Capítulo 4
Interpolación

Este capítulo me inspira a proponer una de las ideas que por al menos el último siglo viene rondando la mente de muchos matemáticos. Cuál es la fórmula de los números primos.

Vamos a considerar la sucesión de los primeros números primos

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, \ldots$$

¿Será posible construir una sucesión que los contenga exactamente para los primeros elementos de la sucesión?

Sea a_n una sucesión que queremos construir de tal forma que cada n para $n = 1, 2, \ldots, k$ genere un número primo de la sucesión anterior, consideremos el primer término

$$a_1 = 2$$

¿Cómo podemos construir el segundo término $a_2 = 3$?

$$\begin{aligned} a_2 &= (n-1)\,x + a_1 \\ 3 &= (2-1)\,x + 2 \\ 3 &= x + 2 \\ 1 &= x \end{aligned}$$

de tal forma que $a_2 = (n-1) + 2$ con ello podemos asegurar los dos primeros términos de la sucesión.

¿Cómo podemos construir el tercer término $a_3 = 5$?

$$\begin{aligned}
a_3 &= (n-2)(n-1)x + a_2 \\
5 &= (3-2)(3-1)x + (3-1) + 2 \\
5 &= (1)(2)x + 4 \\
1 &= 2x \\
\frac{1}{2} &= x
\end{aligned}$$

así el término a_3 se puede obtener así

$$a_3 = \frac{1}{2}(n-2)(n-1) + (n-1) + 2$$

Si continuamos encontrando valores para a_n la expresión crecerá aun más, por lo cual vamos a simplificar algunos términos y obtenemos

$$a_3 = \frac{n^2}{2} - \frac{n}{2} + 2$$

Ahora para $a_4 = 7$

$$\begin{aligned}
a_4 &= (n-3)(n-2)(n-1)x + a_3 \\
a_4 &= (n-3)(n-2)(n-1)x + \frac{n^2}{2} - \frac{n}{2} + 2 \\
7 &= (4-3)(4-2)(4-1)x + \frac{4^2}{2} - \frac{4}{2} + 2 \\
7 &= 6x + 8 \\
-1 &= 6x \\
-\frac{1}{6} &= x
\end{aligned}$$

entonces

$$\begin{aligned}
a_4 &= -\frac{1}{6}(n-3)(n-2)(n-1) + \frac{n^2}{2} - \frac{n}{2} + 2 \\
&= -\frac{n^3}{6} + \frac{3n^2}{2} - \frac{7n}{3} + 3
\end{aligned}$$

Esto lo podemos hacer así indefinidamente (hasta que sea posible calcular el polinomio analíticamente). A continuación se muestra un polinomio de grado doce (12)

$$
\begin{aligned}
a_n = & \frac{1213n^{12}}{479001600} - \frac{17371n^{11}}{79833600} + \frac{362767n^{10}}{43545600} - \frac{38867n^{9}}{207360} \\
& + \frac{5706469n^{8}}{2073600} - \frac{66963781n^{7}}{2419200} + \frac{8481952741n^{6}}{43545600} - \frac{1395663287n^{5}}{1451520} \\
& + \frac{35761373867n^{4}}{10886400} - \frac{1948933907n^{3}}{259200} + \frac{1287969941n^{2}}{118800} \\
& - \frac{241928479n}{27720} + 2914
\end{aligned}
$$

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Unidad-004-Interpolacion-Aproximacion-Polinomial\Programas\Evaluacion-Polinomio-Primos.ipynb*

4.1 Introducción

En este capítulo vamos a **aproximar** una función f cualquiera por un miembro de la familia más conocida y más sencilla de tratar: **los polinomios**.

Recordemos que un **polinomio de grado** n tiene una expresión de la forma siguiente:

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = \sum_{i=0}^{n} a_i x^i$$

con $a_n \neq 0$ para que tenga grado n.

Los valores a_i, $i = 0, 1, \ldots, n$ se denominan **coeficientes** del polinomio P_n.

Cuanto “mejor” sea la función f a **aproximar**, es decir, cuanto más alto sea el valor de k donde $f \in \mathcal{C}^k$, mejor **control** sobre el **error cometido** en la aproximación tendremos.

Para justificar la aproximación de una función f por polinomios veamos el **Teorema de Weierstrass** que dice básicamente que cualquier función continua puede aproximarse por un polinomio con un error tan pequeño como se quiera:

Teorema 1: Teorema de Weierstrass

Sea $f \in \mathcal{C}^0[a,b]$ una función continua en un intervalo $[a,b]$. Entonces dado un valor $\epsilon > 0$, existe un polinomio $P_n(x)$ de un grado n tal que:

$$|f(x) - P_n(x)| < \epsilon,$$

para todo valor $x \in [a,b]$

Ejercicio 1:

Crear un *programa* en **Python** que grafique una función y la traslade un valor ϵ por encima y por debajo. Como ejemplo se puede graficar la función $f(x) = x \cdot \sin\left(\frac{\pi}{2}x\right)$. Luego las funciones $g_1(x) = \frac{1}{2}x^3 - 3x^2 + \frac{7}{2}x + \epsilon$ y $g_2(x) = \frac{1}{2}x^3 - 3x^2 + \frac{7}{2}x - \epsilon$. Determinar para que valores de ϵ la gráfica de la función f se encuentra dentro de las funciones $g_1(x)$ y $g_2(x)$

```python
# Importar las librerias necesarias
import numpy as np
import matplotlib.pyplot as plt

# Definir las funciones a graficar
def f(x):
    pi2 = np.pi / 2
    return x*np.sin(pi2*x)

def g(x,epsilon):
    return (1/2)*x**3-3*x**2+(7/2)*x+epsilon

# Definir el valor de epsilon
epsilon = 4

# Definir el dominio de graficación
x = np.linspace(0, 5, 500)

```

```
# Calcular las imágenes de las funciones
y1 = f(x)
y2 = g(x,epsilon)
y3 = g(x,-epsilon)

# Dibujar las funciones
plt.plot(x,y1)
plt.plot(x,y2)
plt.plot(x,y3)

# Configurar la rejilla
plt.grid(linestyle='--')

# Configurar las etiquetas del gráfico
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.title('Gráfico de las funciones $f(x)$, $g(x)+\\epsilon$ y $g(x)-\\epsilon$')
plt.legend(['$f(x)=x\\cdot\\sin\\left(\\frac{\\pi}{2}x\\right)$', '$g(x)=\\frac{1}{2}x^{3}-3x^{2}+\\frac{7}{2}x+\\epsilon$', '$g(x)=\\frac{1}{2}x^{3}-3x^{2}+\\frac{7}{2}x-\\epsilon$'])

# Mostrar el gráfico
plt.show()
```

