42,-48,-13),
  },
  {
    "nome": "Palmas-TO",
    "lat": (-10,-10,-42),
    "lon": (-48,-419,-47),
  },
  {
    "nome": "São Paulo-SP",
    "lat": (-23,-32,0),
    "lon": (-46,-37,-59),
  },
  {
    "nome": "Porto Alegre-RS",
    "lat": (-30,-1,-59),
    "lon": (-51,-13,-48),
  }
]

eventos = [
    "Solstício de Verão",
    "Equinócio de Outono",
    "Solstício de Inverno",
    "Equinócio de Primavera"
  ]
datas = [
    datetime(2024,12,21),
    datetime(2024,3,21),
    datetime(2024,6,22),
    datetime(2024,9,22)
]

def get_diaurnal_arc(cidade_dic, N = np.arange(364)):
  omega = 2*math.pi/365.24
  ecl = math.radians(23.44)
  delta = -np.arcsin(math.sin(ecl) * np.cos(omega*(N+10)))
  obs = EarthLocation(
    lat=cidade['lat'][0]*u.deg + cidade['lat'][1]*u.arcmin + cidade['lat'][2]*u.arcsec,
    lon=cidade['lon'][0]*u.deg + cidade['lon'][1]*u.arcmin + cidade['lon'][2]*u.arcsec
  )
  phi = obs.lat
  h = np.arccos(-np.tan(delta) * math.tan(phi.radian))
  T = (np.degrees(2*h)/360 * u.sday.to(u.h))
  return T

# Define start and end dates
start_date = datetime(2024, 1, 1)
end_date = datetime(2024, 12, 31)

# Calculate the number of days between start_date and end_date
num_days = (end_date - start_date).days + 1
```

```
# Create a numpy array of dates
date_array = np.array([start_date + timedelta(days=i) for i in range(num_days)])


N = np.arange(len(date_array))
cores = ["blueviolet", "seagreen", "sandybrown", "steelblue", "salmon", "darkslateblue"]
plt.figure(figsize=(8, 4.5))
for i, cidade in enumerate(cidades):
  cor = cores[i]
  T = get_diaurnal_arc(cidade, N)
  label=f"{cidade['nome']:<15s} Lat.: " + \
    "".join([f"{abs(li):02d}{s}" for li, s in zip(cidade['lat'], ['°', '\'', '\"'])])
  label += "N" if cidade['lat'][2]>=0 else "S"
  plt.plot(
      date_array, T, c=cor, label=label
    )

for evento, data in zip(eventos, datas):
  plt.text(
      data, plt.gca().get_ylim()[1]*0.72, evento, rotation=90,
      horizontalalignment='right', verticalalignment='bottom',
      bbox=dict(facecolor='white', edgecolor='none', alpha=0.5)
    )
  plt.axvline(x=data, color='k', linestyle='--', linewidth=1)

plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%b/%Y'))
# %b para mostrar abreviações dos meses
plt.xticks(rotation=45);
plt.xlabel("Data")
plt.ylabel("Duração do dia (h)")
plt.grid(ls="dashed")
plt.legend(loc='upper center')
plt.tight_layout()
plt.savefig("figure/chap-02/arco-diurnal-cidades-br.pdf")
plt.close()
```

Figure 4: Duração do dia ao longo do ano para três cidades brasileiras



1.3 Observação de Objetos Celestes

```
import astropy.units as u
from astropy.coordinates import \
  SkyCoord, EarthLocation, AltAz, get_sun

# Posição geográfica do observador
obs = EarthLocation(
  lat=(-10)*u.deg + (-10)*u.arcmin + (-42)*u.arcsec,
  lon=(-48)*u.deg + (-19)*u.arcmin + (-47)*u.arcsec
)

# latitude do observador
phi = obs.lat
```

Conforme utilzação do código acima, a latitude em Palmas-TO vale -10.1783333.

```
from astropy.time import Time

utc_shift = 2*u.hour # CEST time zone (+2h)
noon_cest = Time("2024-07-25 12:00:00") - utc_shift
```

```
elapsed = np.arange(0, 24*60, 5) * u.min
time = noon_cest + elapsed
frame_local_24h = AltAz(obstime=time, location=obs)
```

```
# estrela que desejamos observar
betelgeuse = SkyCoord.from_name('Betelgeuse')
betelgeuse_local = betelgeuse.transform_to(frame_local_24h)
```

```
# time-dependent coordinates of the Sun in equatorial system
sun = get_sun(time)
sun_local = sun.transform_to(frame_local_24h)
```

```
elapsed_night = elapsed[np.where(sun_local.alt<0)]
betelgeuse_night = \
  betelgeuse_local.alt[np.where(sun_local.alt<0)]

plt.figure(figsize=(8, 4.5))
plt.plot(
  elapsed.to(u.h), sun_local.alt, color='orange', label='Sol'
)
plt.plot(
  elapsed.to(u.h), betelgeuse_local.alt, color='red',
  linestyle=':', label='Betelgeuse (dia)'
)
plt.plot(
  elapsed_night.to(u.h), betelgeuse_night, color='red',
  label='Betelgeuse (noite)'
)

plt.xlabel("Tempo a partir do meio-dia [h]")
plt.xlim(0, 24)
```

```
plt.xticks(np.arange(13)*2)
```

```
#plt.ylim(0, 80)
plt.ylabel('Altitude [deg]')
plt.legend(loc='upper right')

plt.grid(ls='dashed')
plt.tight_layout()
plt.savefig("figure/chap-02/altitude_sol_betelgeuse_palmas_br.pdf")
plt.close()
```

Figure 5: Altitude do Sol e da estrela Betelgeuse vista de Palmas-TO