F625 Lista 1

Bruno Oliveira, RA: 165304

Setembro 2018

1 Nota

Além de estarem inclusos em anexo, os *prints* dos Programas se encontram no final deste documento, com a excessão do exercício 2.10 pois este era muito grande para caber em uma única imagem.

2 Exercício 2.2

2.1 item a)

Igualando a força gravitacional com a força centripeta, temos:

$$\frac{GMm}{D^2} = \frac{mv^2}{D} \tag{1}$$

onde G é a constande gravitacional, M é a massa da Terra, m a massa do objeto e D é a distância entre o centro da Terra e do objeto. Utilizando que $v = \frac{2\pi}{T}D$ com T sendo o período da órbita obtemos:

$$D^3 = \frac{GMT^2}{4\pi^2} \tag{2}$$

Substituindo $D = h + R$ com h a altura da orbita a partir da superfície e R o raio da Terra, obtemos a equação desejada.

2.2 item b)

Código em anexo, arquivo Exercise2.2.py

2.3 item c)

Os valores de h obtidos pelo programa Exercise2.2.py são mostrados na Tabela 2.3

Vemos que para T = 45 minutos, a trajetória esta abaixo da superfície da Terra, logo esta não é uma trajetória possível.

Tabela 1: Tabela com os valores obtidos na saída do programa Exercise2.2.py

T	h (metros)
24 horas	35855910.17617497
90 minutos	279321.62537285965
45 minutos	-2181559.8978108233

2.4 item d)

Para T = 23.93 horas, a altidude é de 35773762.329895645 metros, o que implica em uma diferença de 82147.8462793 metros.

3 Exercício 2.4

O programa que se encontra em anexo (Exercise2.4.py) faz a conta para ambos os referenciais

3.1 item a)

No referencial do observador na Terra, o tempo encontrado pelo programa foi de $10.10 anos$

3.2 item b)

No referencial do observador na Nave, o tempo encontrado pelo programa foi de $1.425 anos$

4 Exercício 2.5

Através do código encontrado em anexo (Exercise 2.5.py) os valores encontrados foram Probabilidade de transmissão $T = 0.7301261363877617$ e Probabilidade de reflexão $R = 0.2698738636122384$. considerando um partícula de massa $m = 9.11 * 10^{-31} kg$ com energia $E = 10eV$ e um potêncial $P = 9eV$.

5 Exercício 2.10

Código em anexo Exercise2.10.py

5.1 item a)

A energia de ligação obtida pelo programa Exercise2.10.py usando $A = 58$ e $Z = 28$ foi de $B = 497.5620206224374 MeV$

5.2 item b)

A energia de ligação por nucleon obtida pelo programa Exercise2.10.py usando $A = 58$ e $Z = 28$ foi de $B/A = 8.578655527973059 MeV$

5.3 item c)

O valor de A encontrado para $Z = 28$ foi de $A = 58$ e a energia de ligação por nucleon foi de $B/A = 8.516131151747729 MeV$

5.4 item d)

A maior energia de ligação por nucleon ocorreu quando $Z = 24$ com um valor de $B/A = 8.532622751365931 MeV$, vemos que para o niquel ($Z = 28$), a energia de ligação por nucleon foi de $B/A = 8.516131151747729 MeV$.

6 Exercício 3.1

O código referente a este exercício se encontra em anexo (Exercise3.1.py).

6.1 item a)

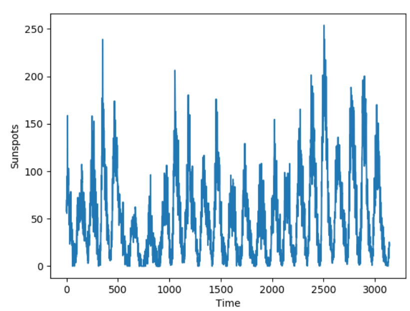

Figura 1: item 3.1-a

6.2 item b)

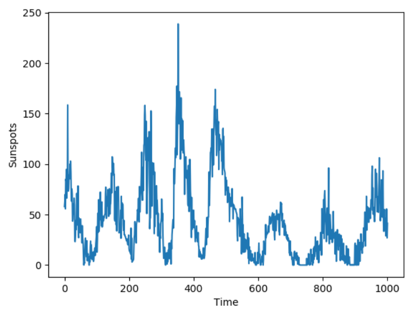

Figura 2: item 3.1-b

6.3 item c)

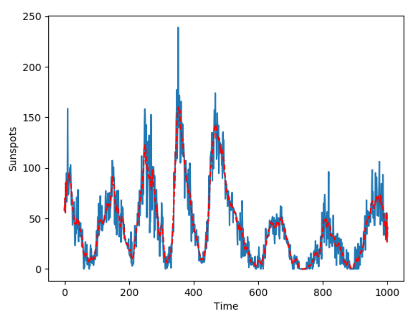

Figura 3: item 3.1-c

Para obter estas figuras basta rodar o programa Exercise3.1.py com o arquivo sunspots.txt (também em anexo) na mesma pasta.