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Unidad-004-Interpolacion-Aproximacion-Polinomial\Programas\Teorema-Weierstrass.ipynb*

Observación

La aproximación anterior es uniforme en el sentido de que el “error” ϵ no depende del valor $x \in [a, b]$

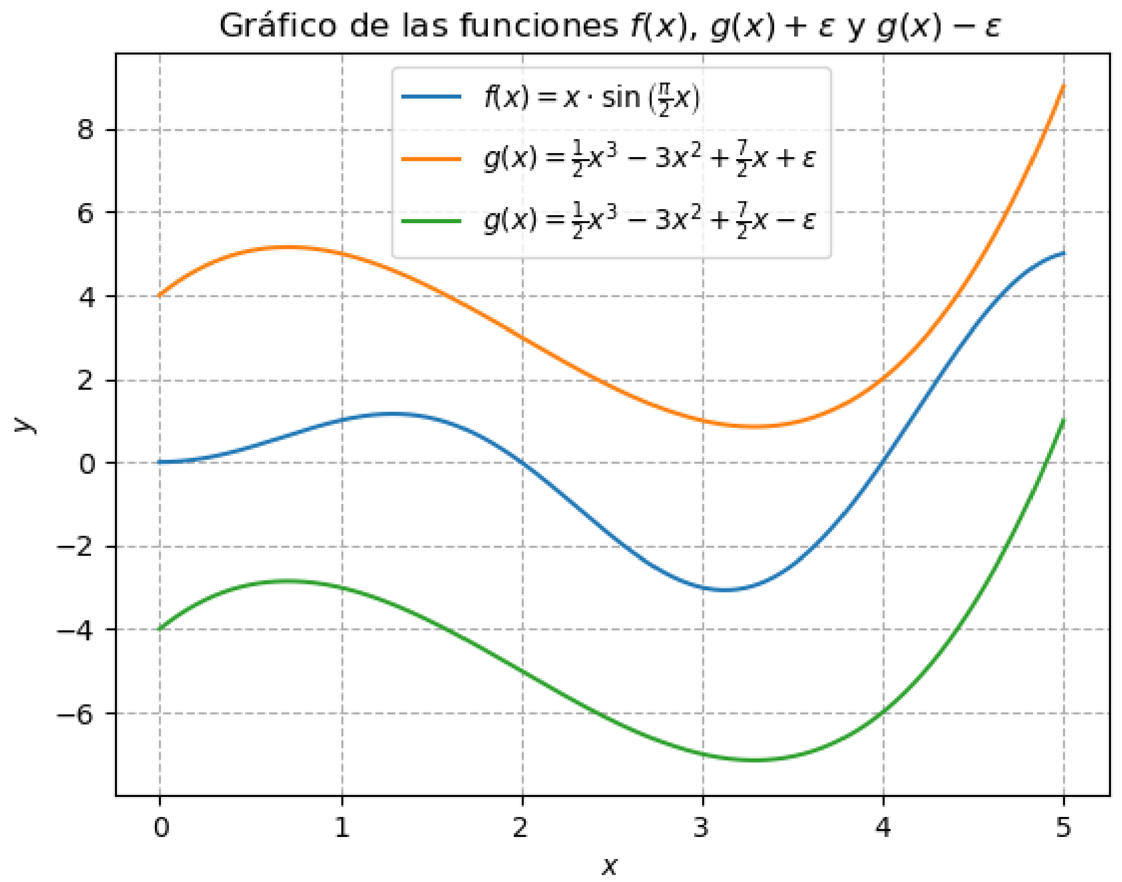

Figura 4.1: Gráfica de la función $f(x)=\frac{1}{2}x^3-3x^2+\frac{7}{2}x-\epsilon$ con un valor $\epsilon=\pm 4$

El problema que intentamos resolver es el siguiente:

Problema

Dados n valores $(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)$, hallar el **polinomio** P_n de grado mínimo tal que $P_n(x_i)=y_i$, $i=0,1,2,\ldots n$.

Es decir, dados $n+1$ puntos en el plano, hallar un polinomio de grado mínimo, tal que

$$P_n(x_i)=y_i,\ \ i=0,1,2,\ldots n$$

Observación

- Si los puntos son parte de la gráfica de una función f, entonces $y_i=f(x_i)$ y las condiciones que debe verificar el polinomio $P_n(x)$ son $P_n(x_i)=f(x_i)$, con $i=0,1,2,\ldots n$
- Tenemos en total $n+1$ **condiciones**, por tanto, el número de incógnitas debe ser $n+1$. Pensemos que un polinomio de grado n tiene en total $n+1$ coeficientes.

Sea pues $P_n(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$ el polinomio que queremos hallar.

Las condiciones $P_n(x_i) = y_i$ serían las siguientes en función de los coeficientes a_i, con $i = 0, 1, 2, \ldots, n$:

$$a_0 + a_1x_i + a_2x_i^2 + a_3x_i^3 + \cdots + a_nx_i^n = y_i, \text{ con } i = 0, 1, 2, \ldots, n$$

Los coeficientes a_i deben verificar el siguientes sistema de ecuaciones lineal:

$$A = \begin{cases} a_0 + a_1x_0 + \cdots + a_nx_0^n & = y_0 \\ a_0 + a_1x_1 + \cdots + a_nx_1^n & = y_1 \\ & \vdots \\ a_0 + a_1x_n + \cdots + a_nx_n^n & = y_n \end{cases}$$

Vemos que el **sistema lineal** anterior tiene solución única.

El **determinante** del sistema es el siguiente

$$\det(A) = \begin{vmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & \cdots & x_n^n \end{vmatrix}$$

El determinante anterior se llama **Determinante de Vandermonde** y su valor es:

$$\prod_{0 \leq i \leq j \leq n} (x_i - x_j)$$

Por tanto, si $x_i \neq x_j$ para $i \neq j$, el **determinante del sistema** no será cero y tendremos **solución única** para nuestro problema, dado que todo sistema que tiene determinante definido tiene única solución.

