Universidad Autónoma de Querétaro

Facultad de Ingeniería

Examen Parcial II

Análisis numérico

Autor:
David Gómez Torres

15 de Noviembre del 2021

ÍNDICE GENERAL

CAPÍTULO 1

METODOLOGÍA

1.1. Estaciones de radar

Problema 1

Las estaciones de radar A y B, separadas por una distancia $a = 500$ m, rastrean el avión C grabando los ángulos α y β en intervalos de 1 segundo.

Si tres sucesivas mediciones son:

t(s)	9	10	11
α	54.8°	54.06°	53.34°
β	65.59°	64.59°	63.62°

calcule la velocidad v del avión y el ángulo de subida γ a $t = 10$ s.

Se puede mostrar que las coordenadas del avión son:

$$x = a\frac{\tan\beta}{\tan\beta - \tan\alpha} \tag{1.1}$$

$$y = a\frac{\tan\alpha\tan\beta}{\tan\beta - \tan\alpha} \tag{1.2}$$

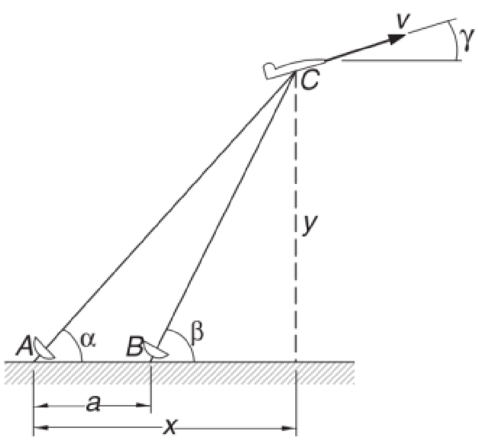


Figura 1.1: Representación gráfica del problema

1.1.1. Planteamiento del problema

La tabla que nos presenta en el problema, nos brinda tanto el ángulo beta β como alfa α en los tiempos: 9, 10 y 11.

De las ecuaciones (1.1) y (1.2) calculamos las coordenadas x_i y y_i en el intervalo de tiempo.

Recordemos que la velocidad se define como

$$v_i = \frac{dx_i}{dt} = \dot{x}_i \tag{1.3}$$

entonces, debemos calcular la derivada de x y y en el tiempo $t = 10s$.

Ahora, para encontrar la magnitud de la velocidad utilizamos la expresión

$$v = \sqrt{v_x^2 + v_y^2} = \sqrt{\dot{x}^2 + \dot{y}^2} \tag{1.4}$$

Mientras que para el ángulo γ, tenemos de la trigonometría la expresión para calcular

$$\gamma = \tan^{-1}\left(\frac{y(t_i)}{x(t_i)}\right) \tag{1.5}$$

1.1.2. Pseudocódigo

Algorithm 1 Cálculo de la velocidad y ángulo del avión

Entrada:
alpha,beta,t; list
a; int
func coordenadas
$x = a\dfrac{\tan\beta}{\tan\beta - \beta\alpha}$ $\qquad$ $y = a\dfrac{\tan\alpha\tan\beta}{\tan\beta - \tan\alpha}$
return x_i, y_i
func angulo
$\gamma = \tan^{-1}\left(\dfrac{y(t_i)}{x(t_i)}\right)$
return γ
func derivada $\qquad$ ▷ Datos espaciados irregularmente
print ``X1'' $\leftarrow X_1$
Dx $\leftarrow$
print ``Y1'' $\leftarrow Y_1$
Dy $\leftarrow$
vel $\leftarrow \sqrt{Dx^2 + Dy^2}$
return vel

