

Tarea 8-Métodos numéricos

1st Daniel Vallejo Aldana
Maestría en Ciencias de la Computación
Centro de Investigación en Matemáticas
daniel.vallejo@cimat.mx

Resumen—En este trabajo se implementan y evalúan los métodos de interpolación de mínimos cuadrados, interpolación polinomial, interpolación por polinomios de Lagrange e interpolación por polinomios de Newton.

Index Terms—Interpolación, Mínimos Cuadrados, Interpolación Polinomial, Polinomio de Lagrange, Polinomio de Newton

I. INTRODUCCIÓN

Los métodos de interpolación son ampliamente usados en el campo de ciencias de la computación cuando se quiere estimar el valor de una función f en un rango de puntos $[a, b]$ pero no se tiene una expresión analítica de f y solo se conoce el valor de f en ciertos puntos x_0, \dots, x_m . En el presente trabajo se implementan y evalúan cuatro métodos diferentes, el primero de ellos es el de mínimos cuadrados, el segundo es el de interpolación polinomial, el tercero utiliza los polinomios de Lagrange y el último utiliza los polinomios de Newton.

II. MÉTODO/ALGORITMO

Dividiremos la sección de metodología en cuatro partes, la primera referente a la parte de mínimos cuadrados, la segunda acerca de la interpolación polinomial y la tercera y la cuarta se refieren a los polinomios de Lagrange y de Newton respectivamente.

II-A. Mínimos Cuadrados

Consideremos el siguiente problema

$$\operatorname{argmin}_x \|A\mathbf{x} - \mathbf{b}\|^2$$

Donde $A \in \operatorname{Mat}_{n \times m+1}(\mathbb{R})$ con n los puntos usados para interpolar y m es el grado del polinomio o el número de funciones usadas para interpolar. Notemos que podemos escribir $\|A\mathbf{x} - \mathbf{b}\|^2$ de la siguiente forma

$$\begin{aligned}\|A\mathbf{x} - \mathbf{b}\|^2 &= (A\mathbf{x} - \mathbf{b})^T A\mathbf{x} - \mathbf{b} \\ &= (\mathbf{x}^T A^T - \mathbf{b}^T)(A\mathbf{x} - \mathbf{b}) \\ &= \mathbf{x}^T A^T A\mathbf{x} - 2\mathbf{x}^T A^T \mathbf{b} + \mathbf{b}^T \mathbf{b}\end{aligned}$$

Como queremos minimizar lo anterior necesitamos encontrar los \mathbf{x} para los cuales el gradiente de la norma se convierta en cero, calculando el gradiente de dicha expresión obtenemos

$$\nabla_{\mathbf{x}} \frac{\|A\mathbf{x} - \mathbf{b}\|^2}{2} = A^T A\mathbf{x} - A^T \mathbf{b}$$

Luego, haciendo el gradiente igual a 0 vemos que el vector \mathbf{x} que minimiza la norma al cuadrado de la diferencia de la ecuación $A\mathbf{x} = \mathbf{b}$ es

$$\mathbf{x} = (A^T A)^{-1} A^T \mathbf{b}$$

Que es la conocida fórmula para mínimos cuadrados. En el presente trabajo se resuelve el sistema de ecuaciones $A^T A\mathbf{x} = A^T \mathbf{b}$. Notemos que en este caso $A^T A$ es una matriz simétrica, cuadrada, no singular, por lo que es posible encontrar una solución al sistema de ecuaciones planteado.

Para la construcción de la matriz A , se consideraron dos enfoques diferentes. El primero de ellos es mediante el ajuste de un polinomio de grado m a los n puntos usados para la interpolación. En este caso la matriz A queda definida de la siguiente manera.

$$A = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix}$$

Donde m es el grado del polinomio y n son los puntos necesarios para interpolar. En este caso vemos que n debe ser mayor o igual a $m + 1$ para que el sistema $A^T A\mathbf{x} = A^T \mathbf{b}$ tenga una única solución y que la interpolación pase por los puntos deseados.