Tenemos la siguiente matriz de orden 3×3

$$A = \begin{pmatrix} x^0 & x^1 & x^2 \\ y^0 & y^1 & y^2 \\ z^0 & z^1 & z^2 \end{pmatrix}$$

al calcular el determinante de la matriz por cofactores tenemos

$$
\begin{aligned}
\det A &= \left(x^0\right)\left(y^1\cdot z^2 - y^2\cdot z^1\right) - \left(x^1\right)\left(y^0\cdot z^2 - y^2\cdot z^0\right) + \left(x^2\right)\left(y^0\cdot z^1 - y^1\cdot z^0\right)\\
&= x^0\cdot y^1\cdot z^2 - x^0\cdot y^2\cdot z^1 - x^1\cdot y^0\cdot z^2 + x^1\cdot y^2\cdot z^0 + x^2\cdot y^0\cdot z^1 - x^2\cdot y^1\cdot z^0\\
&= x^0\cdot y^1\cdot z^2 + x^1\cdot y^2\cdot z^0 + x^2\cdot y^0\cdot z^1 - x^0\cdot y^2\cdot z^1 - x^1\cdot y^0\cdot z^2 - x^2\cdot y^1\cdot z^0\\
&= y^1\cdot z^2 + x^1\cdot y^2 + x^2\cdot z^1 - y^2\cdot z^1 - x^1\cdot z^2 - x^2\cdot y^1\\
&= \left(x^2\cdot z^1 - x^2\cdot y^1\right) + \left(x^1\cdot y^2 - y^2\cdot z^1\right) + \left(y^1\cdot z^2 - x^1\cdot z^2\right)\\
&= x^2\left(z^1 - y^1\right) + y^2\left(x^1 - z^1\right) + z^2\left(y^1 - x^1\right)\\
&= x^2\left(z-y\right) + y^2\left(x-z\right) + z^2\left(y-x\right)
\end{aligned}
$$

Tenemos la siguiente productoria

$$
\begin{aligned}
\prod_{0\le i\le j\le 2}\left(x_i - x_j\right) &= \prod_{i=0}^{2}\left(\prod_{j=i}^{2}\left(x_i - x_j\right)\right)\\
&= \prod_{j=0}^{2}\left(x_0 - x_j\right)\cdot\prod_{j=1}^{2}\left(x_1 - x_j\right)\cdot\prod_{j=2}^{2}\left(x_2 - x_j\right)\\
&= \left[\left(x_0 - x_0\right)\left(x_0 - x_1\right)\left(x_0 - x_2\right)\right]\left[\left(x_1 - x_1\right)\left(x_1 - x_2\right)\right]\left[\left(x_2 - x_2\right)\right]\\
&= \left[\left(x_0 - x_1\right)\left(x_0 - x_2\right)\right]\left[\left(x_1 - x_2\right)\right]
\end{aligned}
$$

cambiaremos $x_0 = x$, $x_1 = y$ y $x_2 = z$ entonces

$$
\begin{aligned}
&\left(x-y\right)\left(x-z\right)\left(y-z\right)\\
&= \left(x^2 - x\cdot z - y\cdot x + y\cdot z\right)\left(y-z\right)\\
&= x^2\cdot y - x^2\cdot z - x\cdot y\cdot z + x\cdot z^2 - y^2\cdot x + x\cdot y\cdot z + y^2\cdot z - y\cdot z^2\\
&= x^2\cdot y - x^2\cdot z + z^2\cdot x - y^2\cdot x + y^2\cdot z - z^2\cdot y\\
&= \left(x^2\cdot y - x^2\cdot z\right) + \left(y^2\cdot z - y^2\cdot x\right) + \left(z^2\cdot x - z^2\cdot y\right)\\
&= x^2\left(y-z\right) + y^2\left(z-x\right) + z^2\left(x-y\right)
\end{aligned}
$$

Así hemos demostrado que la fórmula de la productoria es verdadera para matrices de dimensión 3×3

Ejercicio 2:

Construir un programa en Python que verifique que el siguiente determinante se puede calcular mediante la definición del **Determinante de Vandermonde**, comparando el cálculo a través de la formula y mediante el módulo **np.linalg.det**

$$\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 \\ 1 & 3 & 9 & 27 & 81 \\ 1 & 4 & 16 & 64 & 256 \\ 1 & 5 & 25 & 125 & 625 \end{vmatrix}$$

```python
# Importar las librerias necesarias
import numpy as np

# Definir la matriz
A = np.array([
    [1, 1, 1, 1, 1],
    [1, 2, 4, 8, 16],
    [1, 3, 9, 27, 81],
    [1, 4, 16, 64, 256],
    [1, 5, 25, 125, 625]
])

# Imprimir la matriz
print("Matriz a Cacular el Determinante: \n", A)

# Mostrar la forma de la matriz
print("Tamaño de la Matriz: ", np.shape(A))

# Calcular y Mostrar el Determinante con Numpy
detA = np.linalg.det(A)
print("Determinante por Cofactores: ", detA)

# Calcular el Determinante Mediante la Productoria
x = [1, 2, 3, 4, 5]
detA_ = 1              # Se inicia el valor del determinante en 1
n = np.shape(A)[0]     # Se obtiene la dimensión de la matriz
```

```python

# Se crea el ciclo para las productorias anidadas
for i in range(0, n):
    for j in range(i, n):

        # Calcular la productoria, siempre que i sea diferente de j
        if i!=j:
            detA_ = detA_*(x[i]-x[j])

# Mostrar el determinante mediante la productoria
print("Determinante por Productoria: ", detA_)
```

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Unidad-004-Interpolacion-Aproximacion-Polinomial\Programas\Determinante-Vandermonde.ipynb*

4.2 Teorema del Polinomio Interpolador

En resumen, tenemos el teorema siguiente:

Teorema 2:

Sean $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, n valores con $x_i \neq x_j$, para $i \neq j$, (es decir, las abscisas son todas diferentes). Entonces existe un único polinomio $P_n(x)$ de grado n tal que $P_n(x_i) = y_i$, para $i = 0, 1, 2, \ldots, n$.