Salida: gamma,vel

1.1.3. Código fuente del problema

1. Solución de la derivada para datos irregularmente espaciados

```
import math

#distancia entre radares (cte)
a = 500
#lista con angulos (rad)
alpha = [0.9564404,0.94352499,0.93095862]
beta = [1.1447615,1.1273082,1.1103785]
t = [9,10,11]
#listas vacias para coordenadas
x =[]
y =[]

def coordenadas(alpha,beta):
    cadena0 = "velocidad y angulo de subida de un avion\n".capitalize()
    print(cadena0.center(70, " "))

    #para x
    for i in range(0,3):
        xi = (a*math.tan(beta[i]))/(math.tan(beta[i])-math.tan(alpha[i]))
        x.append(xi)
    print("Las coordenadas en 'x'",x,"m")

    #para y
    for i in range(0,3):
        yi = (a*math.tan(alpha[i])*math.tan(beta[i]))
        /(math.tan(beta[i])-math.tan(alpha[i]))
        y.append(yi)
    print("Las coordenadas en 'y'",y,"m\n")

coordenadas(alpha,beta)

def angulo(x,y):
    gamma = math.atan((y[1])/(x[1]))
    print("El angulo en 't(10s)' es: ",gamma,"rad")

angulo(alpha,beta)

def derivada(x,y,t):
#derivada en x
    x1 = float(input("Ingrese el tiempo donde quiere evaluar 'x': "))

    Dx = (t[0]*(2*x1-x[1]-x[2])/((x[0]-x[1])*(x[0]-x[2])))+
    (t[1]*(2*x1-x[0]-x[2])/((x[1]-x[0])*(x[1]-x[2])))
    +(t[2]*(2*x1-x[0]-x[1])/((x[2]-x[0])*(x[2]-x[1])))
    print("La derivada en 't(",x1,"s)' es ",Dx)
```

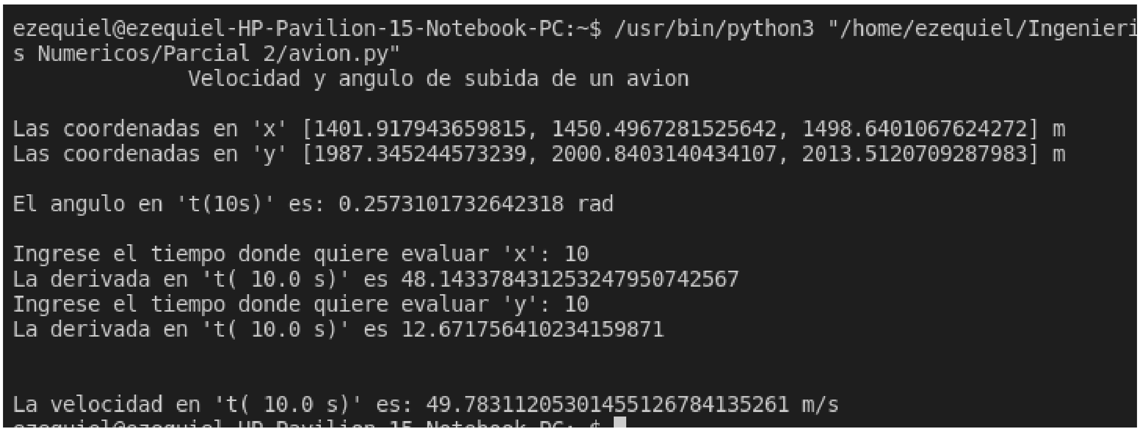
```python
#derivada en y
    y1 = float(input("Ingrese el tiempo donde quiere evaluar 'y': "))

    Dy = (t[0]*(2*x1-y[1]-y[2])/((y[0]-y[1])*(y[0]-y[2])))
    +(t[1]*(2*x1-y[0]-y[2])/((y[1]-y[0])*(y[1]-y[2])))
    +(t[2]*(2*x1-y[0]-y[1])/((y[2]-y[0])*(y[2]-y[1])))
    print("La derivada en 't(",y1,"s)' es ",Dy,"\n\n")

#calcular la velocidad
    vel = math.sqrt(Dx**2+Dy**2)
    print("La velocidad en 't(",x1,"s)' es: ",vel,"m/s")

derivada(x,y,t)
```

