9. Sea $f(x) = (cosx + senx)/2$ con $x_i = i\pi/2$estimar el error al aproximar por un polinomio de grado 3 utilizando polinomios de Lagrange como también la interpolación de Newton para aproximar los valores de $x = \pi/4; \pi/2; 3\pi/4; \pi$, utilice 9 cifras significativas.

Librerias:

polinom

Ya tiene la implementación de lagrant

Ggplot2

Graficar con puntos

Sympy

Manejo de variables

Se calcula el

x

Xi

Fxi

#x   pi/4,pi/2,3pi/4,pi

#x1 0   ,pi/2,pi ,3pi/2

#fx1 1/2 ,1/2 ,-1/2 ,-½

El fxi se saca reemplazando el valor xi en fx

De polinom se usa poly.calc() que nos da el polinomio de lagrant directamente

La función recibe 2 listas de datos

El x y el fxi

Se grafican los puntos fxi y el polinomio obtenido con poly calc para ver como corta los datos

Para el error se usó wolfram comparando la exactitud de cada punto

Input interpretation:

$$-2.5 + 6.578404\,x + x^2 \times (-4.052847) + 0.6880327\,x^3 \quad \text{where } x = \frac{\pi}{4}$$

Result:

0.5

0.5

Input interpretation:

$$-2.5 + 6.578404\,x + x^2 \times (-4.052847) + 0.6880327\,x^3 \quad \text{where } x = \pi$$

Result:

−0.499999

-0.5 debería ser

Da muy cerca

Newton

Solo se usó sympy y ggplot2

Función para calculo con uso de matriz

```
require(rSymPy)
n <- length(x)
q <- matrix(data = 0, n, n)
q[,1] <- y
f <- as.character(round(q[1,1], 5))
fi <- ''

for (i in 2:n) {
  for (j in i:n) {
    q[j,i] <- (q[j,i-1] - q[j-1,i-1]) / (x[j] - x[j-i+1])
  }
  fi <- paste(fi, '*(x - ', x[i-1], ')', sep = '', collapse = '')

  f <- paste(f, ' + ', round(q[i,i], 5), fi, sep = '', collapse = '')
}

x <- Var('x')
sympy(paste('e = ', f, collapse = '', sep = ''))
approx <- sympy(paste('e.subs(x, ', as.character(x0), ')', sep = '', collapse = ''))
```

Input interpretation:

$$0.5 - 0.81057\left(\frac{\pi}{4} - 0.785398163397448\right)\left(\frac{\pi}{4} - 1.5707963267949\right) + 0.68803\left(\frac{\pi}{4} - 0.785398163397448\right)\left(\frac{\pi}{4} - 1.5707963267949\right)\left(\frac{\pi}{4} - 2.35619449019234\right)$$

Result:

0.500000…

0.5

Input interpretation:

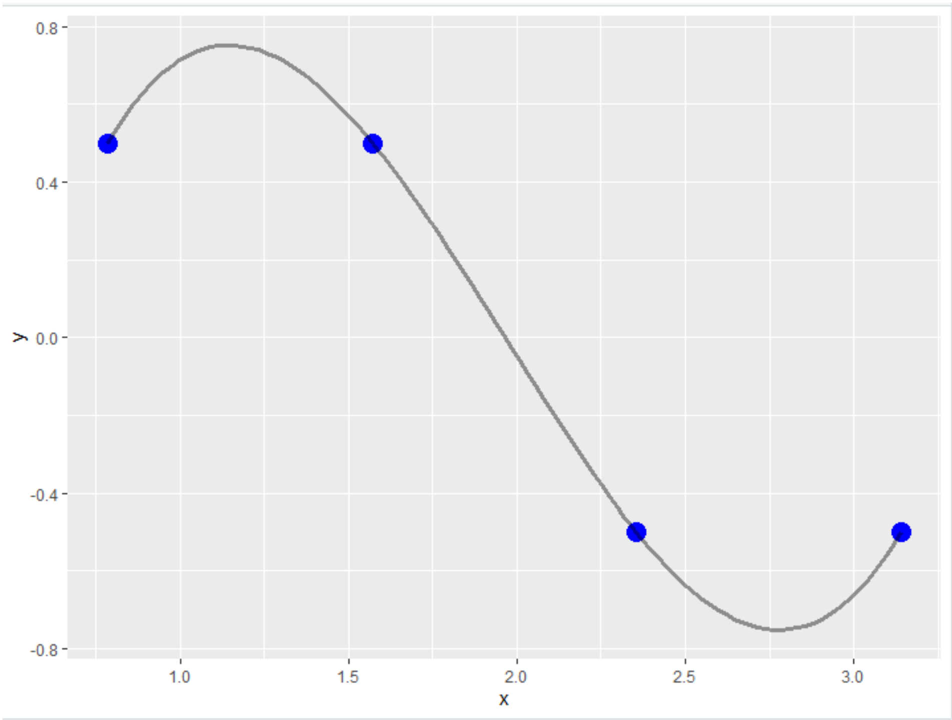
$$0.5 - 0.81057\left(3\times\frac{\pi}{4} - 0.785398163397448\right)\left(3\times\frac{\pi}{4} - 1.5707963267949\right) + 0.68803\left(3\times\frac{\pi}{4} - 0.785398163397448\right)\left(3\times\frac{\pi}{4} - 1.5707963267949\right)\left(3\times\frac{\pi}{4} - 2.35619449019234\right)$$