Ejemplo 1:

Calcular un polinomio $P_5(x)$ que interpola a los puntos siguientes: $(1, 2)$, $(2, 3)$,$(3, 5)$, $(4, 7)$, $(5, 11)$ y $(6, 13)$

Al crear el sistema de coeficientes tenemos

$$P_5(x) = \begin{cases} a_0 + a_1(1) + a_2\left(1^2\right) + \cdots + a_5\left(1^5\right) & = 2 \\ a_0 + a_1(2) + a_2\left(2^2\right) + \cdots + a_5\left(2^5\right) & = 3 \\ a_0 + a_1(3) + a_2(3) + \cdots + a_5\left(3^5\right) & = 5 \\ a_0 + a_1(4) + a_2\left(4^2\right) + \cdots + a_5\left(4^5\right) & = 7 \\ a_0 + a_1(5) + a_2\left(5^2\right) + \cdots + a_5\left(5^5\right) & = 11 \\ a_0 + a_1(6) + a_2(6) + \cdots + a_5\left(6^5\right) & = 13 \end{cases}$$

Esto se puede expresar matricialmente como

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 \\ 1 & 3 & 9 & 27 & 81 & 243 \\ 1 & 4 & 16 & 64 & 256 & 1024 \\ 1 & 5 & 25 & 125 & 625 & 3125 \\ 1 & 6 & 36 & 216 & 1296 & 7776 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 5 \\ 7 \\ 11 \\ 13 \end{pmatrix}$$

Al resolver este sistema encontraremos los valores de los coeficientes $a_0, \ldots, a_n$ del **Polinomio Interpolador.**

Para resolver este sistema vamos a construir un *programa* en **Python**

```python
# Importar Numpy
import numpy as np

# Crear la matriz de coeficientes
a = np.array([
    [1,1,1,1,1,1],
    [1,2,4,8,16,32],
    [1,3,9,27,81,243],
    [1,4,16,64,256,1024],
    [1,5,25,125,625,3125],
    [1,6,36,216,1296,7776]
])

# Crear el vector de términos indepdientes
b = np.array([2,3,5,7,11,13])

# Resolver el sistema
```

```
x = x = np.linalg.solve(a,b.T)

# Mostrar el resultado
print(x)
```

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Unidad-004-Interpolacion-Aproximacion-Polinomial\Programas\Resolver-Sistema-Ecuaciones-PnX.ipynb*

La solución al sistema es el siguiente:

$$a_i = \begin{bmatrix} 15 & -29.13 & 22.75 & -7.79 & 1.25 & -0.075 \end{bmatrix}$$

esto quiere decir que el polinomio interpolador esta dado por

$$P_n\left(x\right) = -0.075x^5 + 1.25x^4 - 7.79x^3 + 22.75x^2 - 29.13x + 15$$

Ejercicio 3:

Crear un *programa* en **Python** que grafique el Polinomio Interpolador $P_n\left(x\right) = -0.075x^5 + 1.25x^4 - 7.79x^3 + 22.75x^2 - 29.13x + 15$. Evalué el polinomio en los puntos $x \in [1, 7]$, con x entero, y grafiquelos juntamente con el polinomio.

```
# Importar MatPlotLib
import matplotlib.pyplot as plt

def pn(x):
    return -0.075*x**5+1.25*x**4-7.79*x**3+22.75*x**2-29.13*x+15

# Crear un vector de términos de x
x = np.linspace(0, 7, 500)

# Evaluar la función en los valores de x
y = pn(x)

# Graficar la función
plt.plot(x,y)
plt.xlabel('$n$')
```

```
plt.ylabel('$P_n$')
plt.title('Gráfica del Polinomio Interpolador')
plt.grid(linestyle='--')

# Graficar puntos
x1 = np.array([1,2,3,4,5,6,7])
y1 = pn(x1)
plt.scatter(x1,y1, c='red')

# Mostrar la leyenda y la gráfica
plt.legend(['$P_n(x)$', 'Puntos'])
plt.show()

# Mostrar la aproximación de los puntos
print(y1)
```

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Unidad-004-Interpolacion-Aproximacion-Polinomial\Programas\Polinomio_Interpolador.ipynb*

A continuación se presenta una gráfica del polinomio interpolador encontrado.

Observación

En este ejercicio no conocemos a $f(x)$, pero suponemos que es una función que genero los valores $2, 3, 5, 7, 11$ y 13 para los nodos $x \in \{1, 2, 3, 4, 5, 6\}$

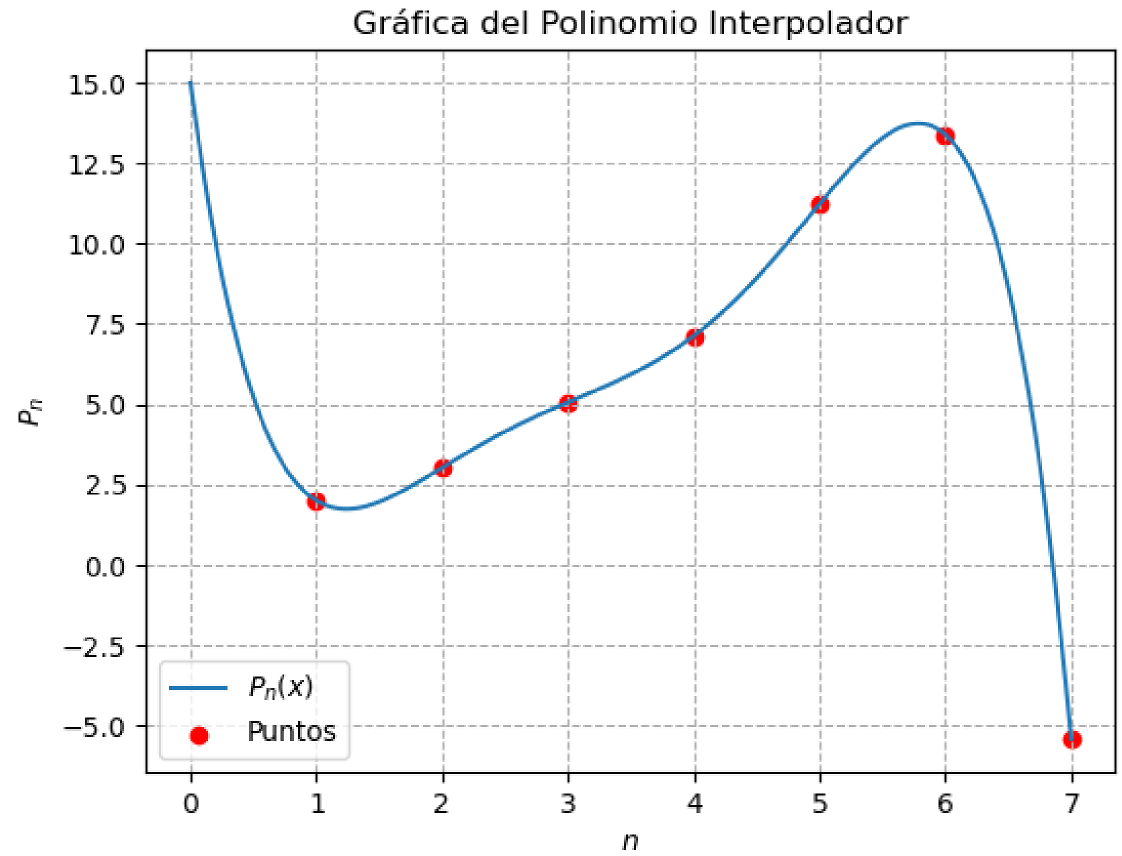

Figura 4.2: Gráfica del Polinomio Interpolador

Los puntos evaluados en el polinomio son:

$$P_n\left([1,7]\right) = \begin{bmatrix}2.005 & 3.02 & 5.055 & 7.12 & 11.225 & 13.38 & -5.405\end{bmatrix}$$

