


# Laboratório 4 – MMQ para posição do sol

- 
- Efrain Marcelo
  - Fernando Lucas
  - Teophilo Vitor




# O experimento

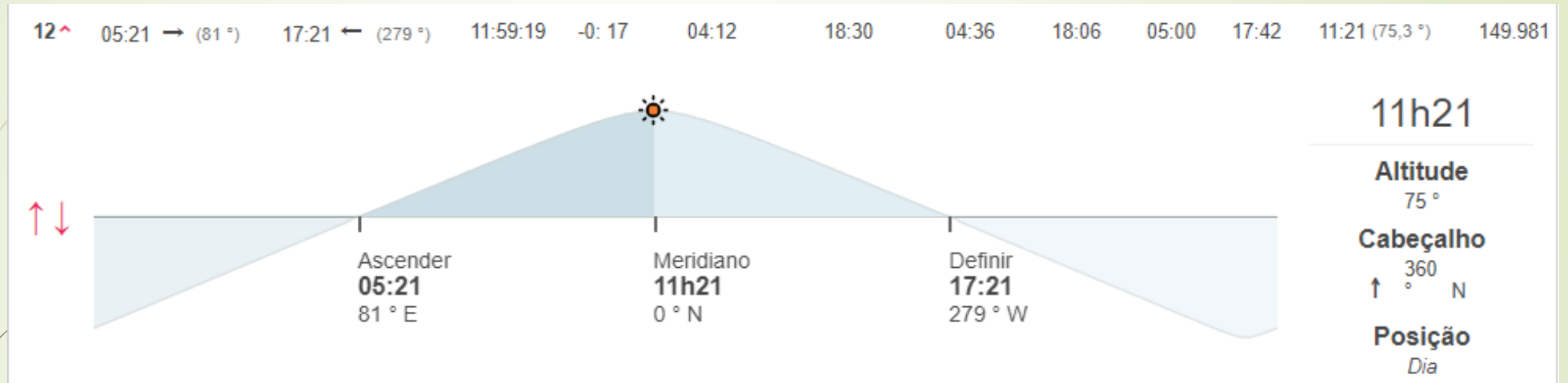
- O experimento visa obter a localização na terra em relação ao sol, para tal será utilizado do ângulo entre o sol e o referencial na terra.
- As medidas consistirão nas alturas referentes a sombra causadas pelo sol no referencial utilizado, justamente com os respectivos horários, com isso através do algoritmo em Python teremos os dados como ângulo, latitude, horário e respectivo dia referencia dos dados.

```
def anguloshoras(h):  
    angulo = np.zeros([len(h)])  
    for i in range(0, len(h)):  
        horas = int(h[i]) * 60  
        min = (h[i]- int(h[i])) * 100  
        total = horas + min  
        angulo[i] = total * (2 * np.pi / 1440) - np.pi  
    return angulo
```

```
def angulosombra(s):  
    angulo = np.arctan(126/s)  
    return angulo
```



# Relação posição solar e horário



# Relação posição solar e horário

# Tabela de dados

	Horário	Medida	Hora angular (rad)	Ângulo (rad)
1	9h	97 cm	-0.61522856	0.91471726
2	9h30	75 cm	-0.48432887	1.03388558
3	10h	57 cm	-0.35342917	1.14596416
4	10h30	40 cm	-0.22252948	1.26339885
5	11h	28 cm	-0.09162979	1.35212738
6	14h	105 cm	0.69376838	0.87605805
7	14h30	142 cm	0.82466807	0.73997488
8	15h	172 cm	0.95556777	0.63224455
9	15h30	227 cm	1.08646746	0.50672439
10	16h	270 cm	1.21736715	0.43662716

# Função do ângulo em relação a altura do sol

$$a = \arcsin(\cos(h) * \cos(\delta) * \cos(\varphi) + \sin(\delta) * \sin(\varphi))$$