Result:

−0.500001…

-0.5

```
$`Tabla de calculos`
     [,1]     [,2]       [,3]      [,4]
[1,]  0.5  0.00000  0.0000000 0.0000000
[2,]  0.5  0.00000  0.0000000 0.0000000
[3,] -0.5 -1.27324 -0.8105695 0.0000000
[4,] -0.5  0.00000  0.8105695 0.6880327
```



10. Considere el comportamiento de gases no ideales se describe a menudo con la ecuación virial de estado. los siguientes datos para el nitrógeno N_2

T(K)	100	200	300	400	450	500	600
$B(cm^3)/mol$	-160	-35	-4.2	9.0		16.9	21.3

Donde T es la temperatura $[K]$ y B es el segundo coeficiente virial.
El comportamiento de gases no ideales se describe a menudo con la ecuación virial de estado

$$\frac{PV}{RT} = 1 + \frac{B}{V} + \frac{C}{V^2} + ..., \qquad (1)$$

Donde P es la presión, V el volumen molar del gas, T es la temperatura Kelvin y R es la constante de gas ideal.Los coeficientes $B = B(T)$, $C = C(T)$, son el segundo y tercer coeficiente virial, respectivamente. En la práctica se usa la serie truncada para aproximar

$$\frac{PV}{RT} = 1 + \frac{B}{V} \qquad (2)$$

Se crea un vector de temperaturas K y uno de los coeficientes viral en B

Con estos vectores se crea una matriz A que relaciona los puntos en K con su parte en B

Con la función solve se determina el polinomio interpolante entre K y B.

Description

This generic function solves the equation `a %*% x = b` for `x`, where `b` can be either a vector or a matrix.

Esta descripcion fue tomada de la pagina de documentacion de R: https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/solve

Graficamos para comparar los puntos obtenidos por los que nos daba el enunciado

Se calcula el resultado del polinomio:

Source on Save

```
temperatura = c(100,200,300,400,500,600)
Bm = c(-160, -35, -4.2, 9, 16.9, 21.3)
A = matrix(c(1, temperatura[1],temperatura[1]^2, temperatura[1]^3, temperatura[1]^4, temperatura[1]^5
             ,1, temperatura[2],temperatura[2]^2, temperatura[2]^3, temperatura[2]^4, temperatura[2]^5
             ,1, temperatura[3],temperatura[3]^2, temperatura[3]^3, temperatura[3]^4, temperatura[3]^5
             ,1, temperatura[4],temperatura[4]^2, temperatura[4]^3, temperatura[4]^4, temperatura[4]^5
             ,1, temperatura[5],temperatura[5]^2, temperatura[5]^3, temperatura[5]^4, temperatura[5]^5
             ,1, temperatura[6],temperatura[6]^2, temperatura[6]^3, temperatura[6]^4, temperatura[6]^5
),nrow = 6, ncol = 6, byrow = TRUE)
sol = solve(A,Bm)
polinomio = function(x){
  sol[1]+sol[2]*x+sol[3]*x^2+sol[4]*x^3+sol[5]*x^4+sol[6]*x^5
}
#Punto b
B = polinomio(450)
cat("El segundo coeficiente viral a 450K es",B,"\n")
#Punto c
plot(temperatura, Bm, main = "Gráfica del polinomio resultante")
curve(polinomio, add = TRUE)
#Punto d
PolinomioLagrange = poly_calc(temperatura, Bm)
cat("Polinomio de Lagrange resultante: \n")
print(PolinomioLagrange)
cat("\n")
```