1.1.4. Ejecución del código



```
ezequiel@ezequiel-HP-Pavilion-15-Notebook-PC:~$ /usr/bin/python3 "/home/ezequiel/Ingenieri
s Numericos/Parcial 2/avion.py"
              Velocidad y angulo de subida de un avion

Las coordenadas en 'x' [1401.917943659815, 1450.4967281525642, 1498.6401067624272] m
Las coordenadas en 'y' [1987.345244573239, 2000.8403140434107, 2013.5120709287983] m

El angulo en 't(10s)' es: 0.2573101732642318 rad

Ingrese el tiempo donde quiere evaluar 'x': 10
La derivada en 't( 10.0 s)' es 48.143378431253247950742567
Ingrese el tiempo donde quiere evaluar 'y': 10
La derivada en 't( 10.0 s)' es 12.671756410234159871


La velocidad en 't( 10.0 s)' es: 49.78311205301455126784135261 m/s
```

Figura 1.2: Ejecución del código

La velocidad del avión: 49.7831 m/s
El ángulo de subida: 0.2573 rad

1.2. Trabajo al estirar un arco

Problema 2

La siguiente tabla muestra la fuerza F del arco en función del tirón x. Si el arco se tira 0.5 m, determine la velocidad de la flecha de 0.075 kg cuando sale del arco.

x(m)	0	0.05	0.10	0.15	0.2	0.25
F(N)	0	37	71	104	134	161
x(m)	0.30	0.35	0.4	0.45	0.5	
F(N)	185	207	225	239	250	

Sugerencia: la energía cinética de la flecha es igual al trabajo realizado al tensar el arco, es decir:

$$\frac{1}{2}mv^2 = \int_0^{0.5m} F\ dx$$

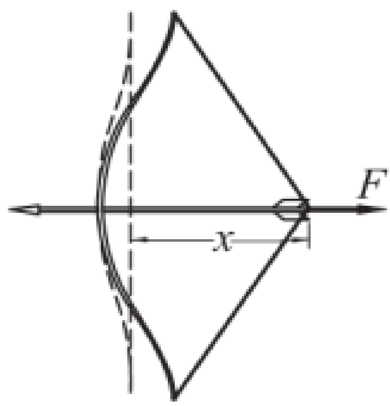


Figura 1.3: Representación del problema

1.2.1. Planteamiento del problema

Para encontrar la velocidad de la flecha, podemos utilizar el siguiente teorema:

Teorema trabajo-energía:

El trabajo neto efectuado por las fuerzas que actúan sobre una partícula es igual al cambio en la energía cinética de la partícula.

La representación matemática de este teorema es la siguiente:

$$W_{\text{neto}} = K_f - K_i = \Delta K \tag{1.6}$$

de donde se sigue, de la definición de energía cinética, con $K_i = 0$

$$W_{\text{neto}} = \frac{1}{2}mv^2 \tag{1.7}$$

Recordemos que podemos escribir el trabajo total efectuado por F al desplazar un cuerpo desde x_i hasta x_f así

$$W = \int_{x_i}^{x_f} F(x)\, dx \tag{1.8}$$

En la ecuación (1.7), sustituimos con (1.8) y despejamos para v^2

$$v^2 = \frac{2}{m}\int_{x_i}^{x_f} F(x)\, dx \tag{1.9}$$

Entonces, la expresión que vamos a utilizar para resolver el problema esta dada por

$$v = \sqrt{\frac{2}{m}W} = \sqrt{\frac{2}{m}\int_{x_i}^{x_f} F(x)\, dx} \tag{1.10}$$

1.2.2. Pseudocódigo

Algorithm 2 Trabajo al estirar un arco

```
Entrada: x,F; list
  m; float
      func trabajo
  h ← 0.05
  sum ← 0
  for i ← n do                                   ▷ Regla del trapecio
      sum += F_i+F_{i+1}
      i ← i-1
  end for
  W = (h/2) * sum
  vel = √(2W/m)
  print ''velocidad'' ← vel
Salida: vel; float
```

