Solución Examen Midterm
FISI 6510

Guillermo Fidalgo

19 de marzo de 2021

a)

Si tenemos que

$$I(\omega) = \frac{\hbar}{4\pi^2 c^2} \frac{\omega^3}{\left(e^{\hbar\omega/k_B T} - 1\right)}$$

Entonces la energía total por unidad de área es $W = \int I(\omega) d\omega$.

$$W = \frac{\hbar}{4\pi^2 c^2} \int_0^\infty \frac{\omega^3}{\left(e^{\hbar\omega/k_B T} - 1\right)} d\omega$$

hacemos la susitución

$$x = \frac{\hbar\omega}{k_B T} \to \omega = \frac{k_B T}{\hbar} x$$

y

$$dx = \frac{\hbar}{k_B T} d\omega \to d\omega = \frac{k_B T}{\hbar} dx$$

y obtenemos que

$$W = \frac{\hbar}{4\pi^2 c^2} \int_0^\infty \frac{(k_B T/\hbar)^3 x^3}{(e^x - 1)} (k_B T/\hbar) dx$$
$$W = \frac{k_B^4 T^4}{4\pi^2 c^2 \hbar^3} \int_0^\infty \frac{x^3}{(e^x - 1)} dx$$

b)

Tomamos los códigos para cuadratura de gauss provista por el libro y para integración impropia

[28]:
```
from numpy import ones,copy,cos,tan,pi,linspace,exp,array,pi

def gaussxw(N):

    # Initial approximation to roots of the Legendre polynomial
    a = linspace(3,4*N-1,N)/(4*N+2)
    x = cos(pi*a+1/(8*N*N*tan(a)))
```

```
    # Find roots using Newton's method
    epsilon = 1e-15
    delta = 1.0
    while delta>epsilon:
        p0 = ones(N,float)
        p1 = copy(x)
        for k in range(1,N):
            p0,p1 = p1,((2*k+1)*x*p1-k*p0)/(k+1)
        dp = (N+1)*(p0-x*p1)/(1-x*x)
        dx = p1/dp
        x -= dx
        delta = max(abs(dx))

    # Calculate the weights
    w = 2*(N+1)*(N+1)/(N*N*(1-x*x)*dp*dp)

    return x,w

def gaussxwab(N,a,b):
    x,w = gaussxw(N)
    return 0.5*(b-a)*x+0.5*(b+a),0.5*(b-a)*w
```

Cambiamos de variable para que la integral ahora sea de la siguiente forma

$$W = \frac{k_B^4 T^4}{4\pi^2 c^2 \hbar^3} \int_0^\infty \frac{x^3}{(e^x - 1)} dx$$

$$= \frac{k_B^4 T^4}{4\pi^2 c^2 \hbar^3} \int_0^1 \frac{z^3/(1-z)^3}{(e^{(z/(1-z)} - 1)} \frac{1}{(1-z)^2} dz = \frac{k_B^4 T^4}{4\pi^2 c^2 \hbar^3} \int_0^1 \frac{z^3}{(e^{(z/(1-z)} - 1)} \frac{1}{(1-z)^5} dz$$

Solo evaluamos el integrando de la expresión (sin incluir las constantes)

[29]:
```
#cambiamos de variable nuestra función en terminos de z

def f(z):
    return ((z)**3/(exp(z/(1-z))-1))*(1/(1-z)**5)

def integrate(f,a,b,N):
    x,w = gaussxwab(N,a,b)
    s = 0.0
    for k in range(N):
        s += w[k]*f(x[k])
    return s
```

