Nome	Nº	P	G

Exame 1

Identifique-se nas páginas 1 e 3.

1. As funções seguintes contêm (~3) erros fatais. Corrija-as. Admita que, em todos os casos se fez `import numpy as np`

(a) Cálculo de (x, y, z)

$$\begin{cases} x + 3z = 0 \\ x - y + 4z = 0 \\ 2x + 3y + 5 = 0 \end{cases}$$

```
M=[[1,0,3],[1,-1,4],[2,3,5]]
b=np.zeros((3))
X=np.linalg.solve(M,b)
x=X[1];y=X[2];z=X[3]
```

(b) Cálculo $N! = N \times (N-1) \times \ldots \times 2 \times 1$

```
def factorial(N):
    for k in range(N):
        y=y*k
    return y
```

(c) Cálculo do desvio padrão

$$\sigma = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(x_k - \bar{x})^2}\,; \bar{x} = \frac{1}{N}\sum_{k=1}^{N} x_k$$

```
def sig(x):
    sigma=0.
    xmed=np.sum(x)/N
    for k in range(1,N):
        sigma=sigma+(x[k]-xmed)**2
    sigma=np.sqrt(sigma/N)
    return sig
```

(d) Cálculo de $\int_a^b f(x)\,dx$, pelo método do ponto médio

```
def intPM(f,a,b,n):
    dx=(b-a)/n
    x=np.arange(a+dx/2,b-dx/2,dx)
    for k in range(n):
        integral=integral+f[k]
    integral=integral*dx
    return integral
```

(a) corrigido

```
M=np.array([[1,0,3],[1,-1,4],[2,3,0]])
b= np.array([0,0,-5])
X=np.linalg.solve(M,b)
x=X[0];y=X[1];z=X[1]
```

(b) corrigido

```
def factorial(N):
    y=1
    for k in range(2,N+1):
        y=y*k
    return y
```

(c) corrigido

```
def sig(x):
    sigma=0.
    xmed=np.sum(x)/N
    for k in range(N):
        sigma=sigma+(x[k]-xmed)**2
    sigma=np.sqrt(sigma/N)
    return sigma
```

(d) corrigido

```
def intPM(f,a,b,n):
    dx=(b-a)/n
    x=np.arange(a+dx/2,b,dx)
    integral=0.
    for k in range(n):
        integral=integral+f(x[k])
    integral=integral*dx
    return integral
```

2. Um painel negro coberto por três vidros de absorvidade a para a radiação infravermelha, em equilíbrio térmico sob um fluxo solar de irradiância E_S (Wm^{-2}), satisfaz o sistema de equações:

$$-E_0 + E_1 + (1-a)E_2 + (1-a)^2E_3 + E_S = 0$$
$$aE_0 - 2E_1 + aE_2 + a(1-a)E_3 = 0$$
$$a(1-a)E_0 + aE_1 - 2E_2 + aE_3 = 0$$
$$a(1-a)^2E_0 + a(1-a)E_1 + aE_2 - 2E_3 = 0$$

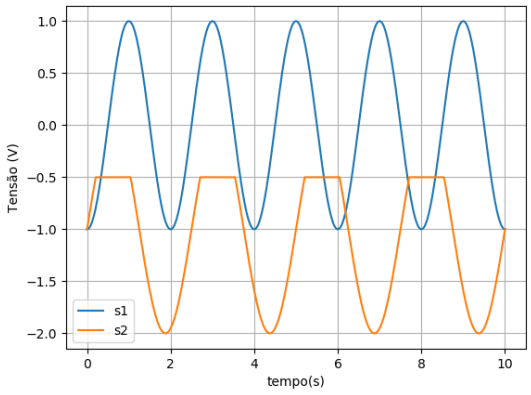
Onde (E_0, E_1, E_2, E_3) são as irradiância do painel e dos vidros, respetivamente, com $E_0 = \sigma T_0^4$ ($\sigma = 5.67 \times 10^{-8} Wm^{-2}K^{-4}$). Escreva uma função ***python*** que devolva a temperatura do painel (T_0), dados os parâmetros (a, E_S), usando a função **np.linalg.solve(M,b).**

```
import numpy as np
def painel(a,Es):
    M=np.array([[-1,1,(1-a),(1-a)**2],[a,-2,a,a*(1-a)],\
                [a*(1-a),a,-2,a],[a*(1-a)**2,a*(1-a),a,-2)]]\
                ,dtype=float)
    b=np.array([-Es,0,0,0],dtype=float)
    E=np.linalg(M,b)
    sigma=5.67e-8
    T0=(E[0]/sigma)**0.25
    return T0
```