En este caso

$$\mathbf{b} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}$$

Con f la función que se desea interpolar y

$$\mathbf{x} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}$$

Donde \mathbf{x} corresponde al vector de coeficientes del polinomio o de la combinación de funciones que se ajusta a los puntos de interpolación.

La segunda forma de construir la matriz A es utilizando funciones g_i . En este caso la matriz A queda definida de la siguiente manera

$$A = \begin{bmatrix} g_1(x_0) & g_2(x_0) & \dots & g_m(x_0) \\ \vdots & \vdots & \vdots & \vdots \\ g_1(x_n) & g_2(x_n) & \dots & g_m(x_n) \end{bmatrix}$$

En este caso $m = 3$ fue el número de funciones usadas para interpolar a la función f

El pseudocódigo para el algoritmo de mínimos cuadrados donde lo que se busca ajustar es un polinomio es el siguiente

Algorithm 1 Mínimos-Cuadrados-Polinomio

Require: Puntos x_0, \dots, x_n y $f(x_0), \dots, f(x_n)$ con $n > (m + 1)$
Ensure: Coeficientes a_0, \dots, a_m con m el grado del polinomio.
 Inicializa $A_{n \times (m+1)}(0)$
for i from 1 to n **do**
 for j from 0 to m **do**
 $A[i][j] = x_i^j$
 end for
end for
 $\text{Solve}(A^T A \mathbf{x} = A^T \mathbf{b})$ donde $\mathbf{b} = [f(x_0), \dots, f(x_n)]^T$
return \mathbf{x}

En este caso la función *Solve* corresponde al método *QR* pero puede sustituirse por algún otro algoritmo de factorización de matrices.

El algoritmo para el ajuste de mínimos cuadrados usando las funciones g_i se describe a continuación.

Algorithm 2 Mínimos-Cuadrados-Funciones

Require: Puntos x_0, \dots, x_n y $f(x_0), \dots, f(x_n)$ con $n > (m + 1)$, g_1, \dots, g_m
Ensure: Coeficientes a_0, \dots, a_m con m el grado del polinomio.
 Inicializa $A_{n \times m}(0)$
for i from 1 to n **do**
 for j from 1 to m **do**
 $A[i][j] = g_j(x_i)$
 end for
end for
 $\text{Solve}(A^T A \mathbf{x} = A^T \mathbf{b})$ donde $\mathbf{b} = [f(x_0), \dots, f(x_n)]^T$
return \mathbf{x}

Vemos que para interpolar un valor z , necesitamos considerar

$$\hat{f}(z) = \sum_{i=1}^m a_i g_i(z)$$

II-B. Interpolación Polinomial

La interpolación polinomial busca ajustar un polinomio de la forma $P_m(x) = \sum_{i=0}^m a_i x^i$ a los puntos de interpolación. Lo anterior nos genera el siguiente sistema de ecuaciones

$$\begin{bmatrix} 1 & x_0 & \dots & x_0^m \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_m & \dots & x_m^m \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} f(x_0) \\ \vdots \\ f(x_m) \end{bmatrix}$$

Dicho sistema de ecuaciones se resuelve para encontrar el polinomio que mejor ajuste los puntos de interpolación.

El pseudocódigo para el método de interpolación polinomial es el siguiente

Algorithm 3 Interpolación-Polinomial

Require: Puntos x_0, \dots, x_m y $f(x_0), \dots, f(x_m)$
Ensure: Coeficientes a_0, \dots, a_m con m el grado del polinomio.
 Inicializa $A_{m+1 \times m+1}(0)$
for i from 0 to m **do**
 for j from 0 to m **do**
 $A[i][j] = x_i^j$
 end for
end for
 $\text{Solve}(A \mathbf{x} = \mathbf{b})$ donde $\mathbf{b} = [f(x_0), \dots, f(x_m)]^T$
return \mathbf{x}