[30]:
```
Err=[]
for i in range(1,31):
    I1=integrate(f,a=0.0,b=1.0,N=i)
    I2=integrate(f,a=0.0,b=1.0,N=i+1)
```

```
    Err.append(abs(I2-I1))
    print("error es {:2.1e}  valor de integral es {:9.6f} con N={}".
 ↪format(abs(I2-I1),I1,i))
# print("Valor de integral es",I1,"con N=%i" %i)
```

```
error es 1.2e+01  valor de integral es  2.327907 con N=1
error es 9.2e+00  valor de integral es 14.327015 con N=2
error es 1.1e+00  valor de integral es  5.106522 con N=3
error es 2.8e+00  valor de integral es  4.055851 con N=4
error es 5.7e-01  valor de integral es  6.851614 con N=5
error es 8.7e-01  valor de integral es  7.418290 con N=6
error es 3.9e-01  valor de integral es  6.552743 con N=7
error es 2.1e-01  valor de integral es  6.164620 con N=8
error es 1.9e-01  valor de integral es  6.378575 con N=9
error es 7.4e-03  valor de integral es  6.572143 con N=10
error es 6.8e-02  valor de integral es  6.564775 con N=11
error es 2.7e-02  valor de integral es  6.496397 con N=12
error es 1.1e-02  valor de integral es  6.469618 con N=13
error es 1.5e-02  valor de integral es  6.480880 con N=14
error es 4.0e-03  valor de integral es  6.495923 con N=15
error es 3.2e-03  valor de integral es  6.499877 con N=16
error es 3.3e-03  valor de integral es  6.496673 con N=17
error es 8.3e-04  valor de integral es  6.493372 con N=18
error es 7.0e-04  valor de integral es  6.492543 con N=19
error es 7.7e-04  valor de integral es  6.493243 con N=20
error es 2.5e-04  valor de integral es  6.494009 con N=21
error es 1.2e-04  valor de integral es  6.494260 con N=22
error es 1.8e-04  valor de integral es  6.494140 con N=23
error es 8.5e-05  valor de integral es  6.493959 con N=24
error es 8.0e-06  valor de integral es  6.493874 con N=25
error es 4.0e-05  valor de integral es  6.493882 con N=26
error es 2.7e-05  valor de integral es  6.493921 con N=27
error es 5.6e-06  valor de integral es  6.493949 con N=28
error es 6.6e-06  valor de integral es  6.493954 con N=29
error es 7.7e-06  valor de integral es  6.493948 con N=30
```

Dejándonos llevar por nuestro estimado del error pienso que el error es del orden 1×10^{-6} con $N = 30$

c)

Estimando la constante σ de Stefan-Boltzmann utilizando $W = \sigma T^4$. Si igualamos ambas ecuaciónes y comparamos tenemos que

$$\sigma T^4 = \frac{k_B^4 T^4}{4\pi^2 c^2 \hbar^3} \int_0^1 \frac{z^3}{(e^{(z/(1-z)} - 1)} \frac{1}{(1-z)^5} dz$$

y por lo tanto σ debería ser

$$\sigma = \frac{k_B^4}{4\pi^2 c^2 \hbar^3} \int_0^1 \frac{z^3}{(e^{(z/(1-z)} - 1)} \frac{1}{(1-z)^5} dz$$

[35]:
```
Int=integrate(f,a=0.0,b=1.0,N=30)
kB=1.38e-23
c=3e8
hbar=6.626e-34/(2*pi)
```

[36]:
```
sigma = kB**4/(4*pi**2 * c**2 * hbar**3)
```

[44]:
```
print("sigma={:.3e}".format(sigma))
```

```
sigma=8.704e-09
```

Problema 2

a)