1.2.3. Código solución al problema

1. Solución de la integral por regla del trapecio con segmentos desiguales

```python
import math

cadena = "Calculo del trabajo al estirar un arco".capitalize()
print(cadena.center(50, " "))

#masa de la flecha (cte)
m = 0.075
#lista con datos del problema
F = [0,37,71,104,134,161,185,207,225,239,250]

def trabajo(F,m):
#regla del trapecio para datos irregularmente espaciados
    h = 0.05 #paso constante
    sum = 0
    for i in range(0,10):
        sum += F[i]+F[i+1]
        i = i-1
    W = (h/2) * sum
    print("El trabajo realizado es: ",W,"N")

#calculo de la velocidad
    vel = math.sqrt(2*W/m)
    print("La velocidad de la flecha es: ",vel,"m/s")

trabajo(F,m)
```

1.2.4. Ejecución del código

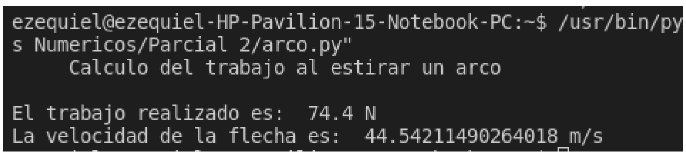


Figura 1.4: Ejecución del código

El trabajo: $74.4N$
La velocidad de la flecha: $44.54\ m/s$

1.3. Ajuste a función exponencial

Problema 3

Ajuste la función $f(x) = axe^{bx}$ a los siguientes datos y calcule la desviación estándar.

x	0.5	1.0	1.5	2.0	2.5
y	0.541	0.398	0.232	0.106	0.052

1.3.1. Planteamiento del problema

En este caso se nos pide ajustar los datos a la función del tipo

$$f(x) = a_1 x e^{\beta_1 x} \tag{1.11}$$

donde a_1 y β_1 son constantes.

Para linealizar la ecuación (1.11), se aplica el logaritmo natural en ambos lados de la ecuación, obteniendo:

$$\ln y = \ln a_1 + \beta_1 x \ln e \tag{1.12}$$

Para encontrar los coeficientes, vamos a utilizar un ajuste de la recta por mínimos cuadrados que están dados por:

Regresión lineal por mínimos cuadrados:

$$a_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} \tag{1.13}$$

$$a_0 = \bar{y} - a_1 \bar{x} \tag{1.14}$$

donde $\bar{y}$ y $\bar{x}$ son las medias de y y x, respectivamente.

Además, se pide calcular la desviación estándar que se formula como sigue:

$$S_{y/x} = \sqrt{\frac{S_r}{n-2}} \tag{1.15}$$

1.3.2. Pseudocódigo

Algorithm 3 Regresión lineal a datos exponenciales

Entrada:
x,y; list
sumx=0: sumxy=0: st=0
sumy=0: sumx2=0: sr=0
for i=1,n **do**
    sumx = sumx + x_i
    sumy = sumy + y_i
    sumxy = sumxy + x_i*y_i
    sumx2 = sumx2 + x_i*x_i
end for
xm = sumx/n
ym = sumy/n
a1 = (n*sumxy-sumx*sumy)/(n*sumx2-sumx*sumx)
a0 = ym - a1*xm
for i=1,n **do**
    st = st + $(y_i - y_m)^2$
    sr = sr + $(y_i - a1 * x_i - a0)^2$
end for
syx = $(sr/(n-2))^{0.5}$
r2 = (st - sr)/st
Salida:
recta; plot

1.3.3. Código solución

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

cadena = "Regresión lineal a datos exponenciales".capitalize()
print(cadena.center(50, " "))

#arreglo con valores
x=[0.5,1,1.5,2,2.5]
y=[0.541,0.398,0.232,0.106,0.052]

#numeros de datos
n = len(x)
#convertir en vectores
x = np.array(x)
y = np.array(y)

#sumatorias
sumx = sum(x)
sumy = sum(y)
sumx2 = sum(x**2)
sumy2 = sum(y**2)
sumxy = sum(x*y)