4.3 Polinomios de Lagrange

Para hallar el polinomio $P_n\left(x\right)$ deberíamos resolver el **sistema anterior** para calcular los coeficientes del polinomio a_i, con $i = 0, 1, 2, \ldots, n$.

Resolver el sistema lineal anterior es computacionalmente costoso, sobre todo si el número de nodos $n+1$ es grande. Por dicho motivo, vamos a estudiar técnicas para obtener el **polinomio interpolador** sin tener que resolver dicho sistema.

Una de dichas técnicas consiste en escribir el **Polinomio Interpolador** de forma explícita en función de unos polinomios especiales denominados **Polinomios Interpoladores de Lagrange:**

Definición 1: Polinomios de Lagrange

Sean $x_0, x_1, \ldots, x_n$, $n+1$ nodos donde suponemos que $x_i \neq x_j$, para $i \neq j$. Se define el polinomio de Lagrange $L_{n,k}(x)$ de grado n asociado al nodo x_k de la forma siguiente:

$$\begin{aligned} L_{n,k}(x) &= \frac{(x-x_0)\cdots(x-x_{k-1})\cdot(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})\cdot(x_k-x_{k-1})\cdots(x_k-x_n)} \\ &= \frac{\prod_{i\neq k}(x-x_i)}{\prod_{i\neq k}(x_k-x_i)} \end{aligned}$$

Es decir de todos los productos posibles, (no se tiene en cuenta) el producto donde aparece el valor x_k.

Los polinomios de Lagrange verifican la proposición siguiente:

Proposición

Sean $x_0, x_1, x_2, \ldots x_n$, $n+1$ nodos donde $x_i \neq x_j$, para $i \neq j$. Sea $L_{n,k}(x)$ el **Polinomio de Lagrange** de grado n asociado al nodo x_k. Entonces dicho polinomio verifique que:

$$L_{n,k}(x_i) = 0, \text{ si } i \neq k, \; L_{n,k}(x_k) = 1$$

El polinomio de Lagrange es cero si es evaluado en cualquier nodo excepto en el nodo x_k y es 1 si es evaluado en el nodo x_k

Demostración

Si $i \neq k$,

$$L_{n,k}(x_i) = \frac{(x_i-x_0)\cdots(x_i-x_i)\cdots(x_i-x_{k-1})\cdot(x_i-x_{k+1})\cdots(x_i-x_n)}{(x_k-x_0)\cdots(x_k-x_i)\cdots(x_k-xk_{-1})\cdot(x_k-x_{k+1})\cdots(x_k-x_n)}$$

ya que tenemos un factor $x_i - x_i = 0$ el numerador se anula.

Por otra parte si $i = k$

$$L_{n,k}(x) = \frac{(x_k-x_0)\cdots(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdots(x_k-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})\cdot(x_k-x_{k+1})\cdots(x_k-x_n)} = 1$$

ya que el numerador y denominador son iguales.

Ejemplo 2:

Consideremos los nodos $x_0 = 0$, $x_1 = 1$, $x_2 = 3$ y $x_3 = 5$. Los nodos de Lagrange asociados a los nodos anteriores son los siguientes.

- $L_{3,0}(x)$

$$
\begin{aligned}
&= \frac{(x-1)(x-3)(x-5)}{(0-1)(0-3)(0-5)} \\
&= -\frac{1}{15}(x-1)(x-3)(x-5) \\
&= -\frac{1}{5}\left(x^3 - 9x^2 + 23x - 15\right)
\end{aligned}
$$

- $L_{3,1}(x)$

$$
\begin{aligned}
&= \frac{(x-0)(x-3)(x-5)}{(1-0)(1-3)(1-5)} \\
&= \frac{1}{8}x(x-3)(x-5) \\
&= \frac{1}{8}\left(x^3 - 8x^2 + 15x\right)
\end{aligned}
$$

- $L_{3,2}(x)$

$$
\begin{aligned}
&= \frac{(x-0)(x-1)(x-5)}{(3-0)(3-1)(3-5)} \\
&= -\frac{1}{12}x(x-1)(x-5) \\
&= -\frac{1}{12}\left(x^3 - 6x^2 + 5x\right)
\end{aligned}
$$

- $L_{3,3}(x)$

$$
\begin{aligned}
&= \frac{(x-0)(x-1)(x-3)}{(5-0)(5-1)(5-3)} \\
&= \frac{1}{40}x(x-1)(x-3) \\
&= \frac{1}{40}\left(x^3 - 4x^2 + 3x\right)
\end{aligned}
$$

Ejercicio 4:

1. Construir un *programa* en **Python** que calcule el los **Polinomios de Lagrange** de una lista de nodos.
2. Crear un *programa* en **Python** que grafique los **Polinomios de Lagrange** calculados en el ejercicio anterior. Comprobar la propiedad
$$L_{n,k}(x_i) = 0, \text{ si } i \neq k, \; L_{n,k}(x_k) = 1$$

```python
# Importar las librerias necesarias
import numpy as np
import Multiplicacion_Sintetica as ms
import Evaluar_Polinomio as ep
import matplotlib.pyplot as plt

# Definir la función filterById
def filterById(lista, id):

    """
    ## ***Función:*** filterById
    - **Descripción:** Dada una lista de valores, devuelve una sublista **excluyendo el índice** enviado en el parámetro
    - **Parámetros:**
        - *lista:* Lista de elementos a filtrar
        - *id:* Índice a excluir de la lista
    - **Valor de Retorno:** Lista filtrada
    """

    # Filtrar el ID de la lista
    p1 = lista[:id]
    p2 = lista[id+1:]
    result = np.concatenate((p1,p2))
    return result

# Probar la función filterByID
lista = [1,2,3,4]
nLista = filterById(lista, 2)
print(nLista)
```

```

# Definir la función PolyLagrange
def PolyLagrange(nodos, console=False):

    """
    ## ***Función:*** PolyLagrange
    - **Descripción:** Dada una lista de nodos (raíces) calcular los polinomios de Lagrange.
    - **Parámetros:**
        - *nodos:* Lista de nodos o raices
        - *console:* Valor booleano. Permite mostrar mensajes de procesos del método si el parámetro esta en True.
    - **Valor de Retorno:** Devuelve una matriz por filas. Cada fila corresponde a los polinomios de Lagrange

    """

    # Definir la matriz de salida
    n = np.shape(nodos)[0]
    mPoly = np.empty((0, n))

    # Crear un ciclo para crear cada polinomio
    for i, nodo in enumerate(nodos):

        # Definir la lista de raices
        raices = filterById(nodos, i)

        # Calcular el denominador de Lagrange
        denominador = 1
        for r in raices:
            denominador = denominador * (nodo - r)

        # Calcular el polinomio
        poly = (1 / denominador) * ms.MultiplicacionSintetica(raices)
```

```
        if console:
            print(poly)

        # Agregar el polinomio a la matriz
        mPoly = np.vstack([mPoly, poly])

    return mPoly

# Probar la función
nodos = [0, 1, 3, 5]
pl = PolyLagrange(nodos)
print(pl)

# ### Graficar los Polinomios de Lagrange
# Usar la función *evalPoly* para graficar los **
    Polinomios de Lagrange** Generados

# Generar un dominio de graficación
dominio = np.linspace(-0.5, 5.5, 500)

# Evaluar los polinomios y graficarlos
for poly in pl:

    # Evaluar el polinomio
    imagen = ep.evalPoly(poly, dominio)

    # Crear la gráfica
    plt.plot(dominio, imagen)

# Configurar y Mostrar la gráfica
plt.grid(linestyle='--')
plt.show()
```

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Unidad-004-Interpolacion-Aproximacion-Polinomial\Programas\Polinomios-Lagrange.ipynb*

Ejercicio 5:

4.3.1 Polinomio Interpolador

El Teorema siguiente nos dice cómo calcular el **Polinomio Interpolador** a partir de los **Polinomios de Lagrange:**

Teorema 3:

Sean $n+1$ puntos $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ con $x_i \neq x_j$, con $i \neq j$. Entonces el **Polinomio Interpolador** $P_n(x)$ se puede expresar de la forma siguiente:

$$\begin{aligned} P_n(x) &= y_0 \cdot L_{n,0}(x) + y_1 \cdot L_{n,1}(x) + \cdots + y_n \cdot L_{n,n}(x) \\ &= \sum_{k=0}^{n} y_k \cdot L_{n,k}(x) \end{aligned}$$

donde $L_{n,k}(x)$ es el **Polinomio de Lagrange** correspondiente al nodo x_k, con $k = 0, 1, 2, \ldots, n$.

Observación

Si los $n+1$ puntos forman parte de la **gráfica** de una función f, es decir $y_i = f(x_i)$, para $i = 0, 1, 2, \ldots, n$, entonces el polinomio interpolador se escribirá de la forma siguiente:

$$\begin{aligned} P_n(x) &= f(x_0) \cdot L_{n,0}(x) + f(x_1) \cdot L_{n,1}(x) + \cdots + f(x_n) \cdot L_{n,n}(x) \\ &= \sum_{k=0}^{n} f(x_k) \cdot L_{n,k}(x) \end{aligned}$$

Demostración

Usando que $L_{n,k}(x_i) = 0$, si $i \neq k$ y $L_{n,k}(x_k) = 1$, entonces:

$$\begin{aligned} P_n(x_i) &= \sum_{k=0}^{n} y_k \cdot L_{n,k}(x_i) \\ &= y_i \cdot L_{n,i}(x_i) = y_i \end{aligned}$$

para $i = 0, 1, 2, \ldots, n$. Dicho e otras palabras, $P_n(x)$ interpola a los puntos $(x_0, y_0), \ldots, (x_n, y_n)$, tal como queríamos demostrar.

Ejemplo 3:

Calculemos el polinomio interpolador en los nodos anteriores $x_0 = 0, x_1 = 1, x_2 = 3$ y $x_3 = 5$ para la función $f(x) = x \cdot \sin\left(\frac{\pi}{2}x\right)$

Los puntos a interpolar serían, por tanto:

$$(0,0), (1,1), (3,-3), (5,5)$$

El polinomio interpolador será:

$$\begin{aligned} P_3(x) = 0 \cdot & \left(-\frac{1}{5}\left(x^3 - 9x^2 + 23x - 15\right)\right) \\ & + 1 \cdot \left(\frac{1}{8}\left(x^3 - 8x^2 + 15x\right)\right) \\ & - 3\left(-\frac{1}{12}\left(x^3 - 6x^2 + 5x\right)\right) \\ & + 5\left(\frac{1}{40}\left(x^3 - 4x^2 + 3x\right)\right) \end{aligned}$$

$$= 0.5x^3 - 3x^2 + 3.5x$$

Ejercicio 6:

Crear un *programa* en **Python** que calcule el **Polinomio Interpolador de Lagrange** dados una lista de nodos y una función a aproximar. Particularmente defina la función

$$f(x) = x \cdot \sin\left(\frac{\pi}{2}x\right)$$

en los nodos $[0, 1, 3, 5]$. Grafique la función y su aproximación.

```
# Importar las librerias necesarias
import numpy as np
import Polinomios_Lagrange as pl
import matplotlib.pyplot as plt
import Evaluar_Polinomio as ep

# Define la lista de nodos
```

```python
nodosX = np.array([0, 1, 3, 5])

# Define la función que queremos aproximar
def f(x):
    pi_2 = np.pi / 2
    return x*np.sin(pi_2 * x)

# Calculamos los Polinomios de Lagrange
polyLagrange = pl.PolyLagrange(nodosX)
print("Polinomios de Lagrange: \n", polyLagrange)

# Evaluamos la función a aproximar en los nodos dados
nodosY = f(nodosX)
print("Valores de Y: ", nodosY)

# Calcular el polinomio interpolador
polInt = np.zeros(np.shape(polyLagrange)[0])

for k, poly in enumerate(polyLagrange):
    poly_ = nodosY[k] * poly
    polInt = polInt + poly_

print("Polinomio Interpolador de Lagrange: ", polInt)

# ## Gráfica de la Función y su Aproximación

# Generar un dominio para graficar la función y el polinomio interpolador encontrado
dominio = np.linspace(-0.5, 5.5, 500)
imagenF = f(dominio)
plt.plot(dominio, imagenF)

imagenP = ep.evalPoly(polInt, dominio)
plt.plot(dominio, imagenP)

# Configurar y mostrar imagen
plt.grid(linestyle='--')
```

```
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.title('Gráfica de la función y el polinomio
    interpolador')
plt.legend(['$f(x)$', '$P_n(x)$'])
plt.show()
```

