MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

CÁLCULO DE PERDAS DE GERAÇÃO FOTOVOLTAICA PARA TELHADOS REAIS

José Antonio Teixeira Filho

Relatório de Atividades desenvolvidas durante o período de Iniciação Científica do Programa PIBIC, orientada pelo Me. André Rodrigues Gonçalves e Dr. Fernando Ramos Martins.

INPE
São José dos Campos
2021

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

CÁLCULO DE PERDAS DE GERAÇÃO FOTOVOLTAICA PARA TELHADOS REAIS

José Antonio Teixeira Filho

Relatório de Atividades desenvolvidas durante o período de Iniciação Científica do Programa PIBIC, orientada pelo Me. André Rodrigues Gonçalves e Dr. Fernando Ramos Martins.

INPE
São José dos Campos
2021

MOTIVAÇÃO

O Brasil é um país com enorme potencial fotovoltaico, sendo um dos mais importantes daqueles que integram o cinturão solar (EPIA, 2010). Por isso, existe a necessidade de criar sistemas e processos otimizados que visam a maximização do uso da irradiância solar. Porém, os modelos presentes na literatura geralmente abrangem latitudes maiores, de países onde a tecnologia solar é mais solidificada. Isso causa extrapolações que nem sempre geram modelos otimizados para geração de energia solar (Cronemberger et al., 2012). Desta maneira, quando olhamos para o cenário brasileiro, vemos que há dúvidas dos instaladores de painéis fotovoltaicos quanto às perdas devido a decomposição do feixe de DNI, uma vez que os telhados reais raramente estão orientados ao Norte e na inclinação da latitude. O objetivo deste projeto é auxiliar o cenário nacional de aproveitamento solar avaliando a perda de eficiência na geração solar referente à orientação e inclinação dos telhados.

FUNDAMENTAÇÃO TEÓRICA

Conceitos básicos de aproveitamento solar e ambientação ao tema foram providos através da leitura dos capítulos 1 e 2 do livro “An Introduction to Solar Irradiation” de Muhammad Iqbal e da leitura da 2a edição do Atlas Brasileiro de Energia Solar, composto pelo esforço colaborativo do Laboratório de Modelagem e Estudos de Recursos Renováveis de Energia (LABREN), do Centro de Ciência do Sistema Terrestre (CCST) e do Instituto Nacional de Pesquisas Espaciais (INPE). Após a situação teórica, houve um período de habituação prática com a principal ferramenta utilizada no projeto, programação em linguagem Python.