Punto A

Utilizamos solve para determinar el polinomio

```
sol = solve(A,Bm)

polinomio = function(x){
  sol[1]+sol[2]*x+sol[3]*x^2+sol[4]*x^3+sol[5]*x^4+sol[6]*x^5
}
```

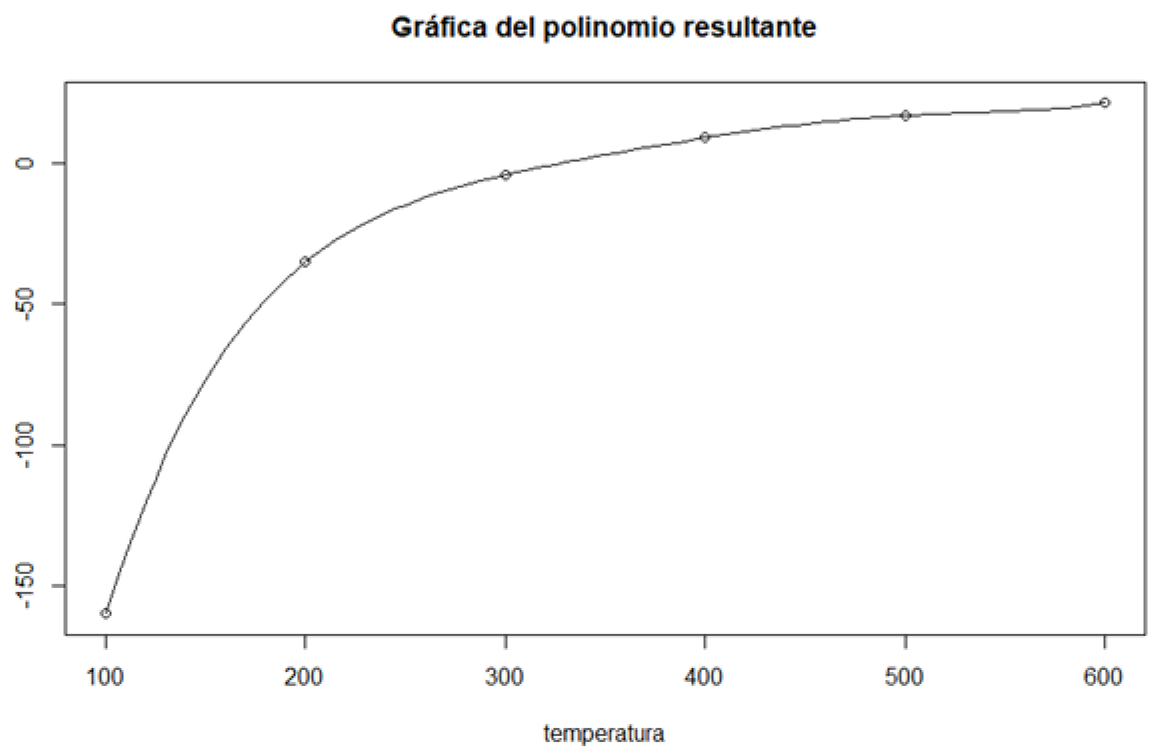
El resultado del sol es: [1] -5.739000e+02 6.635350e+00 -3.183458e-02 7.766667e-05 -9.404167e-08 4.483333e-11

Punto B

Calculamos el punto en 450, buscamos su respectivo B en el polinomio

```r
B = polinomio(450)
cat("El segundo coeficiente viral a 450K es",B,"\n")

## El segundo coeficiente viral a 450K es 13.88437
```

Graficamos (Punto C)



Punto D

Utilizamos Lagrange y vemos el polinomio interpolante

```r
PolinomioLagrange = poly_calc(temperatura, Bm)
cat("Polinomio de Lagrange resultante: \n")

## Polinomio de Lagrange resultante:

print(PolinomioLagrange)

## -573.9 + 6.63535*x - 0.03183458*x^2 + 7.766667e-05*x^3 - 9.404167e-
08*x^4
```