Chamando  $a_0 = \sin(\delta) * \sin(\varphi)$  e  $a_1 = \cos(\delta) * \cos(\varphi)$  teremos:

$$a = \arcsin(a_0 + a_1 * \cos(h))$$

Aplicando a função *sin* dos dois lados temos:

$$\sin(a) = a_0 + a_1 * \cos(h)$$

Encontramos nossa função linear  $Y = a_0 + a_1 * X$

Lembrando que o nosso Y vai ser  $\sin(a)$  então para encontrar a temos que fazer  $\arcsin(Y)$

Para determinar o  $a_0$  e  $a_1$  vamos realizar o código de determinação dos coeficientes pela linearização:

Temos que  $g_0 = 1$  e  $g_1 = \cos(h)$



# Linearização (MMQ)

```
sg02 = sum(x)
sg0g1 = sum(np.cos(x))
sg12 = sum(np.cos(x)**2)
syg0 = sum(np.cos(y))
syg1 = sum(np.cos(y)*np.cos(x))
A = np.zeros([2,2])
A[0,0] = sg02
A[0,1] = sg0g1
A[1,0] = sg0g1
A[1,1] = sg12
b = np.zeros([2,1])
b[0,0] = syg0
b[1,0] = syg1
print(A)
coeficiente = np.linalg.inv(A).dot(b)
print(coeficiente)
a0 = coeficiente[0,0]
a1 = coeficiente[1,0]
print("A0 = " , a0 , " A1 = " , a1)
#Plotar grafico dos dados coletados
k = np.linspace(-np.pi,np.pi,100) #cria o vetor x de 0.1 a 6 com 100 elementos
plt.scatter(x,y)
plt.show()
# Criar nosso novo vetor y1 referente aos resultados com a nossa nova função encontrada
y1 = np.arcsin(a0 + a1 * np.cos(x))
y2 = np.arcsin(a0 + a1 * np.cos(k))
#Plotar grafico com a nossa função
plt.plot(k,y2)
plt.scatter(x,y1)
plt.show()
```

A0 = -0.09169179837752016 A1 = 0.7431412642151473

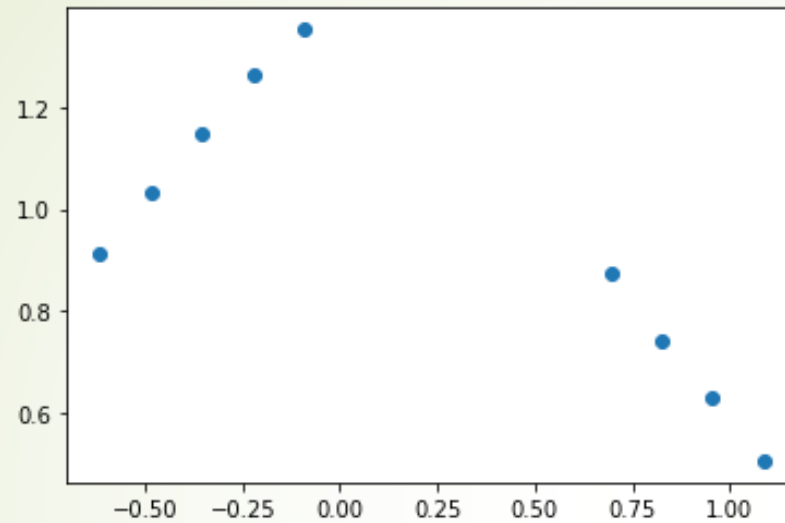


Gráfico  $a \times h$  (com dados coletados no experimento)