Vemos que para interpolar un valor z , necesitamos considerar

$$\hat{f}(z) = \sum_{i=0}^m a_i z^i$$

II-C. Polinomios de Lagrange

El problema de los polinomios de Lagrange surge de definir un polinomio de primer grado que pasa por dos puntos distintos (x_0, y_0) y (x_1, y_1) donde $y_0 = f(x_0)$ y $y_1 = f(x_1)$ respectivamente [1]

Los coeficientes del polinomio de interpolación son de la forma

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}$$

$$L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

Por lo que el polinomio P de grado 1 para x es de la forma

$$P(x) = L_0(x)f(x_0) + L_1(x)f(x_1)$$

Notemos que $L_0(x_1) = 0$ y $L_1(x_0) = 0$ y $L_0(x_0) = 1, L_1(x_1) = 1$, de lo anterior vemos que los coeficientes L_i cumplen lo siguiente. [1]

$$L_i(x_i) = \delta_i$$

Donde δ_i corresponde a la delta de Kronecker. Podemos extender lo anterior a un grado m que cumpla las condiciones descritas anteriormente. La formula para los coeficientes del polinomio de Lagrange de grado m son de la forma

$$L_{m,i}(x) = \prod_{j=0, j \neq i}^m \frac{x - x_j}{x_i - x_j}$$

De forma que el polinomio de lagrange queda de la forma

$$P_m(x) = \sum_{i=0}^m L_{m,i}(x)f(x_i)$$

El algoritmo para interpolar usando el polinomio de Lagrange y m puntos es el siguiente. Dividiremos en dos partes el algoritmo, la primera se basa en calcular solamente los coeficientes del polinomio de Lagrange y la segunda es la forma en la que interpolamos para un punto p de la discretización que hagamos del intervalo a interpolar.

Algorithm 4 Calcula-Coeffs-Lagrange

Require: p, x_1, \dots, x_m y $f(x_1), \dots, f(x_m)$ donde p es el punto a interpolar

Ensure: $L_{1,m}, \dots, L_{m,m-1}, L_{m,m}$ Los coeficientes del polinomio de Lagrange

Inicializa $C[m]$

for i from 1 to m **do**

$$C[i] = \prod_{j=0, j \neq i}^m \frac{p - x_j}{x_i - x_j}$$

end for

return C

Donde CCL representa la función de calcular los coeficientes de Lagrange descrita en el algoritmo 4.

II-C1. Método de las diferencias divididas (Método de Newton): Los métodos de diferencias divididas se usan para generar los polinomios por si mismos. La diferencias divididas de f respecto a x_0, x_1, \dots, x_n se usan para expresar el polinomio $P_n(x)$ de la siguiente forma

$$P_n(x) = a_0 + a_1(x - x_0) + \dots + a_n \prod_{i=0}^{n-1} (x - x_i)$$

Para constantes a_0, \dots, a_n . [1]

Algorithm 5 Interpola-un-punto-usando-Lagrange

Require: p, x_1, \dots, x_m y $f(x_1), \dots, f(x_m)$ donde p es el punto a interpolar

Ensure: $\hat{f}(p)$

$$C = CCL(p, x_1, \dots, x_m, f(x_1), \dots, f(x_m))$$

$\hat{v} = 0$

for i from 1 to m **do**

$$\hat{v} = \hat{v} + C[i]f(x_i)$$

end for

return \hat{v}

Consideremos la diferencia dividida 0 como lo siguiente

$$f[x_i] = f(x_i)$$

Luego definimos la primera diferencia dividida de la forma

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$$

Definimos la k -esima diferencia dividida de la forma

$$f[x_i, \dots, x_{i+k}] = \frac{f[x_{i+1}, \dots, x_{i+k}] - f[x_i, \dots, x_{i+k-1}]}{x_{i+k} - x_i}$$

El proceso termina con la última diferencia dividida definida de la siguiente forma

$$f[x_0, \dots, x_m] = \frac{f[x_1, \dots, x_m] - f[x_0, \dots, x_{m-1}]}{x_m - x_0}$$

Por lo que el polinomio de Newton para interpolar un punto p está dado por la siguiente forma

$$P_m(p) = f[x_0] + \sum_{i=1}^m f[x_0, \dots, x_i] \prod_{j=0}^{i-1} (p - x_j)$$

El pseudocódigo para el algoritmo de diferencias divididas se presenta a continuación.

