Manual – reflexaopy

Autores:

Douglas Parodes Benites	11932355
Henrique Silva Julio	9389779
Mateus Vieira Nunes	11932293
Ricardo Moises Leocadio da Silva	11801435

Finalidade:

O *reflexaopy* é um projeto que busca auxiliar pessoas quanto ao tratamento de raios de luz como vetores em situações que eles refletiriam. O projeto atualmente comporta até dois espelhos e um vetor, mas pode ser expandido facilmente se o usuário desejar e entender o código.

Acesso:

O progrma está disponível em:
https://github.com/Ricardo-MLS/reflexaopy

Como funciona:

Para entender como o programa funciona é importante entender seu código. A primeira coisa a se ver são as bibliotecas que são usadas no programa:

```
import math
import numpy as np
import matplotlib.pyplot as plt
```

Como pode ser visto na imagem, as bibliotecas são "*math*", "*numpy*", que é usada como "*np*", e "*matplotlib.pyplot*", que é usada como "*plt*". A biblioteca *math* será importante pois é com ela que se obtém os valores dos ângulos em radiano, por exemplo. A biblioteca *numpy* é importantíssima, pois ela é usada para definir *arrays*, que são as matrizes a serem usadas, e fazer cálculos com eles. Por fim, a biblioteca *matplotlib* é aquela que serve para gerar os gráficos ilustrando os vetores.

Na sequência, o programa solicita ao usuário que ele insira o valor da angulação do primeiro espelho que será usado posteriormente.

```
degrees1= float(input('digite o angulo de inclinação do espelho 1 em graus:'))
```

Então, um *loop* do tipo "*if*" é usado para garantir que o ângulo será menor do que 90º, ou seja, um ângulo agudo.

```
if degrees1 < 90 :
    degrees1 = degrees1
else:
    degrees1 = 180 - degrees1
```

Depois disso, a função "*radians*" da biblioteca *math* é usada para converter o valor recebido de graus para radianos.

```
x=math.radians(degrees1)
```

Aí, com esses dados, é criada a matriz de reflexão relativa a primeira transformação.

```
A= np.array([[math.cos(2*x), math.sin(2*x)],
             [math.sin(2*x), -1*math.cos(2*x)]])
```

Feito tudo isso, o processo se repete mais uma vez para que sejam registrados os dados do espelho 2.

```
degrees2= float(input('digite o angulo de inclinação do espelho 2:'))
```

```
if degrees2 > 90 :
    degrees2 = degrees2
else:
    degrees2 = 180 - degrees2
```

```
y= math.radians(degrees2)
```

```
B= np.array([[math.cos(2*y), math.sin(2*y)],
             [math.sin(2*y), -1*math.cos(2*y)]])
```

Quando ambas as matrizes dos espelhos são construídas, é criada outra contendo o resultado da multiplicação dessas duas matrizes para determinar a transformação definitiva do vetor.

```
C= B@A
```

Tendo a matriz da transformação definitiva, o programa então pede ao usuário que ele insira os dados do vetor que será refletido.

```
a= float(input('componente da direção em x(+) do vetor do raio de luz:'))
```

Para garantir que a representação será fidedigna com os dados recebidos, sobre os dados inseridos é imposta uma restrição usando um *loop* do tipo *if*. Essa restrição serve para garantir que o componente x da direção será positivo.

```
if a > 0 :
    a = a
else:
    a = -a
```

Depois, as mesmas coisas são feitas para o componente y da direção do vetor, só que a restrição exige que ele seja negativo.

```
b= float(input('componente da direção em y(-) do vetor do raio de luz:'))
```

```
if b < 0 :
    b = b
else:
    b = -b
```

Nesse momento, é criado um *array* que será o vetor. Esse *array* contém os dados inseridos pelo usuário.

```
D= np.array([[a],[b]])
```

Para efetuar a transformação, esse *array* é multiplicado com aquele que possui a multiplicação das matrizes que contém os dados de ambos os espelhos.

```
E= C@D
```

Aí, para obter a direção do último vetor refletido, duas variáveis são criadas e recebem o valor contido nas posições do *array* que estão destacadas.

```
h= E[0]
f= E[-1]
```