ATIVIDADES REALIZADAS

Em um primeiro momento, foram aplicadas diretamente em Python, apenas com funções matemáticas, equações utilizadas no cálculo de variáveis solares. O resultado foi o código apresentado a seguir:

```python
import math as m
import datetime as dt

# Input

input_data = input('Insira a data desejada (DD/MM): ')
input_hora = input('Insira a hora local (HH:MM): ')
lon_padrao = float(input('Insira a longitude padrao, em modulo: '))
orientacao_lon_padrao = int(input('Insira a orientacao da longitude padrao (E=1, W=2 ou 0): '))
lon_local = float(input('Insira a longitude local, considerando Leste positivo : '))


# Transformacoes

t0 = dt.datetime.strptime("00:00:00", "%H:%M:%S")
data = dt.datetime.strptime(input_data, '%d/%m')
hora = dt.datetime.strptime(input_hora,"%H:%M")
horapadrao = hora - t0

# Calculos

dn = data.timetuple().tm_yday
gama = 2*m.pi*((dn-1)/365)
eq_tempo = ((0.000075 + 0.001868*m.cos(gama) - 0.032077*m.sin(gama) - 0.014615*m.cos(2*gama)
             - 0.04089*m.sin(2*gama))*13750.8)
eq_tempo_delta = dt.timedelta(seconds = eq_tempo)
correcao_longitudinal = abs(240*(lon_padrao - (abs(lon_local))))
correcao_longitudinal_delta = dt.timedelta(seconds = correcao_longitudinal)

# Analises

if orientacao_lon_padrao == 0 and lon_padrao < lon_local:
    LAT = horapadrao + correcao_longitudinal_delta + eq_tempo_delta
    LAT = LAT.total_seconds()
    LAT = dt.timedelta(seconds = LAT)
    print('LAT =', LAT)

elif orientacao_lon_padrao == 0 and lon_padrao > lon_local:
    LAT = horapadrao - correcao_longitudinal_delta + eq_tempo_delta
    LAT = LAT.total_seconds()
    LAT = dt.timedelta(seconds = LAT)
    print('LAT =', LAT)

elif orientacao_lon_padrao == 2 and lon_padrao > abs(lon_local):
    LAT = horapadrao + correcao_longitudinal_delta + eq_tempo_delta
    LAT = LAT.total_seconds()
    LAT = dt.timedelta(seconds = LAT)
    print('LAT =', LAT)

elif orientacao_lon_padrao == 2 and lon_padrao < abs(lon_local):
    LAT = horapadrao - correcao_longitudinal_delta + eq_tempo_delta
    LAT = LAT.total_seconds()
    LAT = dt.timedelta(seconds = LAT)
    print('LAT =', LAT)

elif orientacao_lon_padrao == 1 and lon_padrao < abs(lon_local):
    LAT = horapadrao + correcao_longitudinal_delta + eq_tempo_delta
    LAT = LAT.total_seconds()
    LAT = dt.timedelta(seconds = LAT)
    print('LAT =', LAT)

elif orientacao_lon_padrao == 1 and lon_padrao > abs(lon_local):
    LAT = horapadrao - correcao_longitudinal_delta + eq_tempo_delta
```

```
    LAT = LAT.total_seconds()
    LAT = dt.timedelta(seconds = LAT)
    print('LAT =', LAT)

print('Os valores utilizados para as analises, foram: ')
print('dn =', dn, 'dias')
print('gama = ', gama, 'rad ou', m.degrees(gama), '°')
print('Equacao do Tempo =', eq_tempo_delta)
print('Correcao Longitudinal =', correcao_longitudinal_delta)
```

METODOLOGIA E RESULTADOS PARCIAIS

Com maior familiaridade com a ferramenta, foi possível começar a utilizar as bibliotecas que são voltadas para cálculos solares, como a pvlib. O primeiro passo foi a geração de um diagrama solar (Figura 1), além da utilização da própria biblioteca para calcular algumas variáveis solares. O resultado foi o seguinte código:

```
import pandas as pd
import pvlib
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
from pvlib import solarposition
from pvlib import location


#Variaveis

lat, lon = -23.55, -46.63
tz = 'America/Sao_Paulo'
data=dt.datetime.strptime('21/12','%d/%m')
dn=data.timetuple().tm_yday
time = pd.date_range('2019-01-01 00:00:00', '2020-01-
01 00:00:00', closed='left', freq='H', tz=tz)
eq_time = pvlib.solarposition.equation_of_time_spencer71(dn)
hour_angle = pvlib.solarposition.hour_angle(time, lon, eq_time)
location = pvlib.location.Location(lat, lon, tz, altitude=0, name='Sao Paulo')

declination = pvlib.solarposition.declination_spencer71(dn)

print(location)
print("dn =", dn, "dias")
print("Equacao do Tempo = ", eq_time)
print("Angulo horario = ", hour_angle)
print("Declinacao = ", declination)

#Analises

solpos = solarposition.get_solarposition(time, lat, lon)
solpos = solpos.loc[solpos['apparent_elevation'] > 0, :]
ax = plt.subplot(1, 1, 1, projection='polar')
points = ax.scatter(np.radians(solpos.azimuth), solpos.apparent_zenith,
                    s=2, label=None, c=solpos.index.dayofyear)
ax.figure.colorbar(points)

for hour in np.unique(solpos.index.hour):
    subset = solpos.loc[solpos.index.hour == hour, :]
    r = subset.apparent_zenith
    pos = solpos.loc[r.idxmin(), :]
    ax.text(np.radians(pos['azimuth']), pos['apparent_zenith'], str(hour))