Algorithm 6 Newton-Coeffs

Require: x_0, \dots, x_m y $f(x_0), \dots, f(x_m)$

Ensure: $F_{0,0}, \dots, F_{m,m}$ los coeficientes el polinomio de Newton.

Inicializa $F \in Mat_{m+1 \times m+1}(\mathbb{R})$

Set $F_{i,0} = f(x_i) \forall i \in \{0, \dots, m\}$

for i from 1 to m **do**

for j from 1 to i **do**

$$F_{i,j} = \frac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$$

end for

end for

return $F_{i,i} \forall i \in \{0, \dots, m\}$

Por lo tanto, para interpolar un punto p tenemos la

siguiente expresión.

$$P_m(p) = F_{0,0} + \sum_{i=1}^m F_{i,i} \Pi_{j=0}^{i-1} (p - x_j)$$

III. RESULTADOS

Para probar nuestros algoritmos de interpolación consideramos la función

$$f(x) = \sin(x) + \sqrt{\exp(x)} + x$$

en el intervalo $[-6, 6]$, en la figura 1 podemos ver la gráfica de la función real

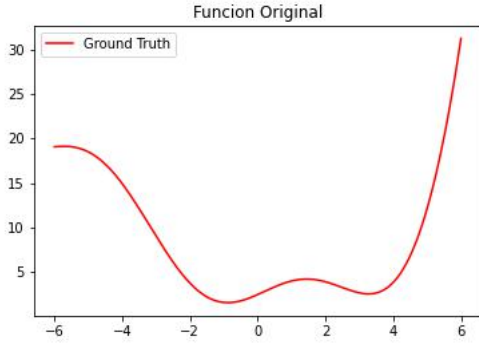


Figura 1. Gráfica de la función de prueba

Notemos que para generar los puntos x_i para interpolar, consideramos un conjunto de $m + 1$ puntos equiespaciados en el intervalo $[a, b]$ donde

$$x_i = a + i \frac{b - a}{m}$$

III-A. Ajuste con mínimos cuadrados usando un polinomio de grado 1

En este caso se busca un polinomio de la forma $ax + b$ que ajuste a la función f , en la figura 2 podemos ver el ajuste de la función con un polinomio de grado 1 mediante mínimos cuadrados.

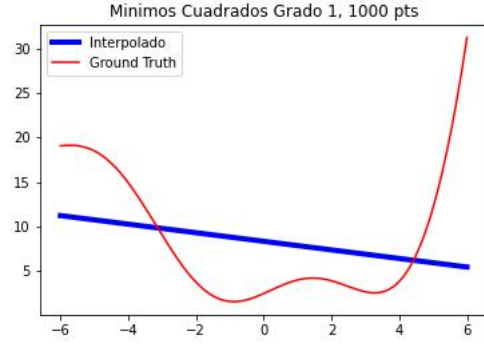


Figura 2. Ajuste con mínimos cuadrados y un polinomio de grado 1

Los 1000 puntos representan la cantidad de puntos usados para crear la matriz $A \in Mat_{n,2}(\mathbb{R})$ para crear el polinomio encontrado de grado 1, en este caso $m = 1$.

