AULA 3

Estruturas de controlo e Organização do Código.

Determinação da constante de Planck por mínimos quadrados.

Modularização do código PYTHON

O código deve ser **legível**, **comentado** e **reutilizável**.

Existem diversas estruturas básicas que ajudam:

Funções (internas ao script, ou importadas)

Estruturas de controlo

```
if ; elif; else
for
while
```

funções

```
def func(x1,x2,x3): #input
      xsum=x1+x2+x3
      xx=x1*x2*x3
      return xsum,xx #output

Soma,SS=func(1,2,4)
print(Soma,SS)
>>7 8
```

Estrutura de execução condicional `if`

```
if x==7:
      s=7
elif x>0:
      s=math.sqrt(x)
elif x!=-10:
      s=math.sqrt(abs(x))
…
else:
      s=0
```

Ciclo com contador (iterador) for

```
X=[0,1,2,3,4,5] #len(X)==6
S=0
for k in range(len(X)):
      S=S+X[k]
#range(len(X))==[0,1,2,3,4,5]

SS=0
for x in X:
      SS=SS+x
```

Ciclo com condição `while`

```
X=[0,1,2,3,4,5]
k=0
S=0
while k<len(X): #len(X)==6
      S=S+X[k]
      k=k+1
```

Problema

Determinação experimental da constante de Planck

$$E = h\nu$$

Experiência: Calcular o potencial de paragem de uma célula fotoeléctrica em função da frequência da radiação incidente:

Energia do electrão = Energia do fotão – Energia de arranque

$$eV_p = h\nu - W$$

2 incógnitas (*h*,*W*). Precisamos de 2 equações (2 medidas de ν e V_p). Mas é conveniente ter muitas medidas para controlar o erro experimental.

Dados experimentais efeito fotoelétrico

Comprimento de onda (nm)	Frequência (10^{12}Hz)	V_P (V)
400	749	1
414	724	0.99
429	699	0.89
444	674	0.79
462	649	0.68
480	624	0.57
500	599	0.47
522	574	0.37
545	549	0.28
571	524	0.17
600	499	0.07

$$eV_p = h\nu - W$$

Discussão ($eV_p = h\nu - W$) $y = ax + b$

Com 1 medida o problema é subdeterminado: não é possível calcular *h*

Com 2 medidas o problema tem solução única. No entanto, se as medidas não forem exactas os valores obtidos têm erro desconhecido

Com mais de duas medidas temos um problema **sobre-determinado**: Nº equações > Nº incógnitas. Qualquer solução implica um erro.

A solução que corresponde ao menor erro médio quadrático é interessante

N medidas $(x_k, y_k)(k=1,2,\ldots,N)$

Erro médio quadrático (função de (a,b))

$$\overline{e^2} = \frac{1}{N}\sum_{k=1}^{N}[y_k - (ax_k + b)]^2$$

Erro mínimo:

$$\frac{\partial}{\partial a}\left(\overline{e^2}\right) = \frac{\partial}{\partial b}\left(\overline{e^2}\right) = 0$$

$$a = \frac{\sum x_k y_k - \frac{1}{N}\sum x_k \sum y_k}{\sum x_k^2 \frac{1}{N}(\sum x_k)^2} \qquad b = \frac{1}{N}\left(\sum y_k - a\sum x_k\right)$$

O que precisamos de fazer

Definir variáveis com *N* valores (medidas experimentais) vectores

- Digitando os valores ou
- Lendo um ficheiro de dados

Calcular somatórios

Calcular os coeficientes

Representar graficamente o resultado (pontos experimentais e curva de ajuste)

Estimar o erro?