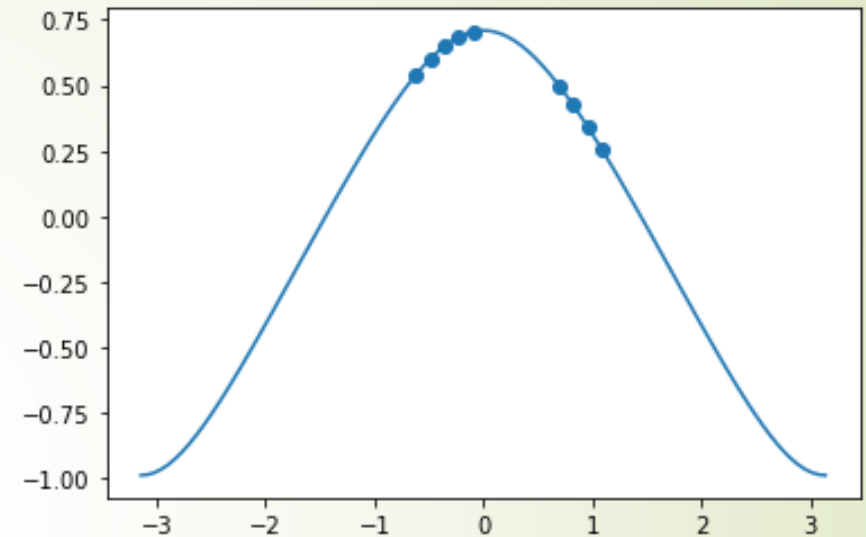


Gráfico  $a \times h$  (com  $a = \arcsin(a_0 + a_1 * \cos(h))$ )

# Plotagem dos gráficos



```

#encontrando a raiz da nossa função nas horas
def f(x):
    global a0
    global a1
    return a0 + a1 * np.cos(x)

#Selecionar o intervalo que contem a raiz
# Os valores de a e b ja assumem que f(a) * f(b) < 0
a = - np.pi
b = 0
p = 6 # Casa de precisão ex: 10^-p
n_intera = 6 # Caso de o numero de interações para fzr
erro = 1
inte = 0
c = a
while erro >= ma.pow(10,-p) and n_intera > inte :
    inte += 1
    x0 = c
    c = a - (f(a) * (b-a)/(f(b) - f(a)))
    if f(a) * f(c) < 0:
        b = c
    else :
        a = c
    erro = abs((c - x0)/c)
print("Raiz: ", c)
raizh , raizmin = ajust(c)
print("Horas da Raiz: ", raizh , ":", raizmin)

```

```

def ajust(j):
    total = ((j + np.pi)/((2 * np.pi / 1440))) - 39
    horas = int(total/60)
    min = int(total % 60)
    return horas , min

```

Nascer do sol em horas angulares: -1.4470970268493142  
 Nascer do Sol em horas normais: 5 : 49

# Horário do nascer do sol através de um problema de raízes

# Angulo de inclinação em relação ao sol

$$\cos(\phi)\sqrt{-a_0^2/\sin(\phi)^2 + 1} - a_1 = 0$$

#Encontrando a raiz da nossa função da latitude

```
def lati(x):  
    global a0  
    global a1  
    return np.cos(x) * np.sqrt(1 - ((ma.pow(a0,2)) / (ma.pow(np.sin(x),2)))) - a1  
  
a = 1  
b = 2  
p = 6 # Casa de precisão ex: 10^-p  
n_intera = 6 # Caso de o numero de interações para fzf  
erro = 1  
inte = 0  
c = a  
while erro >= ma.pow(10,-6) and n_intera >= inte:  
    inte += 1  
    c = a - lati(a)*((a-b)/(lati(a)-lati(b)))  
    b = a  
    a = c  
    erro = abs((a - b)/a)  
print("Raiz Latitude : " , c)
```