```

```
for date in pd.to_datetime(['2019-03-21', '2019-06-21', '2019-12-21']):
    times = pd.date_range(date, date+pd.Timedelta('24h'), freq='5min', tz=tz)

    solpos = solarposition.get_solarposition(times, lat, lon)
    solpos = solpos.loc[solpos['apparent_elevation'] > 0, :]
    label = date.strftime('%Y-%m-%d')
    ax.plot(np.radians(solpos.azimuth), solpos.apparent_zenith, label=label)

ax.figure.legend(loc='upper left')
ax.set_theta_zero_location('N')
ax.set_theta_direction(-1)
ax.set_rmax(90)
```

Bem como o diagrama solar:

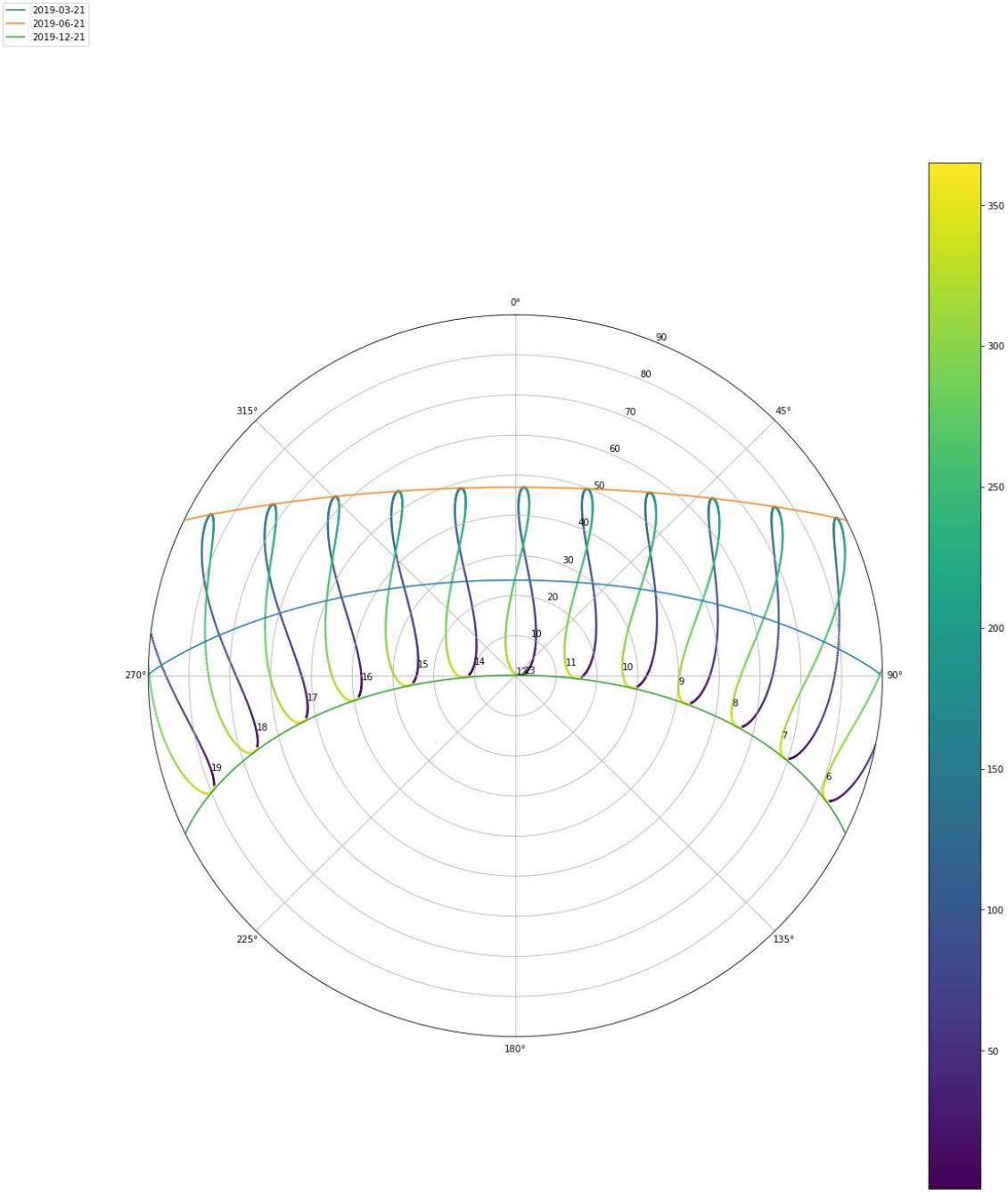

Figura 1. Diagrama solar gerado a partir da biblioteca pvlib. Fonte: autoral.

Desta maneira, com base nos procedimentos de Cronemberger et al., foi-se iniciando os cálculos referentes à criação do modelo de avaliação da eficiência. Porém, foram encontradas divergências entre os valores encontrados e os resultados esperados, de acordo com a referência. Isso ocorreu devido a utilização de dados de irradiância de dias nublados em oposição à aplicação de um modelo de céu claro para identificar os dias em que há céu encoberto. Este modelo deverá analisar todos os dias do ano, mas por enquanto avalia apenas um (16/05). O mesmo código calcula também fatores como:

- Massa de ar absoluta;
- Altitude solar;
- Ângulo de incidência;
- Declinação solar;
- Equação do Tempo;
- Horas do nascer e por do Sol;
- Ângulo horário;
- Pressão da altitude;
- Ângulos zenital e azimutal;
- Dados de irradiância solar:
    - Irradiância Global;
    - Irradiância Difusa; e
    - Irradiância Direta.