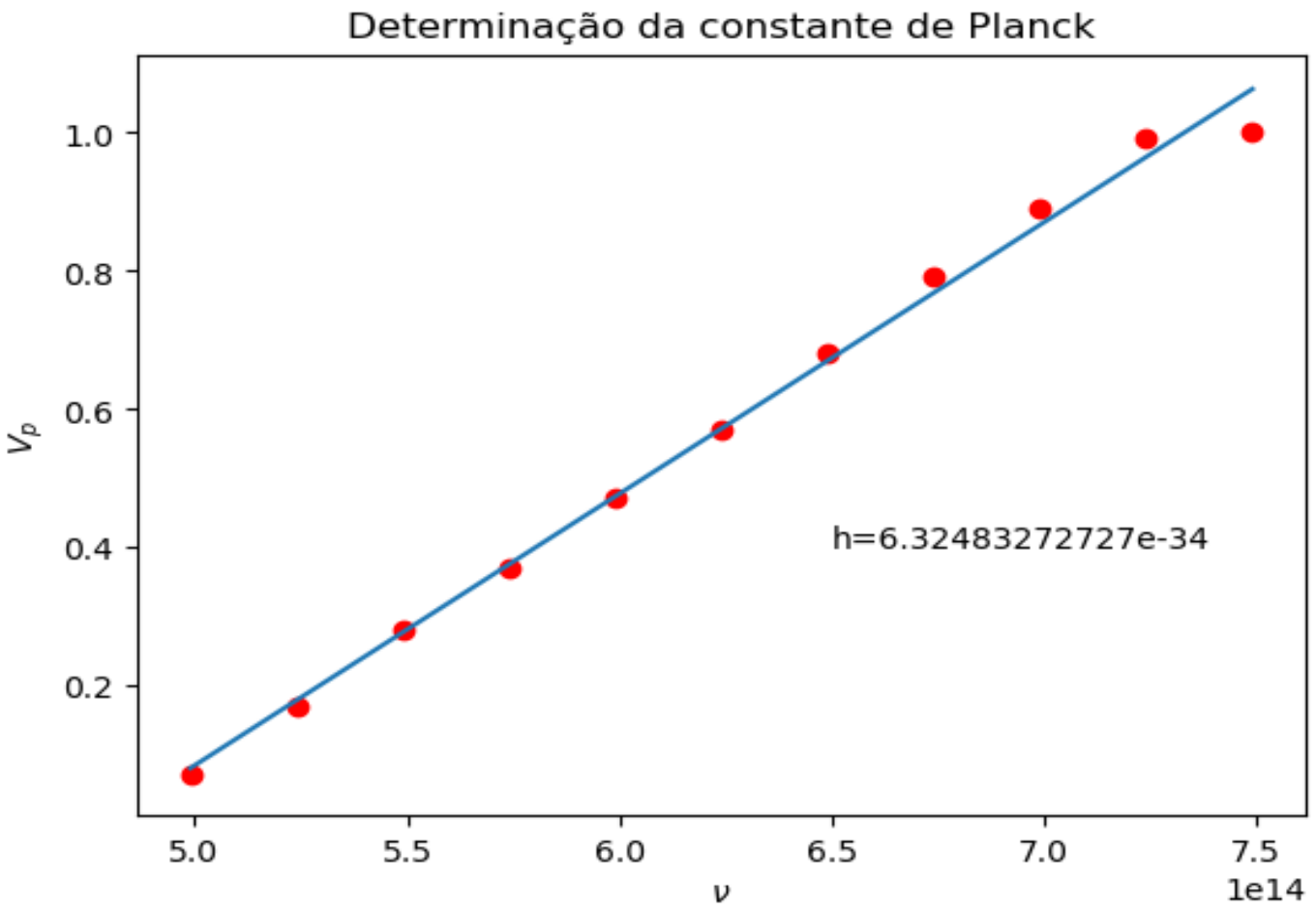
Código(1/2)

```
import numpy as np
import matplotlib.pyplot as plt
nu=1e12*np.array(749,724,699,674,649,624,599,574,549,524,499,dtype
=float)
Vp=np.array(1,0.99,0.89,0.79,0.68,0.57,0.47,0.37,0.2
8,0.17,0.07)
e=1.609e-19
eVp=e*Vp
Sxx=0;Sxy=0;N=len(nu)
for k in range(N):
    Sxy=Sxy+nu[k]*eVp[k]
    Sxx=Sxx+nu[k]*nu[k]
Sx=np.sum(nu)
Sy=np.sum(eVp)
```

Código (2/2)

```
a=(Sxy-Sx*Sy/N)/(Sxx-Sx**2/N)
b=(Sy-a*Sx)/N
h=a
W=-b
plt.scatter(nu,Vp)
fit=(h*nu-W)/e;
plt.plot(nu,fit)
plt.text(6.5e14,0.4,'h='+str(h))
plt.xlabel(r'$\nu$')
plt.ylabel(r'$V_p$')
plt.title('Determinação da constante de
Planck')
print('h=',h)
```

