FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Entrega Teórica 2 - Análisis Numérico I

[75.12] Análisis Numérico I
Curso Tarela
Primer cuatrimestre de 2021

Alumno 1:	Argel, Ignacio
Número de padrón:	104351
Email:	iargel@fi.uba.ar
Alumno 2:	Bareiro, Facundo
Número de padrón:	103807
Email:	fbareiro@fi.uba.ar

Índice

1. Introducción

El siguiente informe busca analizar el método de ajuste continuo por cuadrados mínimos donde se aproximará a una función $f(x)$ mediante un polinomio $P(x)$, se controlará qué tan cercana es esa aproximación y a su vez se analizará el error de la misma para las distintas dimensiones N del polinomio.
Observación: las secciones 2.1 y 2.2 son propias de la ET1, sin embargo nos pareció acorde nombrarlas en este informe para que se logre entender de una mejor manera el método desarrollado.
Para este se utilizo Python como lenguaje de programación y el siguiente conjunto de librerías:

- NumPy : Para trabajar le metodo SOR en general.
- SimPy: Para calcular las integrales en el termino independiente y graficar las funciones.

2. Desarrollo

2.1. Obtener Matriz de Hilbert

Para llegar a la matriz de Hilbert se tuvo en cuenta la expresión del enunciado:

$$H_{j+1,k+1} = (\frac{b^{j+k+1}-a^{j+k+1}}{j+k+1}) \text{ con } j,k = 0,...,N$$

mediante Python con dos bucles for y respaldados por la expresión de arriba se llegó a la siguiente matriz de 10x10.
Observación: Redujimos los dígitos mostrados en pantalla para permitir la visualización. La matriz mostrada es la matriz mayor, y cada caso menor es una submatriz de esta. El cálculo está hecho sobre los 16 decimales por defecto de simple precisión en Python.

$$\begin{bmatrix}
1{,}088 & 0{,}591 & 0{,}429 & 0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 \\
0{,}591 & 0{,}429 & 0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 \\
0{,}429 & 0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 \\
0{,}350 & 0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 \\
0{,}305 & 0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 \\
0{,}276 & 0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 \\
0{,}257 & 0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 \\
0{,}245 & 0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 & 0{,}247 \\
0{,}237 & 0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 & 0{,}247 & 0{,}253 \\
0{,}232 & 0{,}230 & 0{,}229 & 0{,}230 & 0{,}232 & 0{,}236 & 0{,}241 & 0{,}247 & 0{,}253 & 0{,}261
\end{bmatrix}$$

2.2. Obtención del término independiente

En este caso se presentaron dos tipos de formas de hallar los valores del término independiente. La primera sería llegar mediante la resolución analítica de la integral a una expresión que dependa de las constantes a y b; La segunda, utilizada finalmente para la resolución es mediante la librería SimPy resolver la integral en cuestión sin ningún paso intermedio.

$$c_{j+1} = \int_a^b -4x(x-1)x^j dx$$

Finalmente se obtuvo el siguiente vector:

$$\begin{bmatrix} 0{,}650243217488076 \\ 0{,}315938924200197 \\ 0{,}181570231662825 \\ 0{,}113799256785772 \\ 0{,}0745258771503912 \\ 0{,}0494591342881323 \\ 0{,}0322441905633196 \\ 0{,}0197003950628422 \\ 0{,}0100896502469084 \\ 0{,}00239487716254494 \end{bmatrix}$$

2.3. Marco Teórico

Para poder aproximar funciones con polinomios se debe utilizar el ajuste continuo por cuadrados mínimos, donde se plantea en principio que el error:

$$E = \int_a^b e^2 \, dx$$

sea mínimo, o bien que :

$$E = \int_a^b [f(x) - P_n]^2 \, dx$$

sea mínimo. Siendo $P_n(x) = \sum_{k=0}^{n} a_k x^k$ resulta:

$$E = \int_a^b [f(x) - \sum_{k=0}^{n} a_k x^k]^2 \, dx$$

y como se puede observar en la expresión, E depende solo de los ak; por lo tanto se debe plantear que:

$$\frac{\partial E}{\partial a_j} = 0 \qquad \text{con } j = 0, ..., n$$

Desarrollando esta derivada, finalmente queda expresada la ecuación normal para el ajuste continuo como:

$$\sum_{k=0}^{n} a_k \int_a^b x^{k+j} \, dx = \int_a^b f(x) x^j \, dx \qquad \text{con } j = 0, ..., n$$

que analizando en detalle se puede ver la siguiente equivalencia: $\int_a^b x^{k+j} \, dx$ es la matriz de Hilbert que se expresa como:

$$H_{j+1,k+1} = \frac{b^{j+k+1} - a^{j+k+1}}{j+k+1}$$

Por otro lado, se nombra a $\int_a^b f(x) x^j \, dx$ como r_{j+1} Y por lo tanto para averiguar las a_k, debe resolverse el siguiente sistema de ecuaciones lineales:

$$\mathbf{H}a = r$$

Que puede ser resuelto mediante cualquier técnica de resolución de sistemas de ecuaciones lineales. En el presente informe se utilizó el método SOR para llegar a la solución con una tolerancia de 10^{-4} y un $\omega_{optimo} = 1,64$ el cual fue calculado en la ET1. Una vez obtenidas las a_k, se las reemplaza en la expresión del polinomio que es:

$$P_{N-1}(x) = \sum_{k=0}^{N-1} a_k x^k$$

2.4. Polinomios

Los siguientes polinomios tienen un truncamiento a fin de mostrarse en pantalla de manera correcta. Los cálculos se hicieron con 18 dígitos.

Para $N = 4$:

$$P_3 = 0{,}0459369x^3 - 4{,}0820703x^2 + 4{,}040793x - 0{,}004612$$

Para $N = 5$:

$$P_4 = -1{,}554584x^4 + 3{,}3799198x^3 - 6{,}350823x^2 + 4{,}557759x - 0{,}0283896$$

Para $N = 6$:

$$P_5 = 0{,}8301467x^5 - 2{,}1025039x^4 + 1{,}794878x^3 - 4{,}564356x^2 + 4{,}036644x + 0{,}003149$$

Para $N = 7$:

$$P_6 = -0{,}5286731x^6 + 1{,}3917792x^5 - 2{,}070516x^4 + 2{,}260004x^3 - 5{,}357846x^2 + 4{,}326986x - 0{,}0178281$$

Para $N = 8$:

$$P_7 = 0{,}456086x^7 - 0{,}761213x^6 + 0{,}990313x^5 - 2{,}029976x^4 + 1{,}972677x^3 - 4{,}681108x^2 + 4{,}044084x + 0{,}004380$$

Para $N = 9$:

$$P_8 = -0{,}16912x^8 + 0{,}56189x^7 - 0{,}69301x^6 + 1{,}07038x^5 - 2{,}05010x^4 + 1{,}98930x^3 - 4{,}82968x^2 + 4{,}12191x - 0{,}00250$$

Para $N = 10$:

$$P_9 = 0{,}1846x^9 - 0{,}2949x^8 + 0{,}4801x^7 - 0{,}7738x^6 + 1{,}0951x^5 - 2{,}0539x^4 + 2{,}1491x^3 - 4{,}9085x^2 + 4{,}1183x - 0{,}0005$$

3. Comparativa N=4

El siguiente grafico es la comparación entre la funcion $f(x) = -4x(x-1)$ en azul y el polinomio $P_3 = \sum_0^3 x_j x^j$ en rojo.

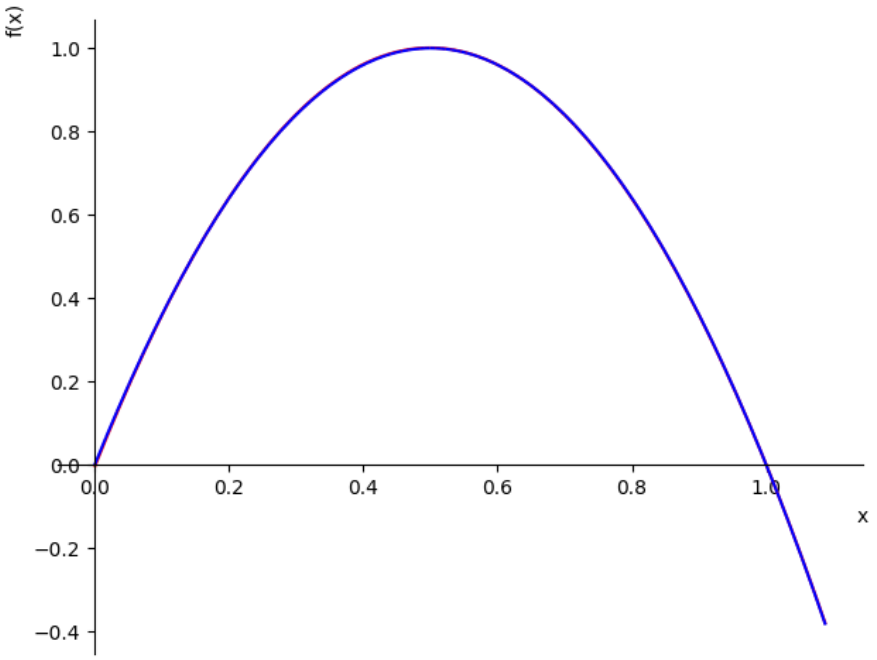

Figura 1: Grafica comparativa en N=4 de $f(x)$ contra P_3 .