Além dessas variáveis, o código gera o gráfico de modelo de dia de céu claro de Ineichen (Figura 2), de irradiância em comparação com a altitude solar (Figura 3), de comparação do modelo de Ineichen com os dados observados (Figura 4) e de determinação de dia de céu claro (Figura 5). Segue abaixo a íntegra do código, bem como os *plots* dos gráficos.

```
1.  import matplotlib.pyplot as plt
2.
3.  import pandas as pd
4.
5.  import pvlib
6.
7.  import datetime as dt
8.
9.  import math as m
10.
11. import numpy as np
12.
13. from pvlib import clearsky, atmosphere, solarposition
14.
15. from pvlib.location import Location
```

```
16.
17.
18.
19. #Solar position
20.
21.
22.
23. lat, lon = -9.0689, -40.3197 #Petrolina
24.
25. tz = 'America/Sao_Paulo'
26.
27. data = dt.datetime.strptime('16/05/2012','%d/%m/%Y')
28.
29. dn = data.timetuple().tm_yday
30.
31. time = pd.date_range('2012-05-16 00:00:01', '2012-05-
    17 00:00:01', closed='left', freq='H', tz=tz)
32.
33.
34.
35. eq_time = pvlib.solarposition.equation_of_time_spencer71(dn)
36.
37. hour_angle = pvlib.solarposition.hour_angle(time, lon, eq_time)
38.
39. loc = pvlib.location.Location(lat, lon, tz, altitude=387, name='Petrolina')
40.
41. declination = pvlib.solarposition.declination_spencer71(dn)
42.
43. solpos = solarposition.get_solarposition(time, lat, lon)
44.
45. horasssr = pvlib.solarposition.sun_rise_set_transit_ephem(time, lat, lon)
46.
47.
48.
49. print(loc)
50.
51.
52.
53. #Clear sky (get_clearsky)
54.
55.
56.
57. petrolina = Location(lat, lon, tz, 387, 'Petrolina')
58.
59.
60.
61. #Clear sky (Ineichen)
62.
63.
64.
65. zenith = solpos['apparent_zenith']
66.
67. airmass = pvlib.atmosphere.get_relative_airmass(zenith)
68.
69. pressao = pvlib.atmosphere.alt2pres(387)
70.
71. airmass = pvlib.atmosphere.get_absolute_airmass(airmass, pressao)
72.
73. linke = pvlib.clearsky.lookup_linke_turbidity(time, lat, lon)
74.
75. dni_extra = pvlib.irradiance.get_extra_radiation(time)
76.
77. ineichen = clearsky.ineichen(zenith, airmass, linke, 387, dni_extra)
78.
79. plt.figure()
80.
```

```
ax = ineichen.plot()

ax.set_ylabel('Irradiance $W/m^2$')

ax.set_title('Ineichen Clear Sky Model')

ax.legend(loc=2)


#Irradiância x Altitude solar


altitude = pd.Series(np.linspace(-10, 90, 101))

aod700 = 0.1

precipitable_water = 1

      solis = clearsky.simplified_solis(altitude, aod700, precipitable_water, 101325, 1364)

      ax = solis.plot()

      ax.set_xlabel('Altitude (deg)')

      ax.set_ylabel('Irradiância $W/m^2$')

      ax.set_title('Irradiância x Altitude solar')

      ax.legend(loc=2)


      #Detect clear sky


      detect_petrolina = Location(-9.0689, -40.3197, altitude=387)

      detect_time = pd.date_range(start='2012-05-16 00:01:00', tz='Etc/GMT+3', periods=1440, freq='1min')

      detect = detect_petrolina.get_clearsky(detect_time)

      detect_alt = detect.resample('60T', closed='right').mean()

      beam = detect_alt['ghi'] - detect_alt['dhi']

      detect_alt['beam'] = beam

      ghi = detect['ghi']*.97


      #Plot irradiance


      fig, ax = plt.subplots()

      ghi.plot(label='input')

      detect['ghi'].plot(label='ineichen clear')

```

```
145.        ax.set_ylabel('Irradiance $W/m^2$')
146.
147.        plt.legend(loc=4)
148.
149.
150.
151.        #Plot clearsky
152.
153.
154.
155.        clear_samples = clearsky.detect_clearsky(ghi, detect['ghi'], detect.ind
    ex, 60)
156.
157.        fig, ax = plt.subplots()
158.
159.        clear_samples.astype(float).plot()
160.
161.        ax.set_ylabel('Clear (1) or Cloudy (0)')
162.
163.
164.
165.        aoi = pvlib.irradiance.aoi(30, 150, solpos['zenith'], solpos['azimuth']
    )
```