III-B. Ajuste de mínimos cuadrados usando un polinomio de grado m

Consideramos ahora ajustar la función f con un polinomio de grado m mediante mínimos cuadrados, en este caso tomamos $m = 5$ por lo que tenemos un polinomio de la forma

$$P_5(x) = a_0 + \sum_{i=1}^5 a_i x^i$$

En la figura 3 podemos ver el ajuste de un polinomio de grado 5 sobre la función f

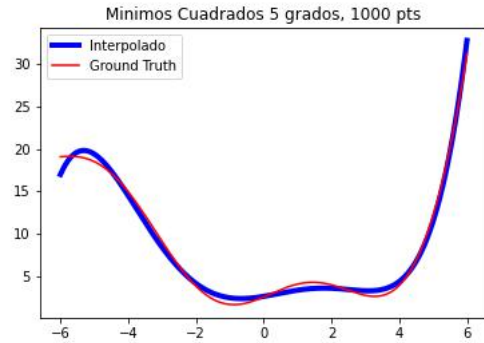


Figura 3. Ajuste con un polinomio de grado 5 con mínimos cuadrados

Podemos ver que el polinomio de grado 5 ajusta mejor a la función f con relación al polinomio de grado 1 descrito anteriormente.

III-C. Ajuste con mínimos cuadrados usando una combinación de funciones g_i

En este caso consideramos tres funciones g_i las cuales son

$$\begin{aligned} g_1 &= \cos(x) \\ g_2 &= \sqrt{7} \exp(x) \\ g_3 &= x^2 \end{aligned}$$

La elección de dichas funciones fue porque cada una pertenece a un tipo de función usado en la función original, en este caso g_1 es una función trigonométrica y relacionada con $\sin(x)$, la segunda función está relacionada con g_2 por un escalar y g_3 es una función polinomial al igual que x . El ajuste por mínimos cuadrados usando esta combinación de funciones lo podemos ver en la figura 4

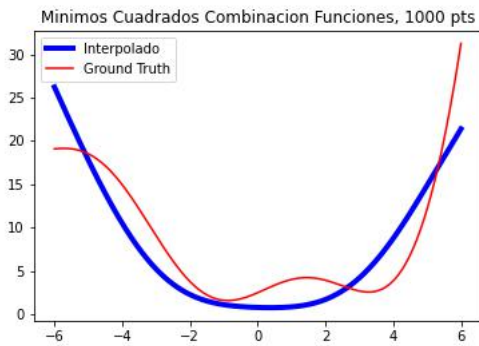


Figura 4. Ajuste por mínimos cuadrados usando la combinación de funciones g_1, g_2 y g_3

III-D. Ajuste por interpolación polinomial

Se probó un ajuste por interpolación polinomial de la función f con un polinomio de grado 10, el grado se obtuvo a partir de resultados experimentales. La gráfica de la interpolación mediante interpolación polinomial se muestra en la Figura 5.

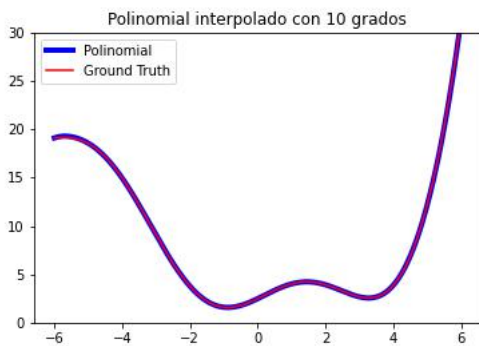


Figura 5. Ajuste con interpolación polinomial con un polinomio de grado 10

Podemos ver que al ajuste de interpolación polinomial logró ajustar muy bien a los puntos en los que fue evaluado para crear \hat{f}

III-E. Ajuste con polinomios de Lagrange

La función f fue interpolada usando polinomios de Lagrange de grado 10.

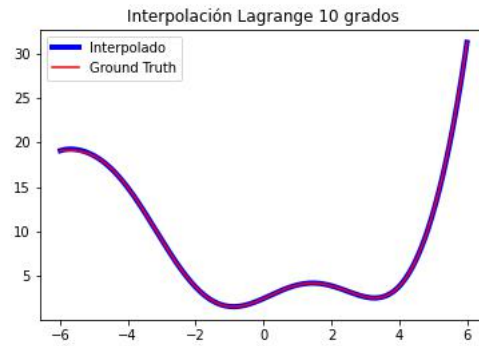


Figura 6. Aproximación de la función f usando polinomios de Lagrange de grado 10

En la figura 6 podemos encontrar la aproximación de la función f mediante polinomios de Lagrange, nuevamente vemos que dicho polinomio ajusta correctamente a la función buscada.