#promedio de datos
xm = sumx/n
ym = sumy/n
#valores de la recta
a1 = (sumx*sumy-n*sumxy)/((sumx)**2 - n*sumx2) #pendiente (m)
a0 = ym - a1*xm #interseccion con el eje (b)
#errores del metodo
st = sum((y - ym)**2)
sr = sum((y - a1*x - a0)**2)

syx = math.sqrt(sr/(n-2)) #error estand.
r = math.sqrt((st - sr)/st) #coef.correlacion
print("Error estandar: ", syx)
print("Coeficiente de correlacion: ", r)

print('Pendiente               Interseccion con el eje X')
print(f'{a1:10} ==> {a0:10f}')
```

```
#graficar
plt.plot(x,y, 'o', label='Datos',color='red') #puntos
plt.plot(x, np.log(a0)+a1*x, label='-0.254x+0.6468',color='green') #recta
#titulo de ejes
plt.xlabel('Eje x')
plt.ylabel('Eje y')
plt.grid()#generar malla
plt.title('Regresion lineal')#titulo de grafica
plt.legend()#mostrar recuadro
plt.show()#mostrar grafica
```

1.3.4. Ejecución del código

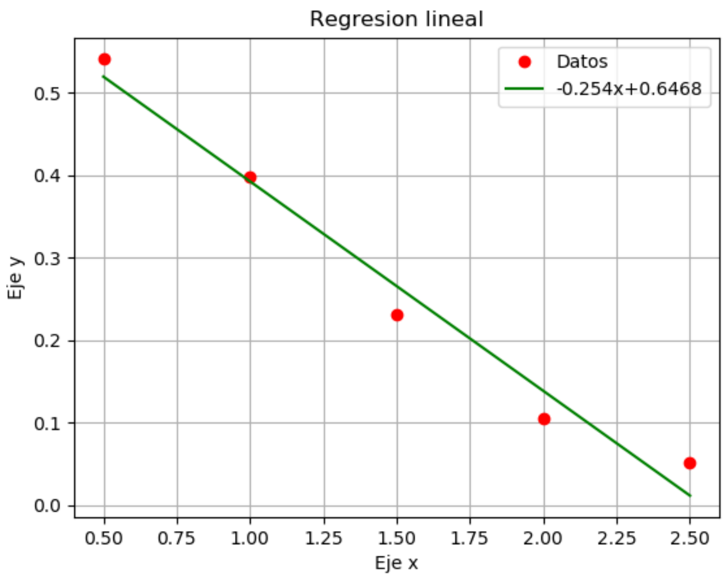
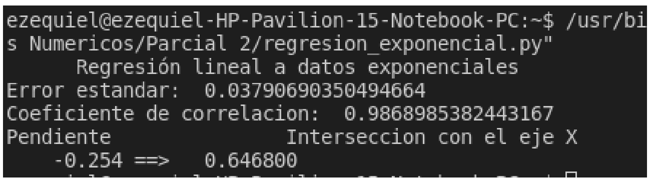


Figura 1.5: Ajuste del modelo a los datos



```
ezequiel@ezequiel-HP-Pavilion-15-Notebook-PC:~$ /usr/bi
s Numericos/Parcial 2/regresion_exponencial.py"
      Regresión lineal a datos exponenciales
Error estandar:  0.03790690350494664
Coeficiente de correlacion:  0.9868985382443167
Pendiente                Interseccion con el eje X
    -0.254 ==>   0.646800
```

Figura 1.6: Ejecución del código

Recta ajustada: -0.254x+0.6468

CAPÍTULO 2

BIBLIOGRAFÍA

1. Chapra, S. C., Canale, R. P. (2011). *Numerical methods for engineers* (Vol. 1221). New York: Mcgraw-hill.

2. Kiusalaas, J. (2013). *Numerical methods in engineering with Python 3.* Cambridge university press.

3. Resnick, R., Halliday, D., Krane, K. (2004). *Física Vol. I. I.*