3. Escreva um script que produza a figura, incluindo anotações, com uma resolução no eixo das abcissas de 0.01 s (Nota: as partes curvas das linhas são sinusóides).

```
import numpy as np
import matplotlib.pyplot as plt
plt.close('all')

t=np.linspace(0,10,1001)
s=-np.cos(2.*np.pi/2.*t)
plt.plot(t,s,label='s1')
s2=np.sin(2*np.pi*t/2.5)-1
for k in range(len(s2)):
    s2[k]=min(s2[k],-0.5)
plt.plot(t,s2,label='s2')
plt.legend()
plt.xlabel('tempo(s)')
plt.ylabel('Tensão (V)')
plt.grid()
```



Nome	Nº	P	G

4. A função $y = x^3 - e^{-x} - 30$ tem uma raiz no intervalo $[0,5]$. Escreva um script python capaz de localizar essa raiz com um erro inferior a 10^{-6}, usando o método da bissecção.

```
plt.figure()  #opcional não pedido
def f(x):
    fun=x**3-np.exp(-x)-30
    return fun

x=np.linspace(0,10,1001)
plt.plot(x,f(x))
plt.grid()

maxerr=1e-6
xmin=0;xmax=5.
raiz=0.5*(xmin+xmax)
err=0.5*(xmax-xmin)
nit=0
while err>maxerr and nit<30:
    nit=nit+1
    if f(xmin)*f(raiz)<0:
        xmax=raiz
    else:
        xmin=raiz
    raiz=0.5*(xmin+xmax)
    err=0.5*(xmax-xmin)
    print(raiz)
plt.scatter(raiz,f(raiz))
```

5. A função **np.fft.fft(s)**, calcula a transformada discreta de Fourier da série **s**. Escreva uma função ***python*** que, dados uma série de números reais **s** (com **n** termos), um passo de amostragem **dt** e um período de teste **T**, devolva a amplitude da oscilação de período **T** presente na série.

```
import numpy as np
def spectrAmp(s,dt,T):
    n=len(s)
    S=np.fft.fft(s)
    fNyq=1/(2*dt)
    freq=np.linspace(0,fNyq,n//2+1)
    index=np.argmin(np.abs(freq-1/T))
    Amp=np.abs(S[index]/(n//2))
    return Amp
n=100 #opcional, não pedido
dt=1
T=20
t=np.arange(0,n*dt,dt)
s=5*np.sin(2*np.pi*t/T)
plt.figure()
plt.plot(t,s)
AMP=spectrAmp(s,dt,T)
print(AMP)
```

6. Um oscilador harmónico amortecido satisfaz:

$$F = m\frac{d^2x}{dt^2} = -kx - c\frac{dx}{dt}$$

em que k, **m** e **c** são constantes. Escreva a equação na forma de um sistema de 2 equações de primeira ordem. Escreva uma função python a usar na forma **V,X=oscil(v0,x0,t,k,c,m),** que dados os parâmetros (**k,c,m),** a posição e a velocidade iniciais **(x0,v0)** e um vetor de tempos regularmente espaçados (**t**), calcule as séries temporais da posição (**X**) e da velocidade (**V**) ao longo desse tempo. Utilize o método de Euler, com uma iteração para se aproximar do método do ponto médio.

```
# equações
# dv/dt=a=-k/m*x-c/m*v
# dx/dt=v
def oscil(v0,x0,t,k,c,m):
    dt=t[1]-t[0]
    n=len(t)
    X=np.zeros((n))
    V=np.zeros((n))
    X[0]=x0
    V[0]=v0
    for k in range(2,n):
        V[k]=V[k-1]+(-k/m*X[k-1]-c/m*V[k-1])*dt #Euler
        Vm=0.5*(V[k-1]+V[k])
        X[k]=X[k-1]+Vm*dt
        Xm=0.5*(X[k-1]+X[k])
        V[k]=V[k-1]+(-k/m*Xm-c/m*Vm)*dt
        Vm=0.5*(V[k-1]+V[k])
        X[k]=X[k-1]+Vm*dt
    return X,V
```