Este grafico muestra el error de P_3 en comparacion con $f(x)$.

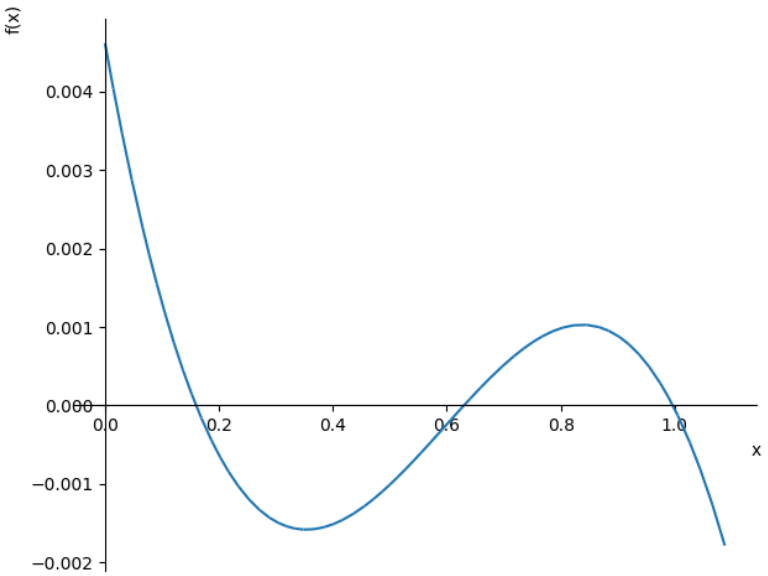

Figura 2: Grafica error P_3 .

4. Comparativa N=5

El siguiente grafico es la comparación entre la funcion $f(x) = -4x(x-1)$ en azul y el polinomio $P_4 = \sum_0^4 x_j x^j$ en rojo.

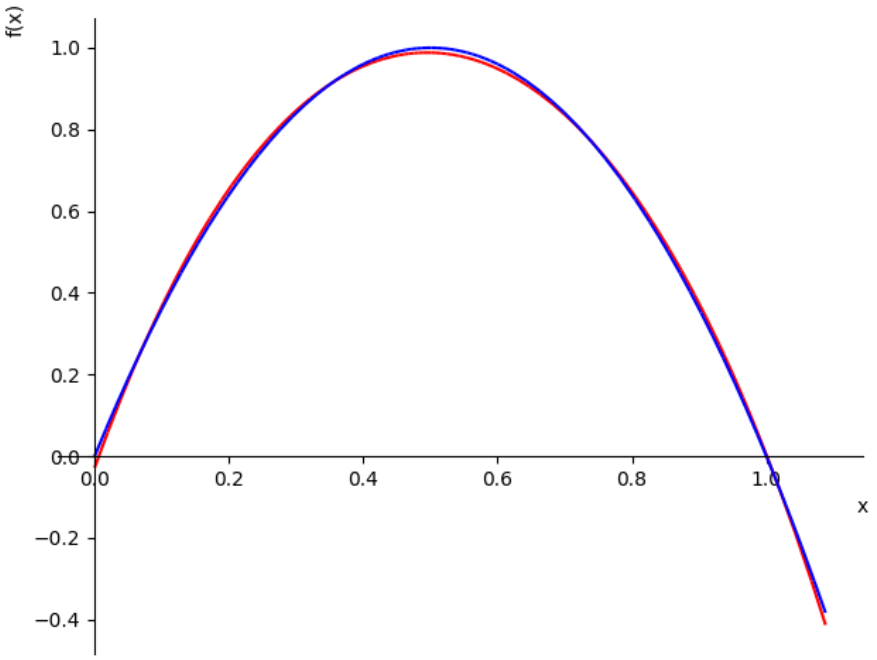

Figura 3: Grafica comparativa en N=5 de $f(x)$ contra P_4 .

Este grafico muestra el error de P_4 en comparacion con $f(x)$.