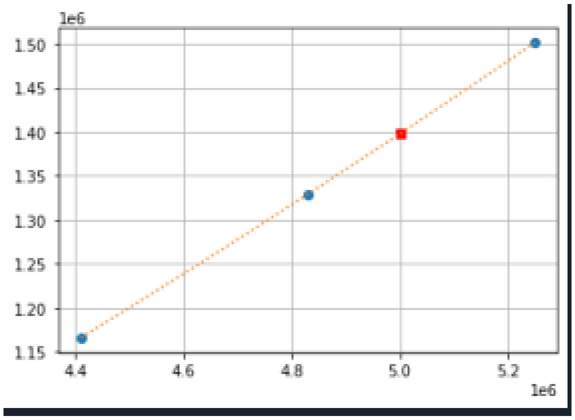
13.

Base imponible	Cuota íntegra	Tipo
4.410.000	1.165.978	38,86%
4.830.000	1.329.190	41,02%
5.250.000	1.501.474	43,18%
5.670.000	1.682.830	

Metodo de Lagrange

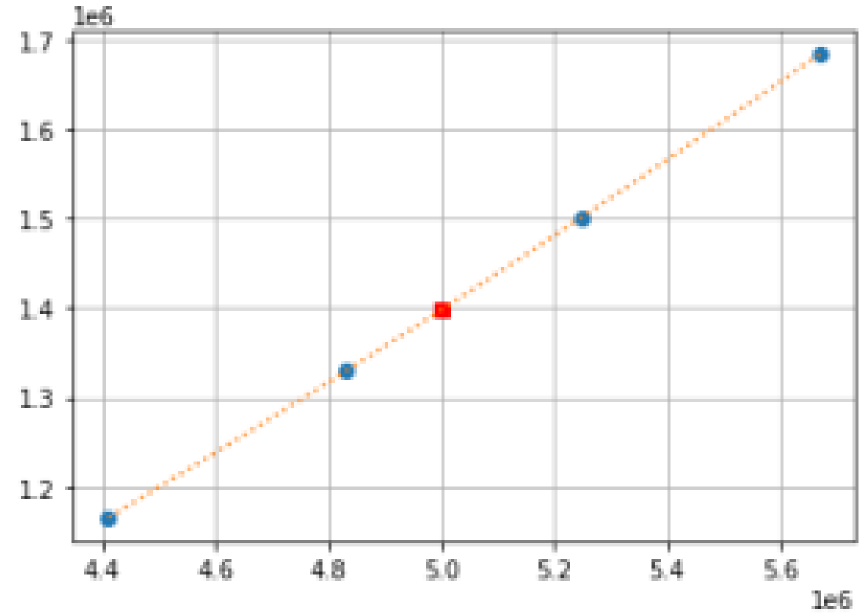
```
import numpy as np
from scipy.interpolate import lagrange
import matplotlib.pyplot as plt

x=np.array([4410000 , 4830000, 5250000])
y=np.array([1165978, 1329190, 1501474])

polinomio=lagrange(x,y)
xi=5000000
yi=polinomio(xi)
print(yi)

xs=np.linspace(x.min(),x.max())
ys=polinomio(xs)

plt.plot(x,y,'o')
plt.plot(xi,yi,'sr')
plt.plot(xs,ys,':')

plt.grid()
plt.show()
```

```
1397831.142857121
```




```python
import numpy as np
from scipy.interpolate import lagrange
import matplotlib.pyplot as plt

x=np.array([4410000 , 4830000, 5250000, 5670000])
y=np.array([1165978, 1329190, 1501474, 1682830])

polinomio=lagrange(x,y)
xi=5000000
yi=polinomio(xi)
print(yi)

xs=np.linspace(x.min(),x.max())
ys=polinomio(xs)

plt.plot(x,y,'o')
plt.plot(xi,yi,'sr')
plt.plot(xs,ys,':')

plt.grid()
plt.show()
```

```
1397831.142856656
```