7 Exercício 3.5

O código referente a este exercício se encontra em anexo (Exercise3.5.py).

7.1 item a)

A Figura 4 mostra a saída referente ao item a), no entanto, o código Exercise3.5.py irá gerar diretamente a animação pedida no item b)

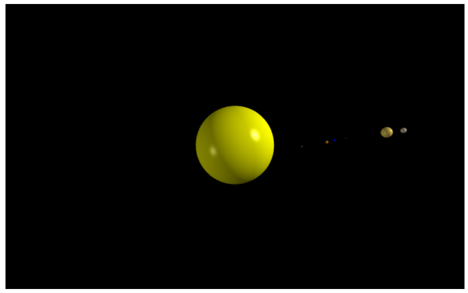

Figura 4: imagem do esquema de sistema solar pedido no item a) da questão 3.5, nesta o raio dos planetas foi aumentado em por um fator de 250 e o raio solar foi aumentado por um fator de 50

7.2 item b)

Para visualizar a animação é necessário executar o arquivo Exercise3.5.py em uma máquina com os módulo vpython instalado.

8 Exercício 3.8

O código referente a este exercício se encontra em anexo (Exercise3.8.py).

8.1 item a)

código contido no arquivo Exercise3.8.py.

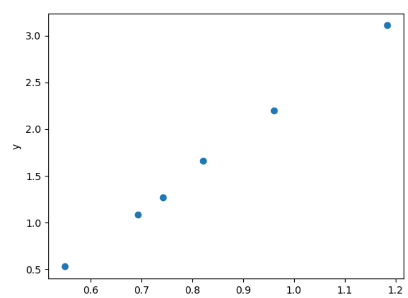

Figura 5: Gráfico obtido para o item a da questão 3.8

8.2 item b)

De acordo com a saída do Programa Exercise3.8.py, foi obtido o coeficiente angular $m = 4.09 * 10^{-15}$ e o coeficiente linear $c = -1.73$.

8.3 item c)

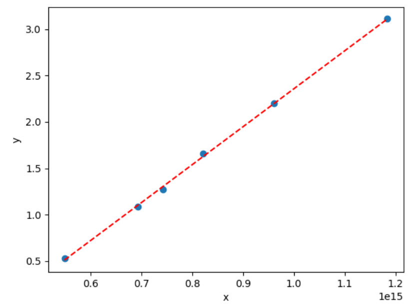

Figura 6: Gráfico obtido para o item c da questão 3.8

8.4 item d)

A partir da equação $V = \frac{h}{e}\nu - \phi$ vemos que o coeficiente angular $m = \frac{h}{e}$ de forma que $h = m * e$. De acordo com o resultado do programa Exercise3.8.py, a constante de Planck é dada por $h = 6.55 * 10^{-34} Js$, valor bastante próximo do valor da literatura $h = 6.62 * 10^{-34} Js$

9 Exercício 4.1

O código referente a este exercício se encontra em anexo (Exercise4.1.py).

9.1 Resolução do exercío 4.1

Ao tentar calcular o valor de 200! utilizando variáveis do tipo *int* ou váriaveis do tipo $float$, foram observadas duas saídas diferentes, observadas na figura 7.

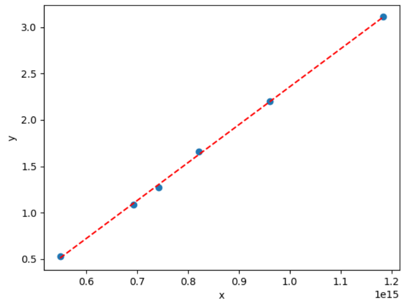

Figura 7: Saída obtida do programa Exercise.4.1.py

Isto se relaciona ao fato de que o *Python* não possui limite de digitos para cálculos com variáveis do tipo *int*, mas possui limite para calculo de variáveis do tipo *float*, e quando este limite é ultrapassado, ele informa o valor da conta como *inf*, ou seja, o programa que este valor é "infinito".