III-F. Ajuste con polinomio de Newton

Finalmente, utilizamos el polinomio de Newton descrito anteriormente para interpolar la función f . En este caso usamos 10 puntos al igual que en los métodos anteriores. La gráfica de la interpolación usando el polinomio de Newton se puede apreciar en la Figura 7

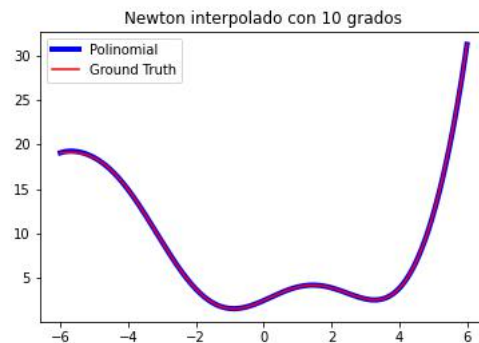


Figura 7. Interpolación usando polinomio de Newton de grado 10

Notemos que al igual que el método de interpolación polinomial e interpolación de Lagrange, el ajuste usando polinomio de Newton es bastante bueno.

III-G. Tabla comparativa de errores de aproximación de los métodos

| Aproximación | $\ f(x) - \hat{f}(x)\ ^2$ |
|-----------------------------------|---------------------------|
| MC:Polinomio de grado 1 | 0.2139 |
| MC:Polinomio de grado 5 | 0.0189 |
| MC:Combinación de funciones | 0.1016 |
| IP:Polinomio de grado 10 | 0.00121441 |
| IP:Polinomio Lagrange de grado 10 | 0.00125227 |
| IP:Polinomio Newton de grado 10 | 0.00121441 |

Cuadro I
TABLA COMPARATIVA DE ERRORES DE APROXIMACIÓN DE FUNCIONES

En la Tabla I podemos observar que los métodos que obtuvieron una mejor aproximación a la función f fueron los métodos de interpolación polinomial, interpolación de Newton e interpolación de Lagrange, de estos la interpolación de Newton y la interpolación polinomial obtuvieron los mejores resultados. Esto respalda los resultados cualitativos mostrados en las figuras 2,3,4,5,6 y 7

IV. CONCLUSIONES

De acuerdo a los resultados obtenidos en el presente trabajo, podemos concluir que el método de interpolación Polinomial y el método de interpolación de Newton dieron los mejores resultados para aproximar la función f . No obstante, el método de newton requiere más operaciones computacionales entre mayor sea el grado del polinomio que se desee ajustar. Por otro lado, vemos que la combinación de funciones propuesta no fue la mejor para aproximar la función f ya que el error de aproximación es mayor al error de los demás métodos, salvo el de ajuste de una recta a los datos en donde claramente se esperaba un mal ajuste porque f no es una recta.

REFERENCIAS

- [1] Richard L Burden, J Douglas Faires, and Annette M Burden. *Numerical analysis*. Cengage learning, 2015.