Para finalizar mais esta etapa do programa, uma variável é declarada para receber uma *string* que será exibida para o usuário. Essa *string* contém os dados dessas duas variáveis recém-criadas.

```
s= f'a direção do ultimo vetor refletido é: ( {h} , {f} )'
print(s)
```

Para conseguir a direção do segundo vetor, declara-se duas variáveis, c e d, que receberam esses valores com base nos cálculos usados.

```
c= a*(math.cos(2*x))+ b*(math.sin(2*x))
d= a*(math.sin(2*x)) - b*(math.cos(2*x))
```

Depois, obtém-se a norma do segundo vetor, a tangente da angulação da segunda reta e o coeficiente angular da reta em direção ao segundo vetor.

```
mod= (c*c + d*d)**0.5
m1= math.tan(y)
m2= d/c
```

Com os dois últimos dados, determina-se os pontos de intersecção com a reta em direção ao segundo vetor e a segunda reta em x e y. Isso é importante para saber onde o vetor se colocará na ilustração.

```
p1= (m1*(-70))/(m2-m1)
p2= (m1*m2*(-70))/ (m2-m1)
```

Aí, é determinado a distância entre a origem e o ponto de intersecção. Isso será usado para que os componentes do segundo vetor estejam encostados na segunda reta.

```
dist= (p1*p1 + p2*p2)**0.5
c= c* dist
d= d* dist
```

Nesse momento, são criados valores que representam vetores unitários com os dados do segundo vetor. Esses valores serão usados na função que se segue para gerar os vetores.

```
c= c/ mod
d= d/ mod
```

Nessa parte, é exibido a direção do segundo vetor refletido.

```
n= f'a direção do segundo vetor refletido é: ( {c} , {d} )'
print(n)
```

Agora, a função que gera o gráfico que contém os vetores e os espelhos. Ela é criada com três argumentos, cada um deles representando um vetor.

```
def draw(vec1, vec2, vec3):
```

Então, é criado um *array* que contém os dados que serão usados para criar os vetores e esses dados são divididos entre quatro variáveis que serão argumentos da função "*quiver*". Essa função, que é própria da biblioteca *matplotlib*, serve para criar setas com determinados preceitos. Documentação mais extensiva sobre ela pode ser encontrada aqui:

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.quiver.html

Depois, o gráfico é determinado com eixo x e y, que possuem limites de -400 e +300 para o x e +400 para o y, com tarjas de "Eixo X" e "Eixo Y", respectivamente.

```
array = np.array([[-6*a, -6*b, vec1[0], vec1[1]],
                  [0, 0, vec2[0], vec2[1]],
                  [vec2[0], vec2[1], vec3[0], vec3[1]]])

X, Y, U, V = zip(*array)
plt.figure()
plt.ylabel('Eixo Y')
plt.xlabel('Eixo X')
ax = plt.gca()
ax.quiver(X, Y, U, V,color='b', angles='xy', scale_units='xy',scale=1)
ax.set_xlim([-400, 300])
ax.set_ylim([-400, 400])
```

Na sequência as retas são geradas, a legenda é colocada e as funções que geram o gráfico são chamadas.

```
x1= np.arange(-400,300,1)
plt.plot(x1,x1*(math.tan(x)), label= 'Espelho 1')
x2= np.arange(-400,300,1)
plt.plot(x2, m1*(x2-70), label= 'Espelho 2')
plt.legend(bbox_to_anchor=(0.,1.02, 1., .102), loc= 'lower left',
           ncol= 2, mode = 'expand', borderaxespad=0.)
plt.draw()
plt.show()
```

Para finalizar o programa, a função criada para ilustrar a transformação sofrida pelos raios de luz representados por vetores que passaram por reflexão é chamada com os valores para os argumentos determinados e destacados.

```
draw([6*a,6*b], [c,d],[19*h,19*f])
```

Exemplo:

Caso os dados inseridos sejam, respectivamente, 30, 120, 10 e -10 os dados exibidos e os gráficos gerados serão os seguintes:

```
a direção do ultimo vetor refletido
é: ( [-10.] , [10.] )
a direção do segundo vetor
refletido é: ( -60.62177826491071 ,
226.24355652982155 )
```