10 Exercício 4.4

O código referente a este exercício se encontra em anexo (Exercise4.4.py). A saída do programa se encontra na Figura 8

```
The value of the integral for N = 100 is  1.5691342555492505
The analitical value of the integral is  1.5707963267948966
The value of the integral after ~1 sec runtime is  1.5707963258896733
Tempo de execução  0.9025533199310303 segundos
>>> |
```

Figura 8: Saída obtida do programa Exercise.4.4.py

10.1 item a)

Para $N = 100$, o valor obtido na saída do Programa Exercise4.4.py para a integral foi de $I = 1.5691342555492505$, enquanto o valor analítico é $I = \frac{\pi}{2} \approx 1.5707963267948966$. Vemos que para $N = 100$, temos uma concordância até a primeira casa decimal.

10.2 item b)

Considerando o tempo de execução como ≈ 1 segundo, foi possível obter o valor de $I = 1.5707963258896733$ que concorda com o valor analítico até a oitava casa decimal.

11 Códigos (Exceto 2.10)

```python
from math import pi

#Exercício 2.2

T = float(input("Digite o período (somente o número)\n"))

G = 6.67*10**-11
M = 5.97*10**24
R = 6371*10**3

unit = input("insira a unidade (ie, segundos, minutos, horas...)\n")

if(unit == 'segundo' or unit == 'segundos'):
    T = T
    h = ((G * M * T ** 2) / (4 * pi ** 2)) ** (1 / 3) - R
elif(unit == 'minuto' or unit == 'minutos'):
    T *= 60
    h = ((G * M * T ** 2) / (4 * pi ** 2)) ** (1 / 3) - R
elif(unit == 'horas' or unit == 'hora'):
    T *= 3600
    h = ((G * M * T ** 2) / (4 * pi ** 2)) ** (1 / 3) - R
elif(unit == 'dias' or unit == 'dia'):
    T *= 3600*24
    h = ((G * M * T ** 2) / (4 * pi ** 2)) ** (1 / 3) - R
elif(unit == 'anos' or unit == 'ano'):
    T *= 3600*24*365
    h = ((G * M * T ** 2) / (4 * pi ** 2)) ** (1 / 3) - R
else:
    print('unidade não suportada: ',unit)
    h = 0

print("A altitude da órbita é",h,"metros!")
```

Figura 9: Programa Exercise.2.2.py

```python
import math

x = float(input('Entre com a distância em anos-luz:\n'))#x é distancia até outro planeta x anos luz de distância
v = float(input('Entre com a velocidade da espaçonave (como fração da velocidade da luz \'c\'):\n'))

print('no referencial da Terra, o tempo de chegada é:  ',x/v ,'anos')
print('no referencial do passageiro na espaçonave, o tempo de chegada é:  ',x/(v*(1/(math.sqrt(1 - v**2)))) ,'anos')
```

Figura 10: Programa Exercise.2.4.py

```python
from math import sqrt,pi

massa = float(input('Insira a massa da partícula (em kg)\n'))
energia = float(input('Insira a energia da partícula (em eV)\n'))
potencial = float(input('Insira o valor do potêncial (em eV)\n'))


c = 299792458 #(m/s)
h_cortado = 6.62607004*10**(-34) #(SI)

energia *= 1.60218*10**(-19) #conversão para SI
potencial *= 1.60218*10**(-19) #conversão para SI

DeltaE = energia-potencial

k_1 = sqrt((2*massa)*energia)/h_cortado
k_2 = sqrt(2*massa*(DeltaE))/h_cortado

T = 4*k_1*k_2/((k_1 + k_2)**2)
R = ((k_1 - k_2)**2)/((k_1 + k_2)**2)

print("Probabilidade de transmissão: ",T)
print("Probabilidade de reflexão: ",R)
```

Figura 11: Programa Exercise.2.5.py

```
from numpy import loadtxt
import pylab as pl

#item (a)
data = loadtxt("sunspots.txt",float)# lê o txt
x = data[:,0]
y = data[:,1]
pl.plot(x,y)
pl.xlabel("Time")
pl.ylabel("Sunspots")
pl.show()
#item (b)
x = data[:1001,0]
y = data[:1001,1]
pl.plot(x,y)
pl.xlabel("Time")
pl.ylabel("Sunspots")
pl.show()
#item (c)
r = 5
ave = [0]*len(y)

for i in range(0,len(y)):
    ave[i] = y[i]

for k in range(r,len(y)-r):

    avesum = 0

    for m in range(k-r,k+r):
        avesum = avesum + y[m]

    ave[k] = (1/(2*r))*avesum

pl.plot(x,y)
pl.plot(x,ave,"r--")
pl.xlabel("Time")
pl.ylabel("Sunspots")
pl.show()
```