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Unidad-004-Interpolacion-Aproximacion-Polinomial\Programas\Polinomio-Interpolador-Lagrange.ipynb*

A continuación se presenta una gráfica de la función $f(x) = x \cdot \sin\left(\frac{\pi}{2}x\right)$ y el polinomio interpolador aproximado $P_3(x) = 0.5x^3 - 3x^2 + 3.5x$

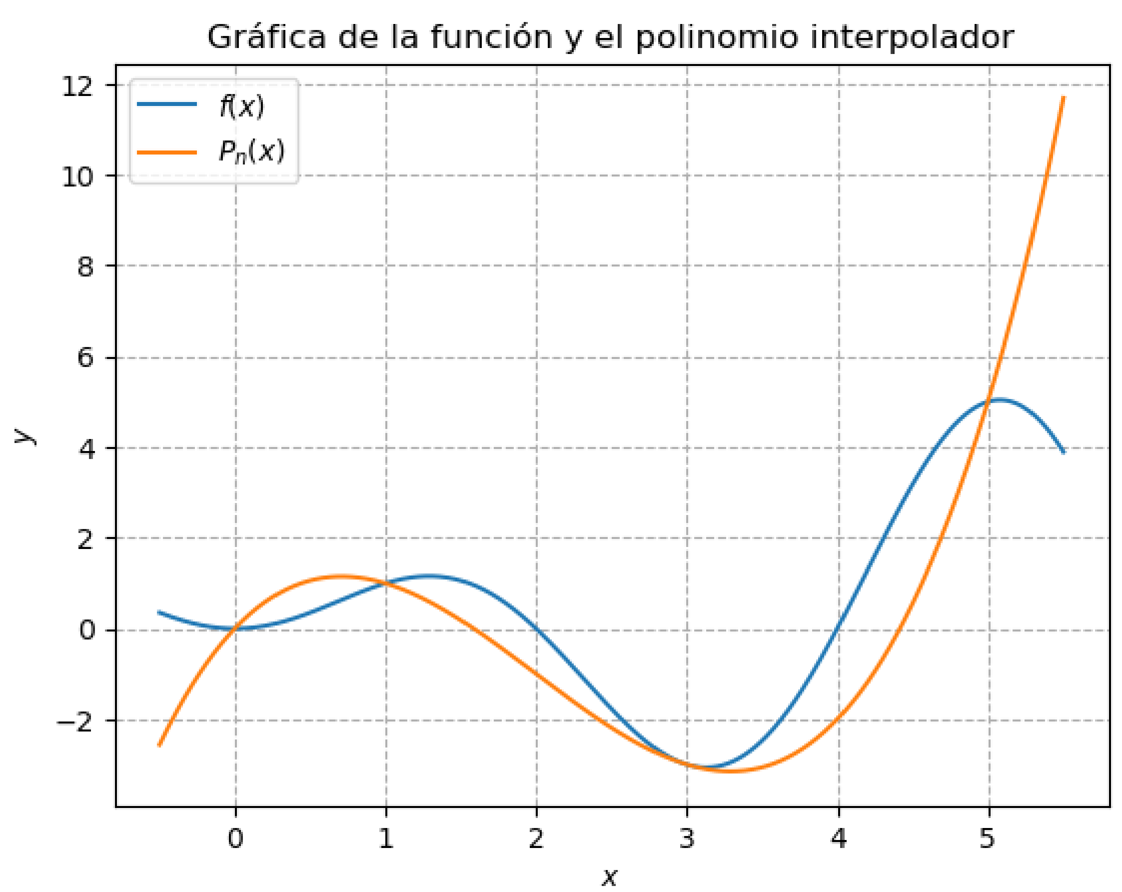

Figura 4.3: Gráfica de la Función y el Polinomio Interpolador

4.3.2 Error de Interpolación

Interpolar una función f en unos nodos determinados puede interpretarse como una manera de aproximar la función f en el entorno de los nodos, es decir, en un dominio que esté relativamente cerca de dichos nodos.

Es fundamental estimar de alguna manera el **error** cometido en un valor cualquiera cuando se intenta aproximar una función. El Teorema siguiente nos da una expresión del **error** cometido cuando **aproximamos** una función f por un **polinomio interpolador.**

Teorema 4: Error de Interpolación

Sea $f \in \mathcal{C}^{n+1}[a,b]$ una función de clase $n+1$ en un intervalo $[a,b]$. Consideramos $n+1$ nodos $x_0,\ldots,x_n$ en dicho intervalo $[a,b]$ con $x_i \neq x_j$, con $i \neq j$. Sea $P_n(x)$ el **Polinomio Interpolador** de una función f en los puntos

$$(x_0, f(x_0)), \ldots, (x_n, f(x_n))$$

Sea x un valor cualquiera dentro del intervalo $[a,b]$. Entonces existe un valor

$$\xi(x) \in \langle x_0,\ldots,x_n,x\rangle = (\min\{x_0,\ldots,x_n,x\}, \max\{x_0,\ldots x_n,x\})$$

(Envoltura Convexa) tal que:

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n)$$

Observación

El error cometido $f(x) - P_n(x)$ cuando interpolamos una función f tiene tres partes bien diferenciables:

1. El producto $(x-x_0)(x-x_1)\cdots(x-x_n)$ nos dice que, cuándo más cerca esté el valor de x de los nodos, mejor será la aproximación.

2. El denominador $(n+1)!$ nos dice que cuantos más nodos tengamos en cuenta, mejor será la aproximación y menor el error cometido.

3. La "caja negra" $f^{(n+1)}(\xi(x))$ que depende de la función f nos dice que tenemos que tener "controladas" las derivadas de la función f ya que si éstas van aumentando sin control, no tendrá ningún sentido interpolar ya que el error cometido puede aumentar indefinidamente.