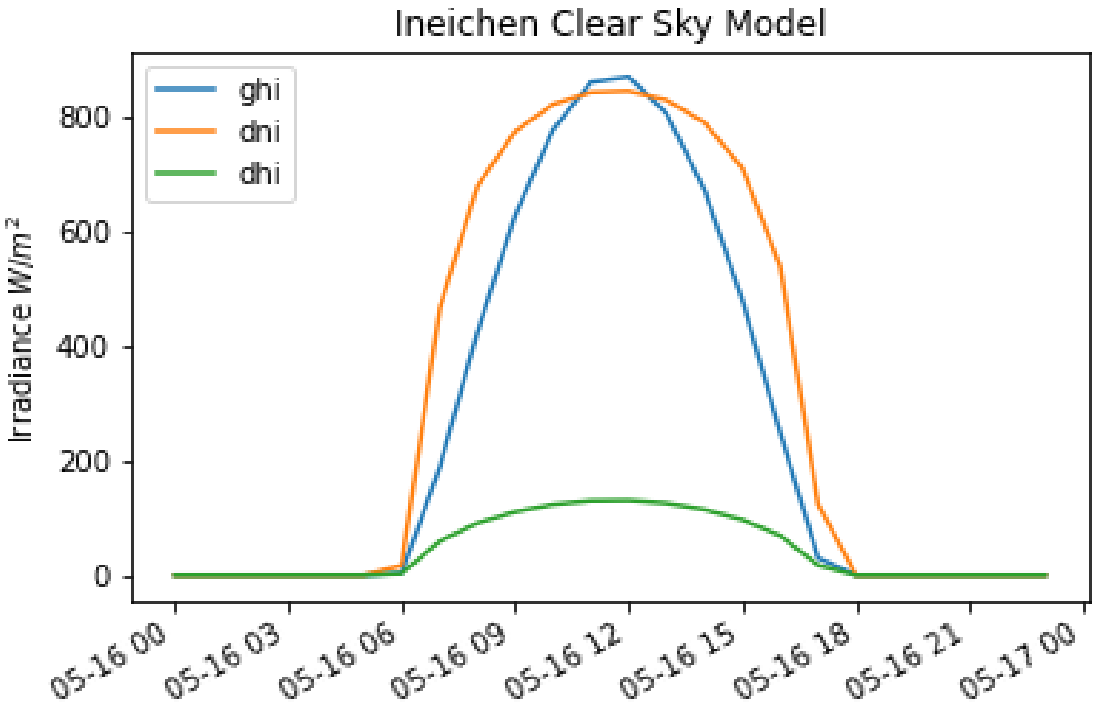

Figura 2. Gráfico do modelo de dia de céu claro de Ineichen demonstrando a variação das componentes solares (irradiância global, direta e difusa) durante o dia. Fonte: autoral.