Figura 12: Programa Exercise.3.1.py

```
from vpython import sphere,vector,color,display,rate
from numpy import empty,arange
from math import cos,sin,pi

#d = display(background=color.white)

s = empty(7,sphere)

R = [695500,2440,6052,6371,3386,69173,57316]
Dist = [0,57.9,108.2,149.6,227.9,778.5,1433.4]
T = [0,88,224.7,365.3,687,4331.6,10759.2]

i=0

s[i]=sphere(pos=vector(Dist[i]*10**6,0,0),radius = 50*R[i])

for i in range(6):
    s[i+1]=sphere(pos=vector(Dist[i+1]*10**6,0,0),radius = 250*R[i+1])

s[0].color = color.yellow
s[1].color = vector(165,113,78)/255
s[2].color = color.orange
s[3].color = color.blue
s[4].color = color.red
s[5].color = vector(250,218,94)/255
s[6].color = vector(213,196,161)/255

for theta in arange(0,1000*pi,(1/100)*pi):
    rate(100)
    for i in range(6):
        x = Dist[i+1]*cos(2*theta/T[i+1])*10**6
        z = Dist[i+1]*sin(2*theta/T[i+1])*10**6
        s[i+1].pos = vector(x,0,z)
```

Figura 13: Programa Exercise.3.5.py

```
from numpy import loadtxt
import pylab as pl
from decimal import Decimal

#item a)
data = loadtxt("millikan.txt",float)# lê o txt
x = data[:,0] #isto será a frequência em hertz
y = data[:,1] #isto será a voltagem em volts
pl.plot(x,y,"o")
pl.xlabel("x")
pl.ylabel("y")
pl.show()
#item b)
sumx = 0
sumy = 0
sumxx = 0
sumxy = 0
for i in range(0,len(x)):
    sumx=x[i]+sumx
    sumy=y[i]+sumy
    sumxx=x[i]*x[i]+sumxx
    sumxy=x[i]*y[i]+sumxy

Ex = (1/len(x))*sumx
Ey = (1/len(x))*sumy
Exx = (1/len(x))*sumxx
Exy = (1/len(x))*sumxy
m = (Exy - Ex*Ey)/(Exx-Ex*Ex)
c = (Exx*Ey-Ex*Exy)/(Exx-Ex*Ex)

print("The slope \"m\" of the best- fit line is",'%.2E' % Decimal(m),"and its intercept \"c\" is",'%.2E' % Decimal(c))
#item c)
v = [0]*len(x)
for i in range(0,len(x)):
    v[i]=m*x[i]+c
pl.plot(x,y,"o")
pl.plot(x,v,'r--')
pl.xlabel("x")
pl.ylabel("y")
pl.show()
#item d)
e = 1.602*10**-19 #carga do eletron em Coulomb
h = m*e #segue da equação dada
print(" The value of Planck's constant is then h =",'%.2E' % Decimal(h),"Js")
```

Figura 14: Programa Exercise.3.8.py

```python
def mod(x):

    if(x<0):
        x*=-1
    return x

def fact(n):

    if(n==0):
        return 1
    else:
        fact = n

        for i in range(1,int(n)):
            fact = fact*(n-i)
        return fact

#item a)

a = mod(int(input("insert an integer \"n\" \n")))

print("n! is equal to",fact(a))

#item b)

b = mod(float(input("insert a real number \"n\" \n")))

print("n! is equal to",fact(b))
```

Figura 15: Programa Exercise.4.1.py

```python
from math import sqrt,pi
from time import time


def Integral(N):

    h = 2/N

    sum = 0
    for k in range(1,N+1):
        sum = sum + h*(sqrt(1 - (-1+h*k)**2))

    return sum

#item a)

I100 = Integral(100)

print("The value of the integral for N = 100 is ",I100)
print("The analitical value of the integral is ",pi/2)

#item b)

t0=time()
I1sec=Integral((10**6)+(5*10**5))
t1=time()

print("The value of the integral after ~1 sec runtime is ",I1sec)
print("Tempo de execução ",t1-t0,"segundos")
```

Figura 16: Programa Exercise.4.4.py