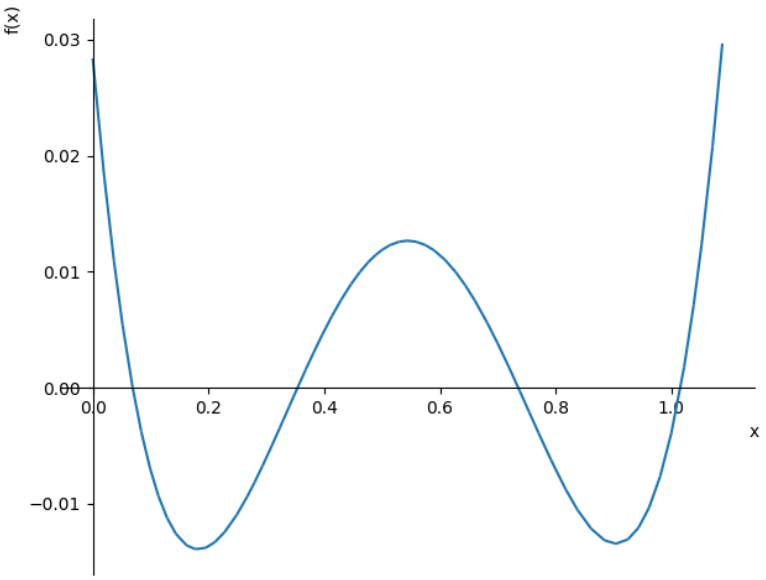

Figura 4: Grafica error P_4 .

5. Comparativa N=6

El siguiente grafico es la comparación entre la funcion $f(x) = -4x(x-1)$ en azul y el polinomio $P_5 = \sum_0^5 x_j x^j$ en rojo.

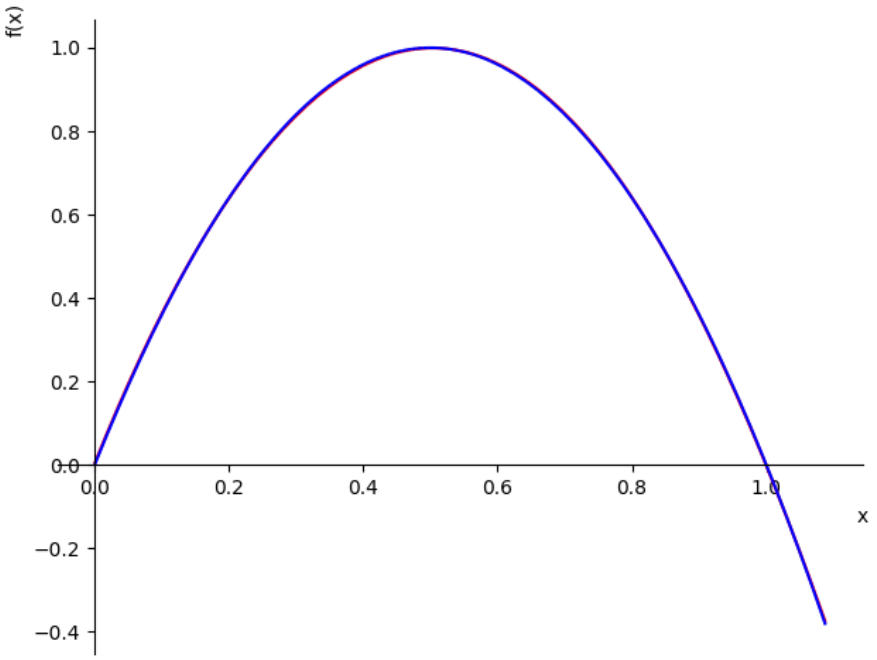

Figura 5: Grafica comparativa en N=6 de $f(x)$ contra P_5 .

Este grafico muestra el error de P_5 en comparacion con $f(x)$.

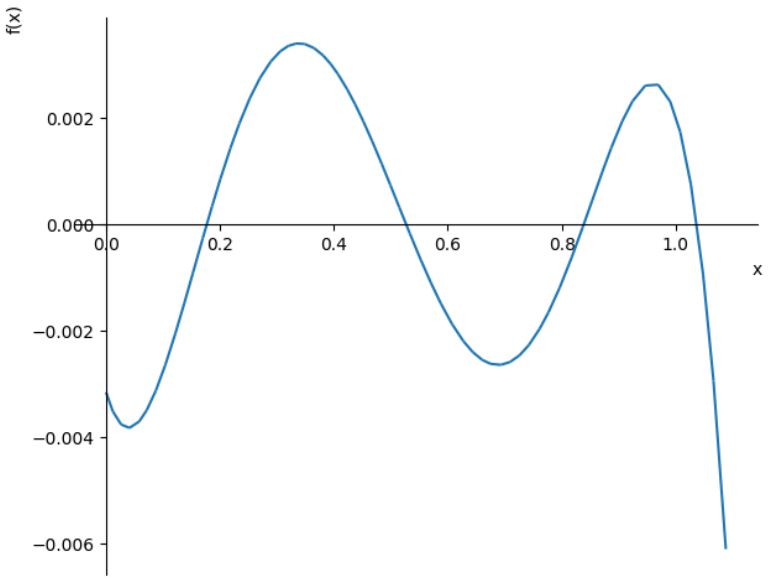

Figura 6: Grafica error P_5 .