Resultado


Determinação da constante de Planck
1.0
0.8
0.6
0.4
0.2
V_p
h=6.32483272727e-34
5.0
5.5
6.0
6.5
7.0
7.5
1e14
ν

Estruturas de controlo: ciclo com contador

```
for k in range(N): #k=0,1,…,N-1
    a=xk**2
    …

X=1,10,100
for xX in X:
    a=xX**2
    …
```

A linha inicial termina com :

O fim da estrutura é definido por indentação

Ciclo com contador *vs* operação vetorial np

benchmark simples

```
import numpy as np; import time
N=10**6
x=np.ones((N),dtype=float)
y=np.ones((N))*5
t0=time.perf_counter()
Sxy=0
for k in range(len(x)):
    Sxy=Sxy+x[k]*y[k]
t1=time.perf_counter()
SSxy=np.sum(x*y)
t2=time.perf_counter()
print('ciclo=%10.7f npsum=%10.7f aceleração=%7.1f' % (t1-t0,t2-t1,(t1-t0)/(t2-t1)))
>>ciclo= 0.6691751 npsum= 0.0048059 aceleração=  139.2
```

Atenção: o tempo gasto é muito sensível ao valor de N

Leitura de um ficheiro ascii

…

```
Dados=np.loadtxt('dados.dat')
nu=1.e12*Dados[:,0]
Vp=Dados[:,1]
```

Dados.dat

7.49e+02 1.00e+00
7.24e+02 9.90e-01
6.99e+02 8.90e-01
6.74e+02 7.90e-01
6.49e+02 6.80e-01
6.24e+02 5.70e-01
5.99e+02 4.70e-01
5.74e+02 3.70e-01
5.49e+02 2.80e-01
5.24e+02 1.70e-01
4.99e+02 7.00e-02

De volta ao problema $y = ax + b$

Vamos escrever uma função genérica para resolver o problema do ajuste de uma reta a um conjunto de dados por mínimos quadrados. Em relação à Aula 3, vamos acrescentar dois detalhes:

A função considera os dois ajustes lineares:

$$y = ax + b \text{ ou } y = ax$$

A função calcula uma medida da qualidade do ajuste linear, o coeficiente de correlação de Pearson

Novos cálculos

No caso $y = ax$, pode mostrar-se (exercício!) que

$$a = \frac{\sum x_k y_k}{\sum x_k x_k}$$

Em ambos os casos, o coeficiente de correlação de Pearson é:

$$r = \frac{\sum x_k y_k - \frac{1}{N} \sum x_k \sum y_k}{\sqrt{\left[\sum x_k^2 - \frac{1}{N} (\sum x_k)^2\right] \left[\sum y^2 - \frac{1}{N} (\sum y_k)^2\right]}}$$