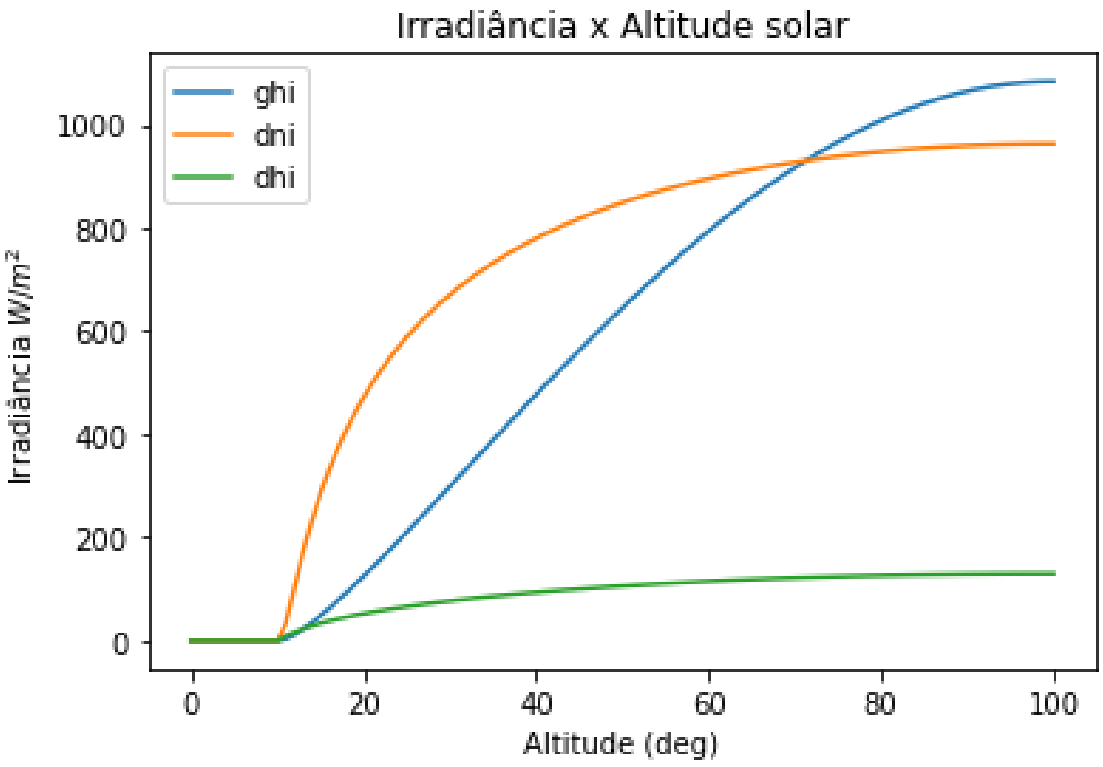

Figura 3. Gráfico comparativo de cada componente solar (irradiância global, direta e difusa) em relação à altitude solar. Fonte: autoral.