6. Comparativa N=7

El siguiente grafico es la comparación entre la funcion $f(x) = -4x(x-1)$ en azul y el polinomio $P_6 = \sum_0^6 x_j x^j$ en rojo.

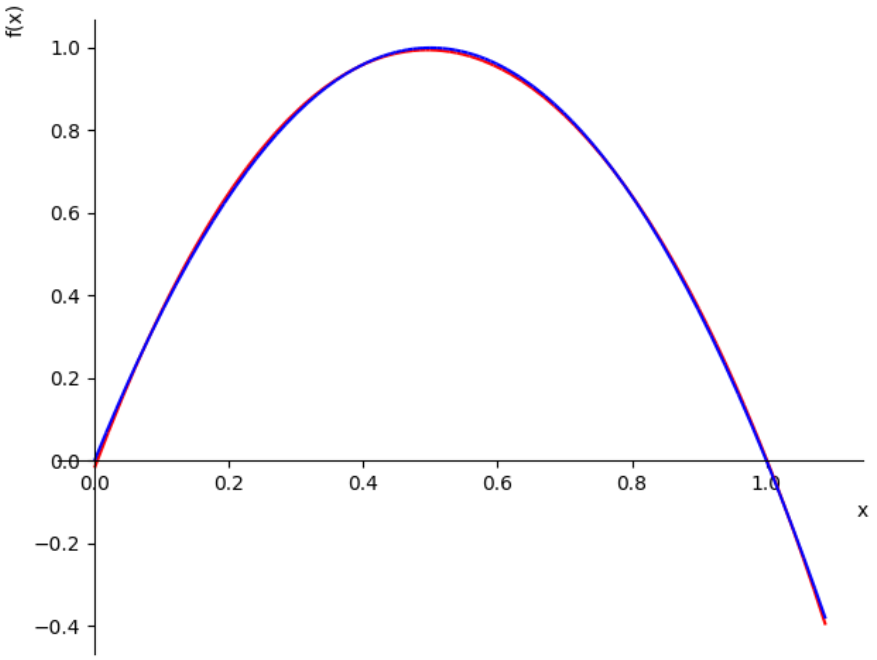

Figura 7: Grafica comparativa en N=7 de $f(x)$ contra P_6 .

Este grafico muestra el error de P_6 en comparacion con $f(x)$.

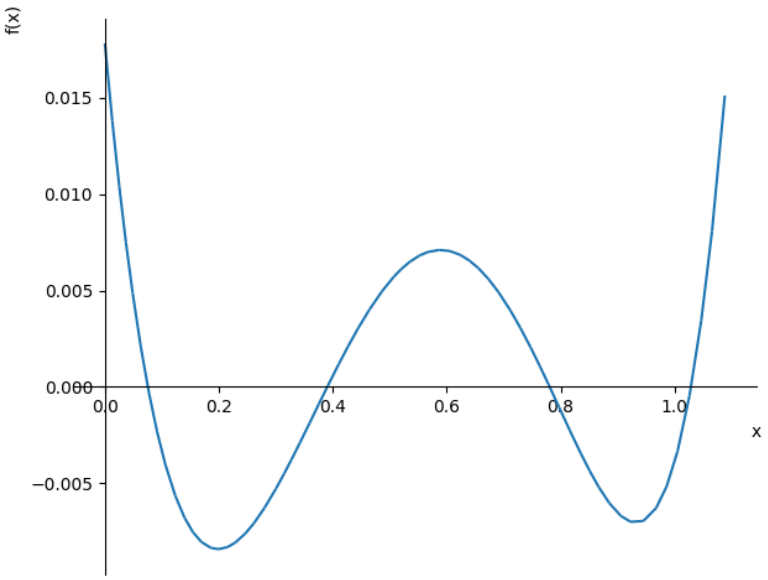

Figura 8: Grafica error P_6 .

7. Comparativa N=8

El siguiente grafico es la comparación entre la funcion $f(x) = -4x(x-1)$ en azul y el polinomio $P_7 = \sum_0^7 x_j x^j$ en rojo.