V. APÉNDICE A: CAPTURAS DE PANTALLA DE LOS MÉTODOS

V-A. Mínimos cuadrados con polinomio de grado 1

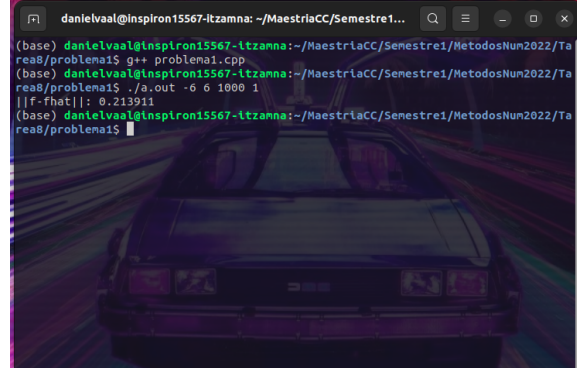


Figura 8. Salida de ejecución del problema 1

V-B. Mínimos Cuadrados con polinomio de grado 5

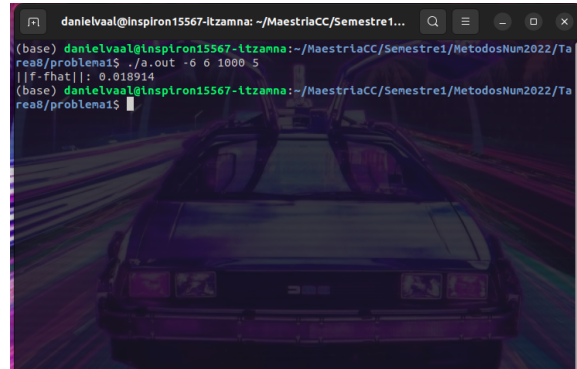


Figura 9. Salida de ejecución del problema 2

V-C. Mínimos Cuadrados con combinación de funcio- nes



Figura 10. Salida de ejecución del problema 3

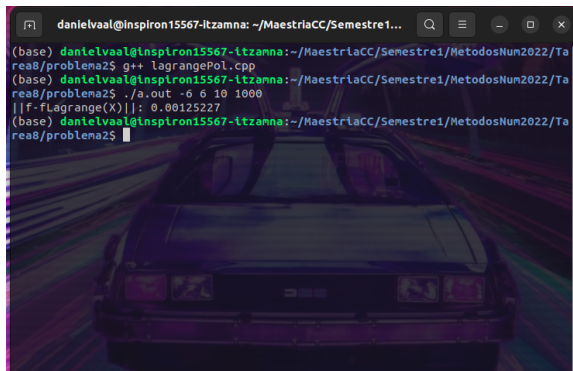
V-D. Interpolación polinomial



```
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1...  
rea8/problema25 g++ degreePol.cpp  
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1/MetodosNun2022/Ta  
rea8/problema25 ./a.out -6 6 10 1000  
11  
||f-Pn(X)||: 0.00121441  
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1/MetodosNun2022/Ta  
rea8/problema25
```

Figura 11. Salida de ejecución del problema 4

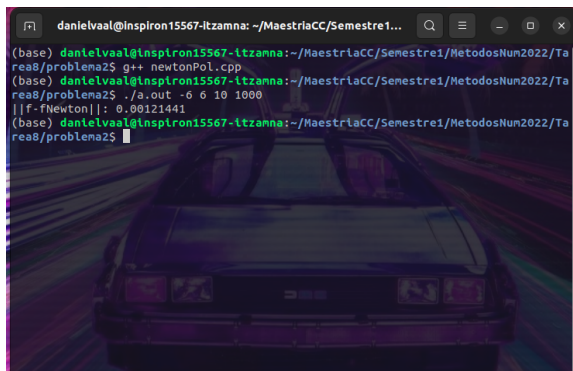
V-E. Interpolación con Polinomio de Lagrange



```
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1...  
rea8/problema25 g++ lagrangePol.cpp  
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1/MetodosNun2022/Ta  
rea8/problema25 ./a.out -6 6 10 1000  
||f-Flagrange(X)||: 0.00125227  
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1/MetodosNun2022/Ta  
rea8/problema25
```

Figura 12. Salida de ejecución del problema 5

V-F. Interpolación con polinomio de Newton



```
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1...  
rea8/problema25 g++ newtonPol.cpp  
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1/MetodosNun2022/Ta  
rea8/problema25 ./a.out -6 6 10 1000  
||f-fNewton||: 0.00121441  
(base) danielvaal@inspiron15567-ltzanna: ~/MaestriaCC/Semestre1/MetodosNun2022/Ta  
rea8/problema25
```

Figura 13. Salida de ejecución del problema 6