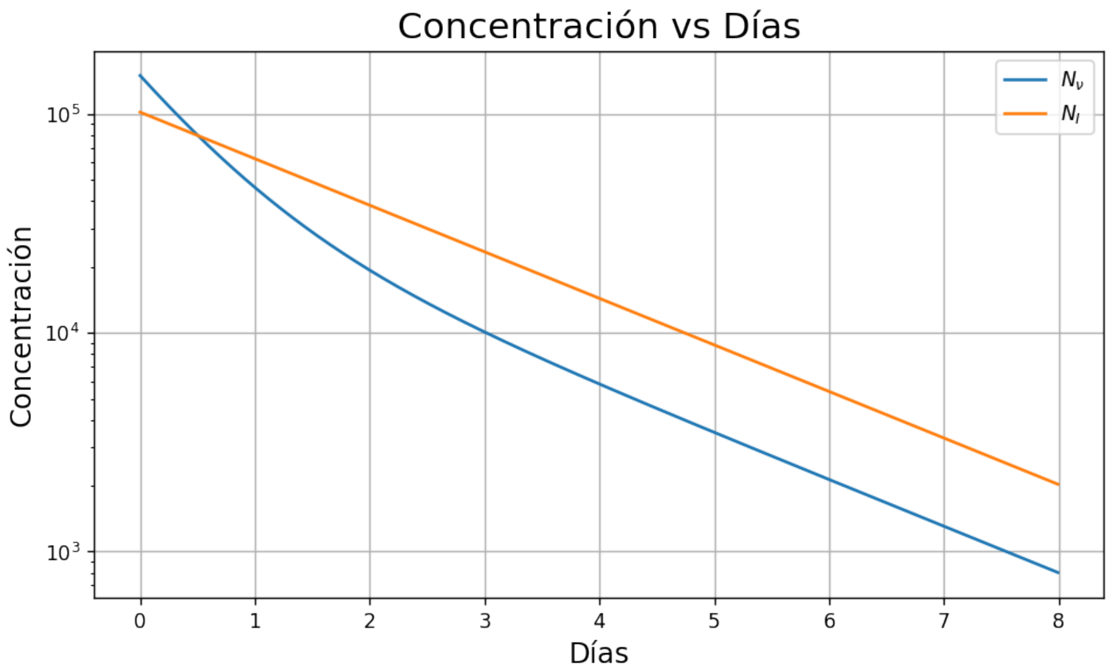
[7]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[12]:
```
##### constantes #####
kv=1.63 # 1/día
kI=0.49 # 1/día
gamma=0.45 # 1/día
a = 0.0
b = 8.0
N = 1000
h = (b-a)/N
######################


def f(r,t):
    Nv=r[0]
    NI=r[1]
    fNv=-kv*Nv + gamma*NI
    fNI=-kI*NI
    return np.array([fNv,fNI],float)

def rk4_2d(f,a,b,N,Init_cond=[0,0]):
    h = (b-a)/N
    tpoints = np.arange(a,b,h)
    xpoints = []
    ypoints = []
    r = np.array(Init_cond,float)
    for t in tpoints:
        xpoints.append(r[0])
        ypoints.append(r[1])
        k1 = h*f(r,t)
        k2 = h*f(r+0.5*k1,t+0.5*h)
        k3 = h*f(r+0.5*k2,t+0.5*h)
        k4 = h*f(r+k3,t+h)
        r += (k1+2*k2+2*k3+k4)/6
    return tpoints,xpoints,ypoints
```

[13]:
```
cond_inciales=[1.5e5,1.02e5]

tpoints,NV,NI=rk4_2d(f,a,b,N,Init_cond=cond_inciales)
```

[14]:
```
plt.figure(figsize=(9,5),dpi=124)
plt.plot(tpoints,NV,label=r'$N_\nu$')
```

```
plt.plot(tpoints,NI,label=r'$N_I$')
plt.semilogy()
plt.ylabel("Concentración",size=14)
plt.xlabel("Días",size=14)
plt.legend()
plt.title('Concentración vs Días',size=18)
plt.grid()
plt.show()
```



b)

Asumiendo que el medicamento antiviral detiene completamente las infecciones de las células T en el día $t = 0$. Observamos que la concentración del virus decae rápidamente de manera exponencial, incluso mueren más rapido que lo que se eliminan las células T del sisteman inmunológico en el primer día. Para el segundo día en adelante ambas concentraciones decaen exponencialmente.