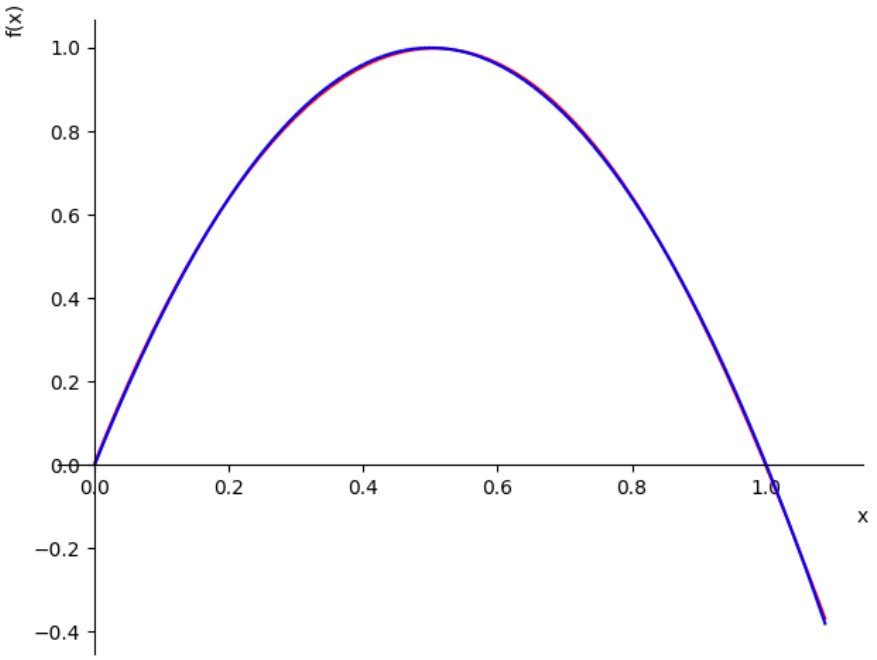

Figura 9: Grafica comparativa en N=8 de $f(x)$ contra P_7 .

Este grafico muestra el error de P_7 en comparacion con $f(x)$.

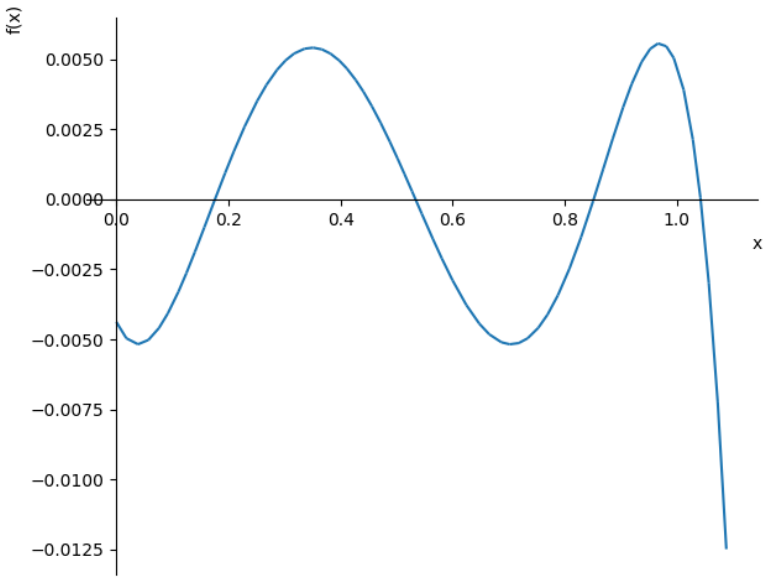

Figura 10: Grafica error P_7 .

8. Comparativa N=9

El siguiente grafico es la comparación entre la funcion $f(x) = -4x(x-1)$ en azul y el polinomio $P_8 = \sum_0^8 x_j x^j$ en rojo.