**Raiz Latitude: 0.722134914960197**

# Dia do ano que foi realizado o experimento

```
fi = c

ndias = 101
delta = -23.44 * np.cos((360/366)*(ndias + 10) * (np.pi/180))
print("Delta: " , delta)

#Descobrimo o numero de dias

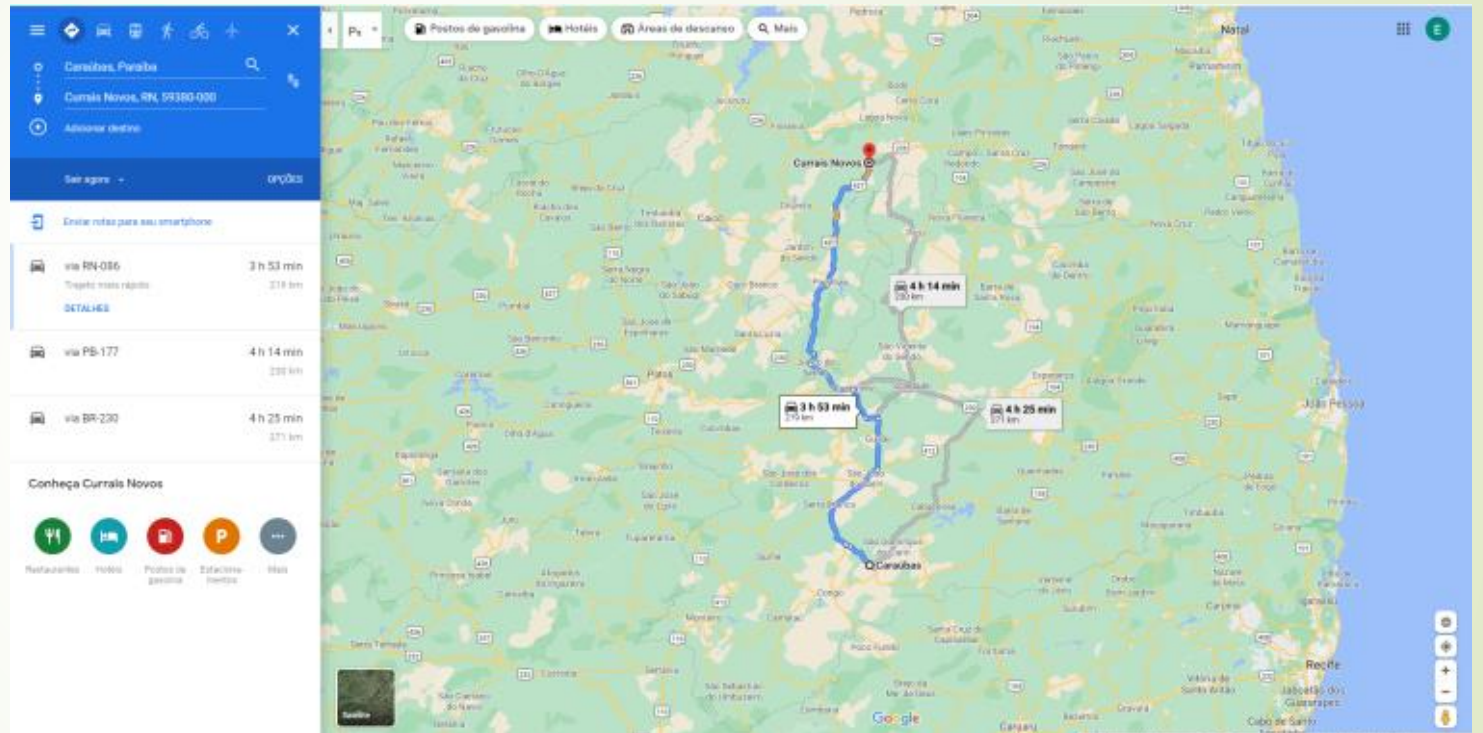
def teste(ndias):
    return (-23.44 * np.cos((360/366)*(ndias + 10) * (np.pi/180))) - fi

a = 70
b = 200
p = 6 # Casa de precisão ex: 10^-p
n_intera = 40 # Caso de o numero de interações para fzr
erro = 1
inte = 0
c = a
while erro >= ma.pow(10,-p) and n_intera > inte :
    inte += 1
    x0 = c
    c = a - (teste(a) * (b-a)/(teste(b) - teste(a)))
    if teste(a) * teste(c) < 0:
        b = c
    else :
        a = c
    erro = abs((c - x0)/c)
print("Raiz dias: " , int(c))
```

**Raiz dias: 83**

**Delta: 7.701033431989966**

# Local Real × Local Experimental





OBRIGADO  
PELA  
ATENÇÃO