Problema 3

b)

Cambiamos estas 2 ecuaciones a 4 de primero orden de la siguiete manera.

$$v_x = \dot{x} \qquad\qquad v_y = \dot{y}$$

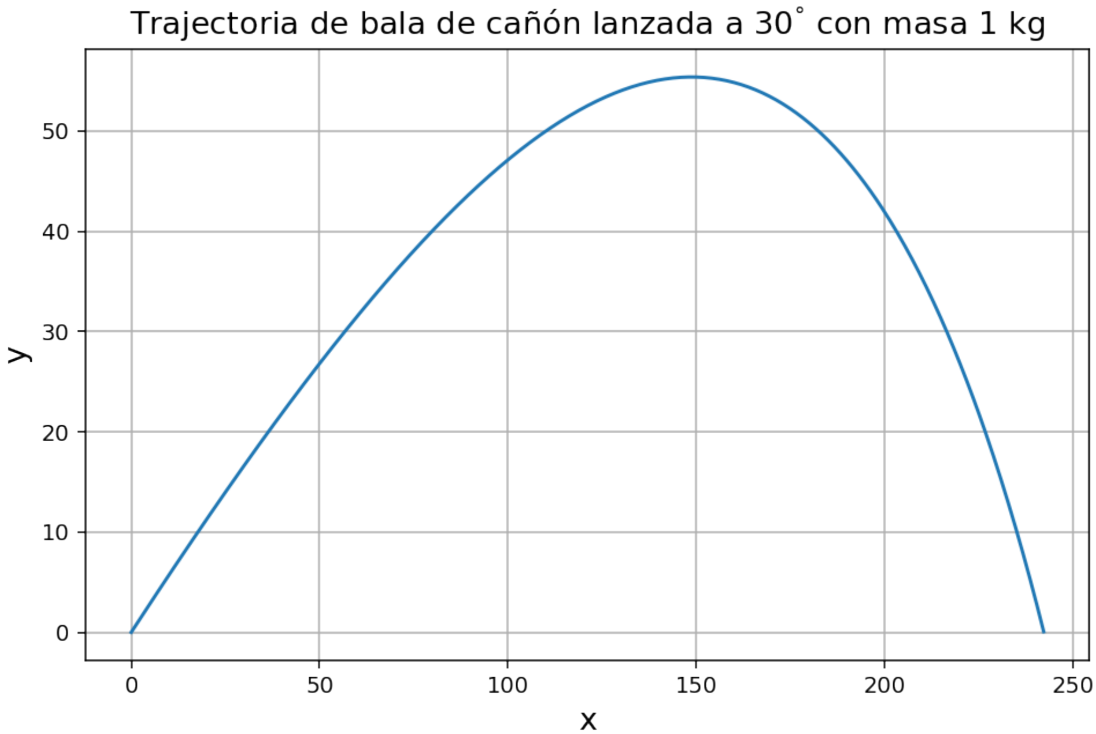
$$\frac{dv_x}{dt} = \ddot{x} \qquad\qquad \frac{dv_y}{dt} = \ddot{y}$$

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
def f(r,t):
    x=r[0]
    y=r[1]
    vx=r[2]
    vy=r[3]
    fxx= -((np.pi*R**2 *rho *C)/(2*m)) *vx*np.sqrt(vx**2 + vy**2)
    fyy= -g -((np.pi*R**2 *rho *C)/(2*m)) *vy*np.sqrt(vx**2 + vy**2)
    return np.array([vx,vy,fxx,fyy],float)

def rk4_2d(f,a,b,N,Init_cond=[0,0]):
    h = (b-a)/N
    tpoints = np.arange(a,b,h)
    xpoints = []
    ypoints = []
    vxpoints= []
    vypoints= []
    r = np.array(Init_cond,float)
    for t in tpoints:
        if r[1] <0:
            #detener el cálculo cuando se llegue al suelo
            #print('breaking at t={}'.format(t))
            break

        xpoints.append(r[0])
        ypoints.append(r[1])
        vxpoints.append(r[2])
        vypoints.append(r[3])
        k1 = h*f(r,t)
        k2 = h*f(r+0.5*k1,t+0.5*h)
        k3 = h*f(r+0.5*k2,t+0.5*h)
        k4 = h*f(r+k3,t+h)
        r += (k1+2*k2+2*k3+k4)/6
    return tpoints,xpoints,ypoints,vxpoints,vypoints
```