Metodo de Newton

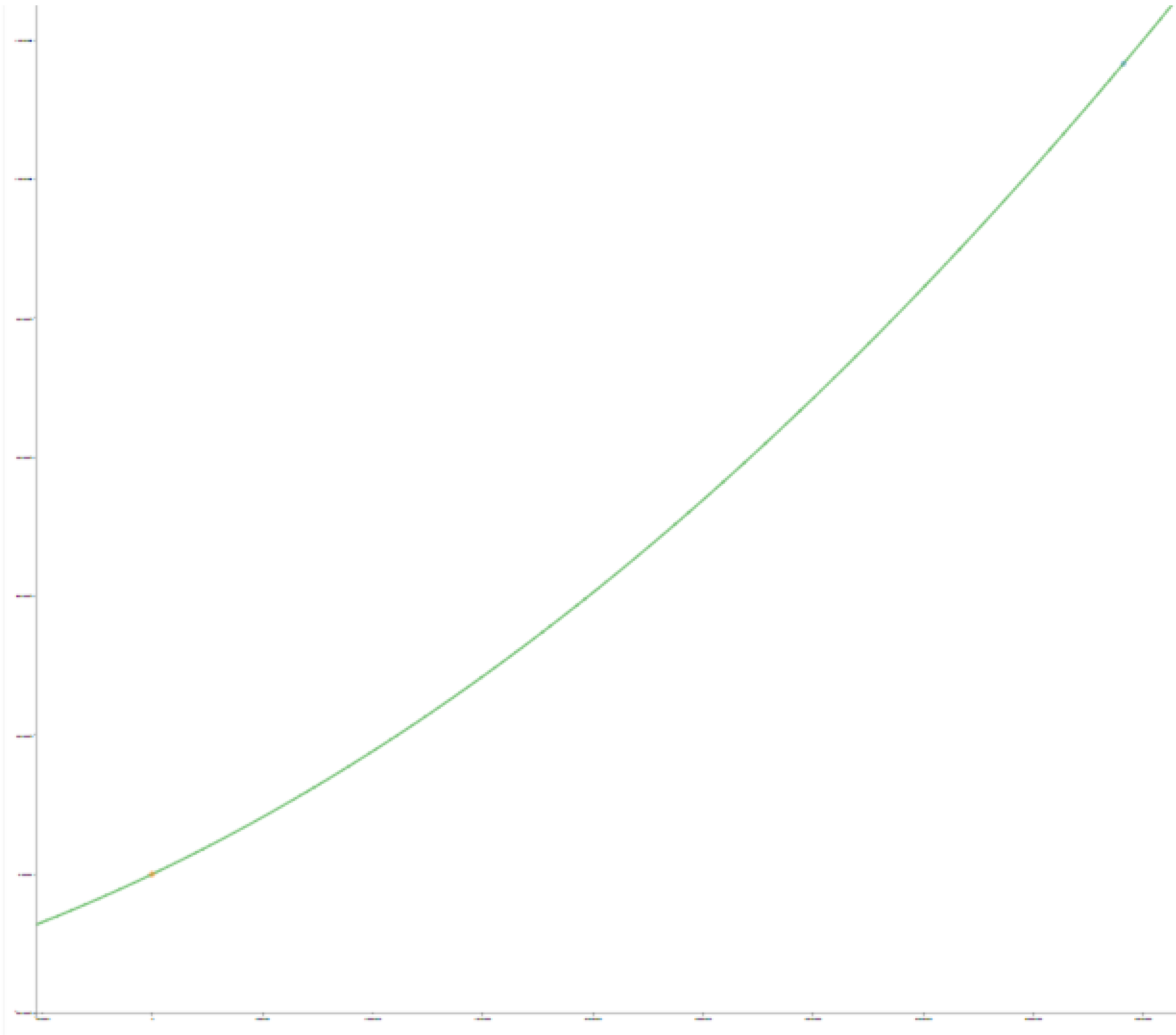
$$1165978 = a * (4410000)^2 + b * 4410000 + c$$

$$1329190 = a * (4830000)^2 + b * 4830000 + c$$

$$1501474 = a * (5250000)^2 + b * 5250000 + c$$

$$\frac{9}{350000000}x^2 + \frac{151}{1000}x - 26$$

$$1574259{,}7$$



$$1165978 = a * (4410000)^3 + b * (4410000)^2 + c * (4410000) + d$$

$$1329190 = a * (4830000)^3 + b * (4830000)^2 + c * (4830000) + d$$

$$1501474 = a * (5250000)^3 + b * (5250000)^2 + c * (5250000) + d$$

$$1682830 = a * (5670000)^3 + b * (5670000)^2 + c * (5670000) + d$$

$$x^3 - \frac{50715000}{135}x^2 + 69870300000000x + 11182657500000000000026$$