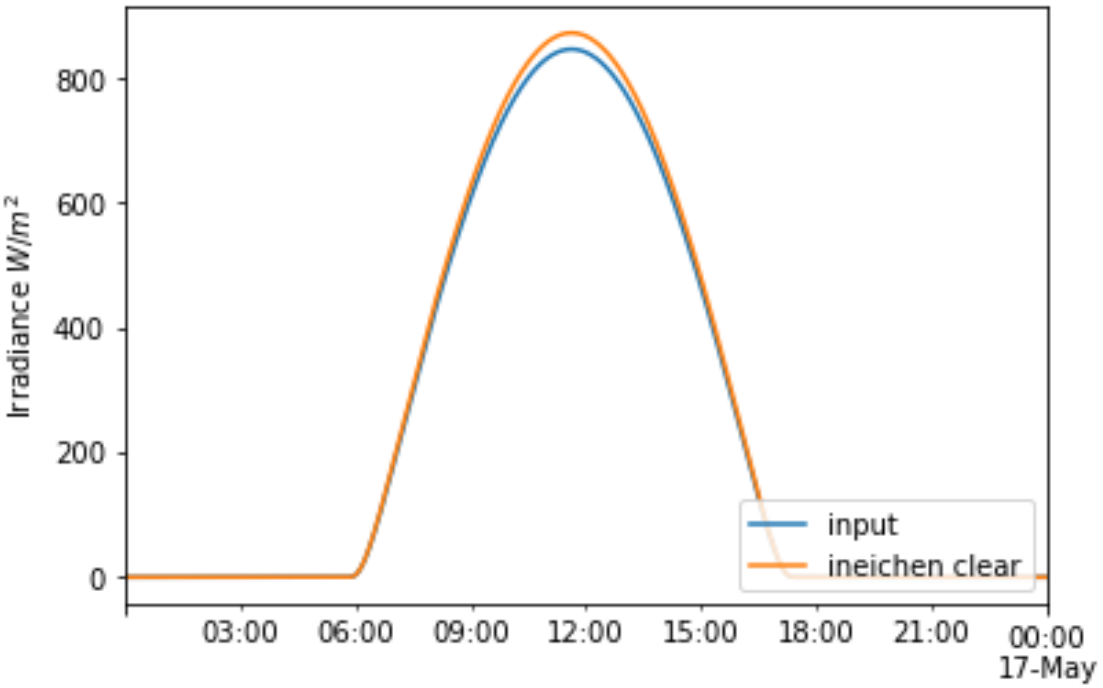

Figura 4. Gráfico comparando os dados de irradiância observados com aqueles esperados pelo modelo de dia de céu claro de Ineichen. Fonte: autoral.