```
m=1 #kg
R=0.08 #m
angle=30*np.pi/180
v0x=100*np.cos(angle)
v0y=100*np.sin(angle)
init_cond=[0,0,v0x,v0y]
rho=1.22 #kg/m^3
C=0.47
g=9.81 #m/s^2

a=0
b=100
N=5000
```

[3]:
```
t,x,y,vx,vy= rk4_2d(f,a,b,N,Init_cond=init_cond)

plt.figure(figsize=(8,5),dpi=140)
plt.plot(x,y)
plt.title(r"Trajectoria de bala de cañón lanzada a $30^\degree$ con masa 1␣
 ↪kg",size=14)
plt.xlabel("x",size=14)
plt.ylabel('y',size=14)
plt.grid()
plt.show()
```

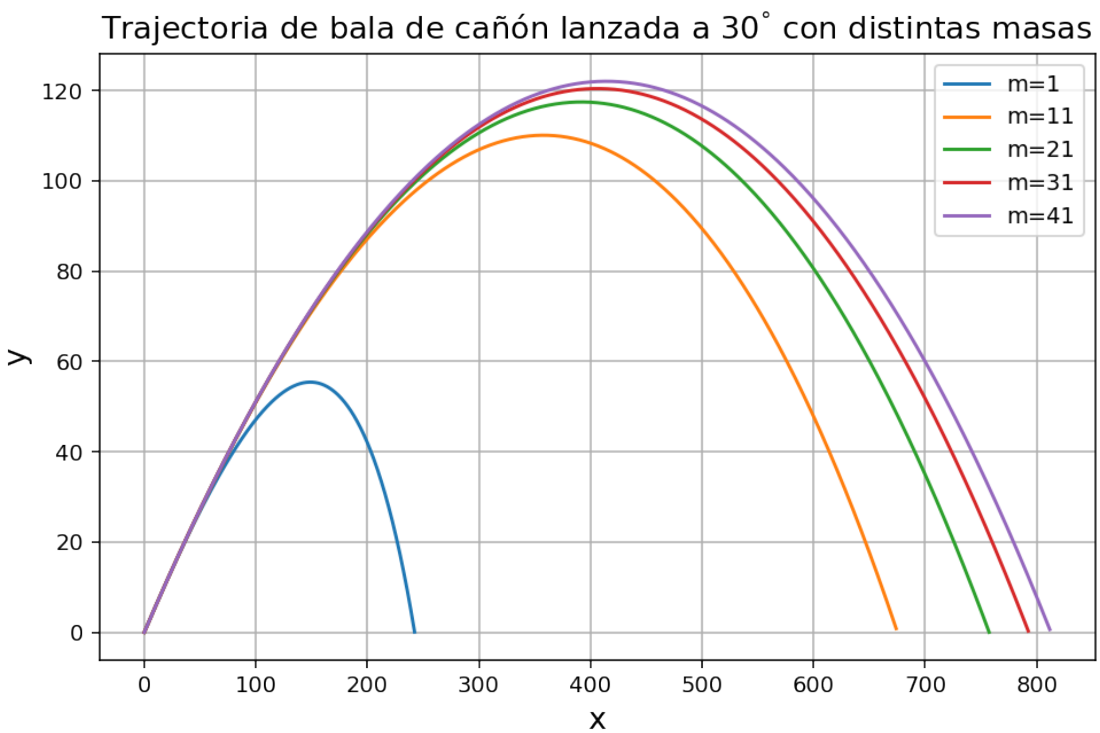


c)

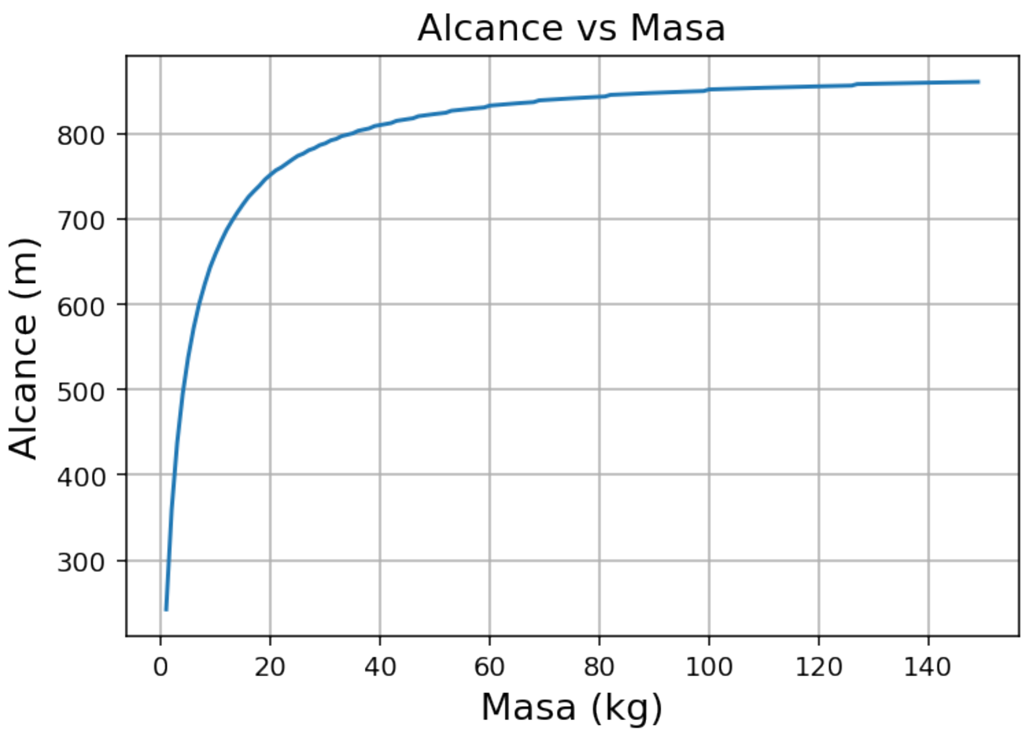
[4]:
```
print("El proyectil (con masa {} kg) viajó una distancia máxima de {:.3f} m con␣
↪resistencia de aire".format(m,x[-1]))
```