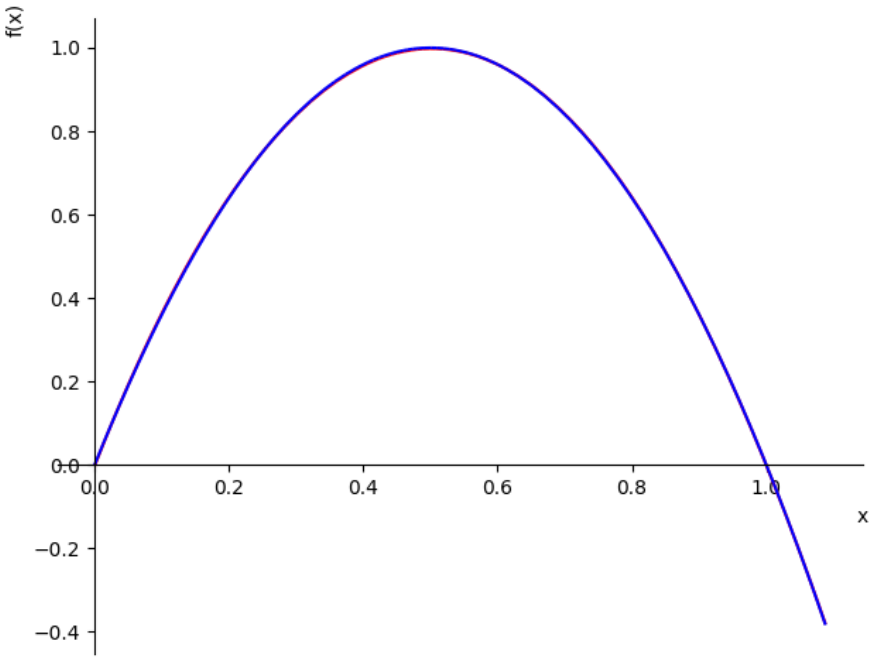

Figura 11: Grafica comparativa en N=9 de $f(x)$ contra P_8 .

Este grafico muestra el error de P_8 en comparacion con $f(x)$.

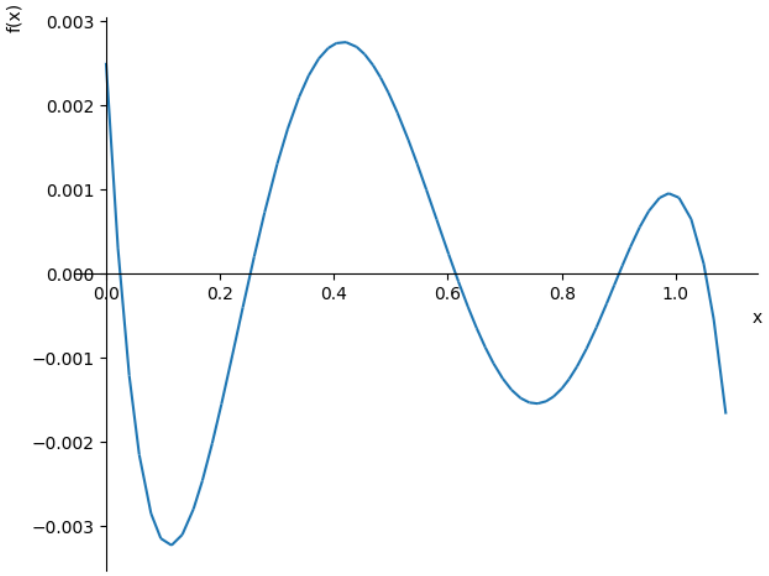

Figura 12: Grafica error P_8 .

9. Comparativa N=10

El siguiente grafico es la comparación entre la funcion $f(x) = -4x(x-1)$ en azul y el polinomio $P_9 = \sum_0^9 x_j x^j$ en rojo.

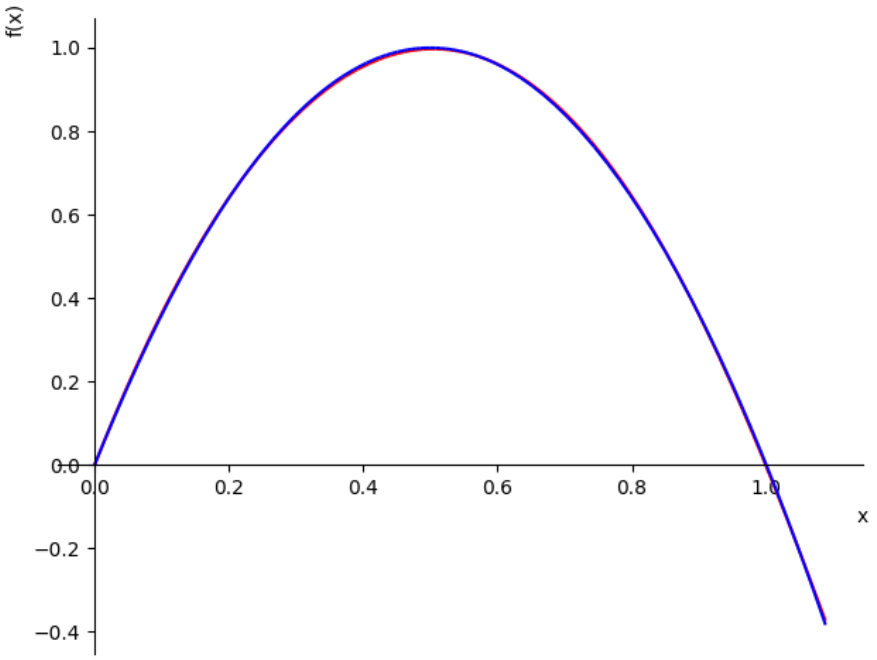

Figura 13: Grafica comparativa en N=10 de $f(x)$ contra P_9 .