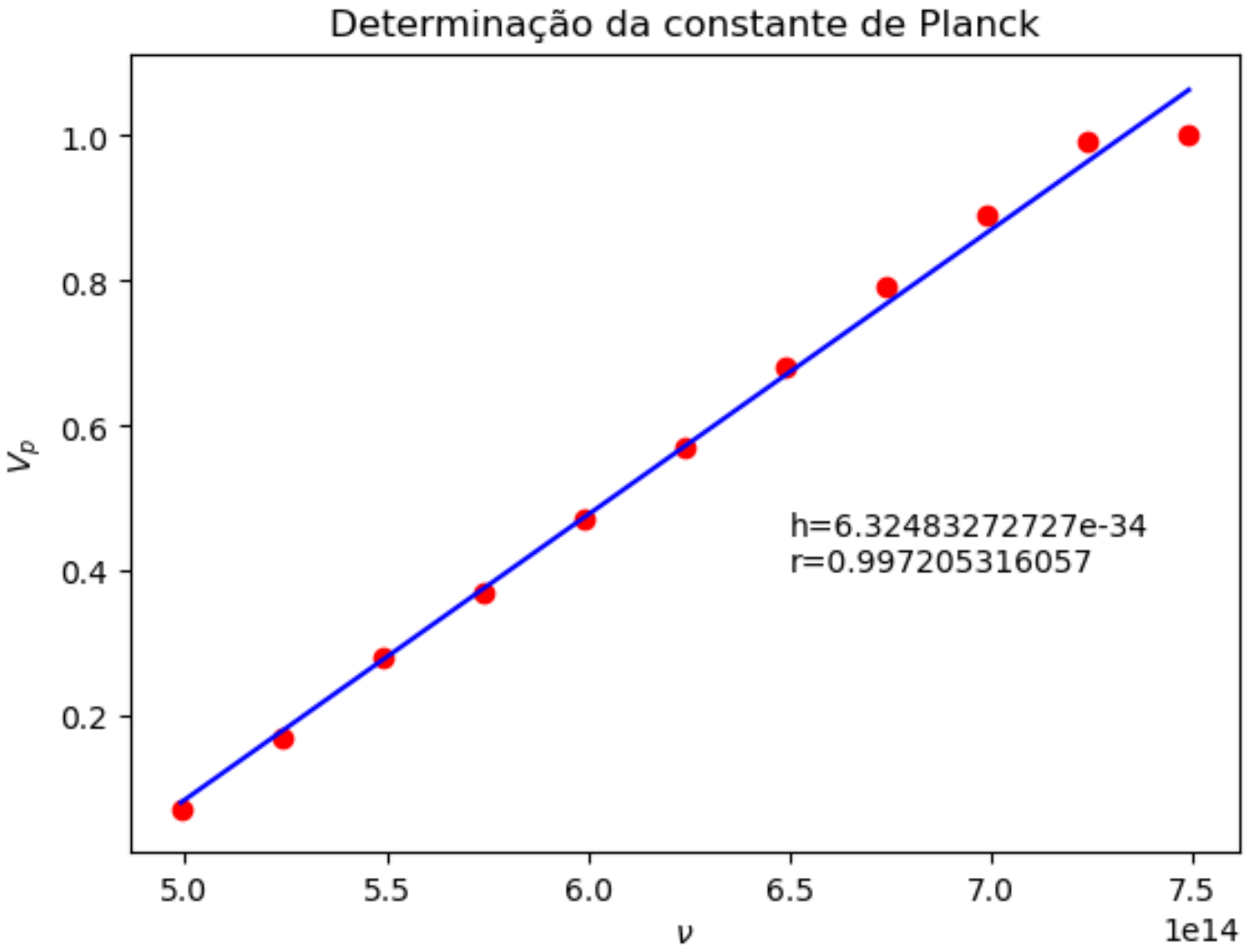
Precisamos de calcular todos os somatórios.

Função linreg (sem ciclos for)

```
import numpy as np #executar antes de chamar linreg
def linreg(x,y,origin):
    n=len(x)
    sx=np.sum(x)
    sy=np.sum(y)
    sx2=np.sum(x*x)
    sy2=np.sum(y*y)
    sxy=np.sum(x*y)
    r=(sxy-sx*sy/n)/np.sqrt((sx2-sx*sx/n)*(sy2-sy*sy/n))
    if origin:
        a=sxy/sx2
        b=0.
    else:
        a=(sxy-sx*sy/n)/(sx2-sx*sx/n)
        b=(sy-a*sx)/n
    return(a,b,r)
```

Utilização da função

```
import matplotlib.pyplot as plt
plt.close('all')
nu=1e12*np.array([749,724,699,674,649,624,599,574,549,524,499],dtype=float)
Vp=np.array([1,0.99,0.89,0.79,0.68,0.57,0.47,0.37,0.28,0.17,0.07])
e=1.609e-19; eVp=e*Vp
(a,b,r)=linreg(nu,eVp,False)
h=a; W=-b
plt.scatter(nu,Vp,color='red') #argumento por dicionário
fit=(h*nu-W)/e;
plt.plot(nu,fit,color='blue')
plt.text(6.5e14,0.4,'h='+str(h)+'\n'+'r='+str(r))
plt.xlabel(r'$\nu$');plt.ylabel(r'$V_p$')
plt.title('Determinação da constante de Planck')
print('h=',h,'r=',r)
```


Determinação da constante de Planck
1.0
0.8
0.6
0.4
0.2
V_p
h=6.32483272727e-34
r=0.997205316057
5.0
5.5
6.0
6.5
7.0
7.5
1e14
ν

$y = ax + b$ *vs* $y = ax$

O coeficiente de correlação não depende do ***fit***

Em certos problemas a reta de regressão **tem** que passar na origem, por razões físicas (por exemplo os valores negativos são inaceitáveis)

Para um dado conjunto de dados as duas opções dão declives diferentes. A opção $y = ax + b$ tem $\leq$ erro médio quadrático (tem mais graus de liberdade).