```
El proyectil (con masa 1 kg) viajó una distancia máxima de 242.296 m con
resistencia de aire
```

[5]:
```
plt.figure(figsize=(8,5),dpi=140)
for m in np.arange(1,50,10):
    t,x,y,vx,vy= rk4_2d(f,a,b,N,Init_cond=init_cond)
    plt.plot(x,y,label='m={}'.format(m))
plt.title(r"Trajectoria de bala de cañón lanzada a $30^\degree$ con distintas␣
↪masas",size=14)
plt.xlabel("x",size=14)
plt.ylabel('y',size=14)
plt.grid()
plt.legend()
plt.show()
```



[6]:
```
plt.figure(dpi=140)
M=[]
alcance=[]
for m in np.arange(1,150):
    t,x,y,vx,vy= rk4_2d(f,a,b,N,Init_cond=init_cond)
    M.append(m)
    alcance.append(x[-1])
plt.plot(M,alcance)
plt.title(r"Alcance vs Masa",size=14)
plt.xlabel("Masa (kg)",size=14)
plt.ylabel('Alcance (m)',size=14)
plt.grid()
plt.show()
```



Lo que podemos observar es que la mientras más masa tiene el objeto mayor distancia puede viajar. Esto es debido a que la esfera tendría cada vez mas densidad ya que no estamos aumentando sus dimensiones (Radio constante) y por lo tanto siente cada vez menos los efectos de la fuerza de arrastre y adicionalmente tendrá más inercia que evite que cambie su estado de movimiento mientras aumentemos la masa.