Este grafico muestra el error de P_9 en comparacion con $f(x)$.

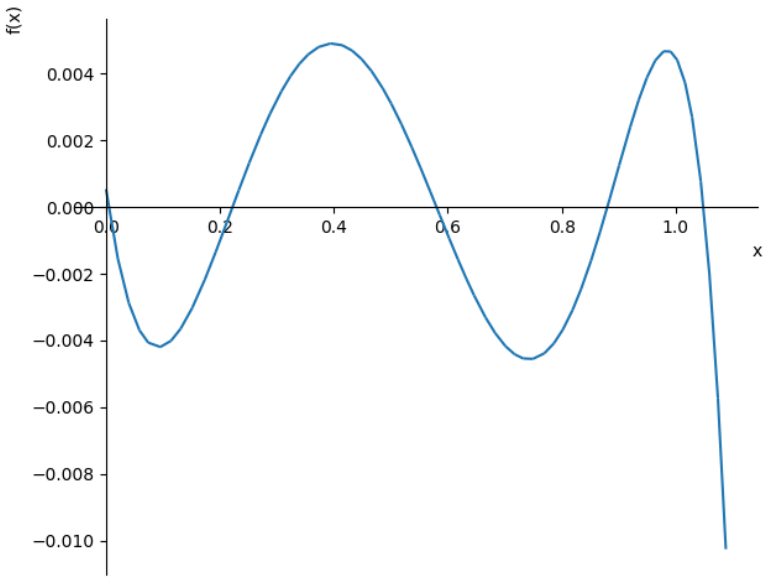

Figura 14: Grafica error P_9 .

10. Cálculo del error

A fin de respaldar con más datos las conclusiones se calculó el error como:

$$E = \int_a^b [f(x) - P_n(x)]^2 \, dx$$

para observar la bondad del ajuste en cada N. Los resultados fueron los siguientes:

$$N = 4 : 1{,}80847978187683 \ 10^{-6}$$
$$N = 5 : 0{,}000118147560826731$$
$$N = 6 : 5{,}82648243356077 \ 10^{-6}$$
$$N = 7 : 3{,}80289088930397 \ 10^{-5}$$
$$N = 8 : 1{,}72105937238771 \ 10^{-5}$$
$$N = 9 : 3{,}04881436669069 \ 10^{-6}$$
$$N = 10 : 1{,}24399451242319 \ 10^{-5}$$

11. Conclusión

En primera instancia, al analizar las gráficas comparativas, podemos ver que a simple vista los polinomios se aproximan bien a la función original en este intervalo $[a, b]$. A su vez en las gráficas del error podemos observar que el mismo toma valores cercanos al 0, lo que es otro indicio de que las aproximaciones son correctas. Para asegurarnos que la aproximación era buena, por último se calculó el error como en la sección 10 y con estos resultados se puede asegurar que el ajuste polinomial es bueno ya que el máximo valor que toma el error es del orden de 10^{-4}.
En los resultados que obtuvimos no se observó ninguna tendencia en la relación entre el aumento de N y el error, por lo que podríamos decir que en nuestro caso, la bondad del ajuste no depende del tamaño que tenga el polinomio. Por último, al analizar nuestros resultados y al ver que los polinomios encontrados para los distintos N aproximaban muy bien a la función consideramos que no era necesario utilizar otro método para hallar el polinomio (como por ejemplo la aproximación discreta).

12. Codigo

El codigo adjuntado toma sorHilbert de la entrega teorica 1.

```
array = []
for N in range(4,11):
    semilla = np.zeros(N)
    array.append(sorHilbert(N,1.64,semilla))

polinomios = []
for i in range(0,7):
    act = array[i]
    x=Symbol('x')
    polinomio = 0
    for j in range(len(act)):
        polinomio += act[j]*x**j
    polinomios.append(polinomio)
    print(polinomio)

for i in range(0,7):

    x = symbols('x')
    px = polinomios[i]
    fx = -4*x*(x-1)

```

```
22        p1 = plot(px , fx , (x, 0, 0.05 + 103807*(10**(-5))),line_color = 'blue',
              show = False)
23        p1[0].line_color = 'red'
24        p1.show()
25
26    for i in range(0,7):
27
28        x = symbols('x')
29        px = polinomios[i]
30        fx = -4*x*(x-1)
31
32        p1 = plot(fx-px , (x, 0, 0.05 + 103807*(10**(-5))), show = False)
33        p1.show()
```