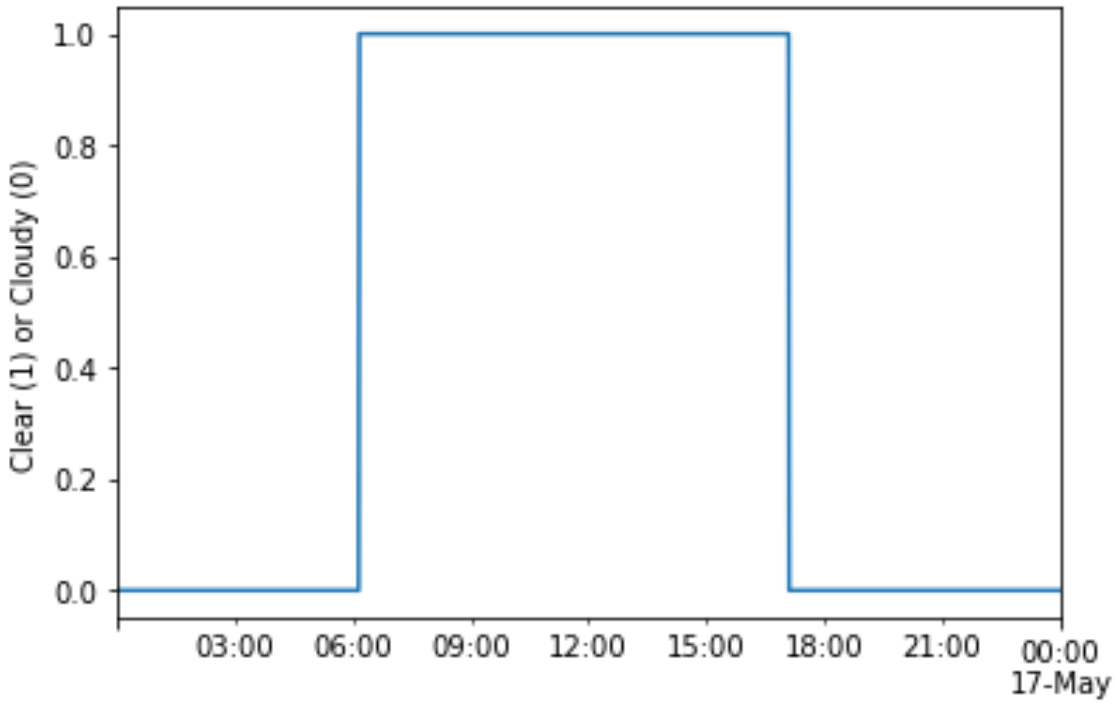

Figura 5. Gráfico do modelo de dia de céu claro. Avalia se está o céu está claro (1) ou encoberto (0) durante as horas do dia. Fonte: autoral.

Os dados obtidos com a execução do código deflagram que o céu sobre a cidade de Petrolina, no dia 16/05/2012 estava claro das 6h até aproximadamente as 17h, configurando um dia hábil para a aplicação do modelo de equações propostos por Cronemberger et al., a fim de iniciar os cálculos de perda de geração fotovoltaica em telhados reais.

REFERÊNCIAS BIBLIOGRÁFICAS

EPIA, European Photovoltaic Industry Association, **Unlocking the Sunbelt Potential of Photovoltaics.** Bruxelas, 2010.

IQBAL, Muhammad. **An Introduction to Solar Radiation.** Vancouver, 1983.

PEREIRA, Enio Bueno; MARTINS, Fernando Ramos; GONÇALVES, André Rodrigues; COSTA, Rodrigo Santos; LIMA, Francisco J. Lopes de; RUTHER, Ricardo; ABREU, Samuel Luna de; TIEPOLO, Gerson Máximo; PEREIRA, Silvia Vitorino; SOUZA, Jefferson Gonçalves. **Atlas Brasileiro de Energia Solar.** São José dos Campos, 2017.

CRONEMBERGER, Joara; CAAMAÑO-MARTÍN, Estefanía; SÁNCHEZ, Sergio Vega. **Assessing the solar irradiation potential for solar photovoltaic applications in buildings at low latitudes – Making the case for Brazil.** Madrid, 2012.