Demostración

En primer lugar, recordemos que como $P_n(x)$ interpola $f(x)$ tenemos que $P_n(x_i) = f(x_i)$, con $i = 0, \ldots, n$.

Sea x un valor cualquiera dentro del intervalo $[a, b]$ distinto de los nodos x_i. Para este valor, definimos la función siguiente dependiendo de la variable t que tiene el valor x anterior como parámetro:

$$\begin{aligned} G(t) &= f(t) - P_n(t) - (f(x) - P_n(x)) \cdot \frac{(t - x_0)(t - x_1)\cdots(t - x_n)}{(x - x_0)(x - x_1)\cdots(x - x_n)} \\ &= f(t) - P_n(t) - (f(x) - P_n(x)) \prod_{k=0}^{n} \frac{(t - x_k)}{(x - x_k)} \end{aligned}$$

La función anterior también será de clase $n + 1$ en el intervalo $[a, b]$ al serlo f.

Veamos cuánto vale $G(t)$ para $t = x_0, \ldots x_n, x$:

$$G(x_i) = f(x_i) - P_n(x_i) - (f(x) - P_n(x)) \prod_{k=0}^{n} \frac{(x_i - x_k)}{(x - x_k)}$$

para $i = 0, \ldots, n$ ya que $f(x_i) = P_n(x_i)$ y en la productoria uno de los factores será $x_i - x_i = 0$

El valor de

$$\begin{aligned} G(x) &= f(x) - P_n(x) - (f(x) - P_n(x)) \cdot \prod_{k=0}^{n} \frac{(x - x_k)}{(x - x_k)} \\ &= f(x) - P_n(x) - (f(x) - P_n(x)) = 0 \end{aligned}$$

En resumen, la función $G(t)$ tiene $n + 2$ ceros, $x_0, \ldots, x_n, x$. Usando el Teorema del Valor Medio Generalizado, tenemos que existe un valor $\xi(x)$ tal que $G^{(n+1)}(\xi(x)) = 0$.

Calculemos la derivada $n + 1$-ésima de $G(t)$:

$$G^{(n+1)(t)} = f^{(n+1)}(t) - P_n^{(n+1)}(t) - (f(x) - P_n(x)) \left(\prod_{k=0}^{n} \frac{(x - x_k)}{(x - x_k)} \right)^{(n+1)}$$

El valor $P_n^{(n+1)}(t) = 0$ vale 0 ya que en general la derivada $n+1$-ésima de un polinomio de grado n es nula.

El valor de $\left(\prod_{k=0}^{n} \frac{(t-x_k)}{(x-x_k)}\right)^{(n+1)}$ consiste en hallar la derivada $n+1$-ésima de un polinomio en t de grado $n+1$ que valdrá el coeficiente del monomio de grado superior, es decir, el coeficiente de t^{n+1} multiplicado por $(n+1)!$

El coeficiente de t^{n+1} del polinomio de grado $n+1$, $\prod_{k=0}^{n} \frac{(x-x_k)}{(x-x_k)}$ valdrá

$$\prod_{k=0}^{n} \frac{1}{(x-x_k)} = \frac{1}{\prod_{k=0}^{n}(x-x_k)}$$

Entonces:

$$G^{(n+1)}(t) = f^{(n+1)}(t) - \frac{(n+1)!\,(f(x) - P_n(x))}{\prod_{k=0}^{n}(x-x_k)}$$

Como $G^{(n+1)}(\xi(x)) = 0$, tenemos que:

$$G^{(n+1)}(\xi(x)) = 0 = f^{(n+1)}(\xi(x)) - \frac{(n+1)!\,(f(x) - P_n(x))}{\prod_{k=0}^{n}(x-x_k)}, \Rightarrow$$

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\prod_{k=0}^{n}(x-x_k)$$

tal como queríamos demostrar.

Ejemplo 4:

Calcular el error cometido al interpolar la función del ejemplo anterior. Recordemos que la función es

$$f(x) = x\sin\left(\frac{\pi}{2}x\right)$$

El error valdrá:

$$f(x) - P_3(x) = \frac{f^{(4)}(\xi(x))}{4!} \cdot x(x-1)(x-3)(x-5)$$

Hallemos $f^{(4)}(x)$:

$$\begin{aligned}
f'(x) &= \sin\left(\frac{\pi}{2}x\right) + \frac{\pi}{2}x\cos\left(\frac{\pi}{2}x\right), \\
f''(x) &= \pi\cos\left(\frac{\pi}{2}x\right) - \frac{\pi^2}{2}x\sin\left(\frac{\pi}{2}x\right), \\
f'''(x) &= -\frac{3\pi^2}{4}\sin\left(\frac{\pi}{2}x\right) - \frac{\pi^3}{8}x\cos\left(\frac{\pi}{2}x\right) \\
f'^{v}(x) &= \frac{\pi^4}{16}x\sin\left(\frac{\pi}{2}x\right) - \frac{\pi^3}{2}\cos\left(\frac{\pi}{2}x\right)
\end{aligned}$$

factorizando $\frac{\pi^3}{16}$ tenemos

$$f'^{v}(x) = \frac{\pi^3}{16}\left[\pi x\sin\left(\frac{\pi}{2}x\right) - 8\cos\left(\frac{\pi}{2}x\right)\right]$$

La expresión del error será:

$$f(x) - P_3(x) = \frac{1}{4!} \cdot \frac{\pi^3}{16}\left[\pi\xi(x)\sin\left(\frac{\pi}{2}\xi(x)\right) - 8\cos\left(\frac{\pi}{2}\xi(x)\right)\right](x)(x-1)(x-3)(x-5)$$

Imaginemos que queremos acotar el error cometido $|f(x) - P_3(x)|$ para todo valor de x es el intervalo $[0,5]$. Entonces tendremos:

$$|f(x) - P_3(x)| \leq \frac{\pi^3(8+5\pi)}{384} \cdot \max_{x\in[0,5]} |x(x-1)(x-3)(x-5)|$$

Para calcular el valor $\max_{x\in[0,5]} |x(x-1)(x-3)(x-5)|$ tenemos que derivar la función

$$\begin{aligned}
h(x) &= x(x-1)(x-3)(x-5) \\
&= x^4 - 9x^3 + 23x^2 - 15x
\end{aligned}$$

ya que el valor máximo de la misma se alcanzará en el interior (en los extremos $h(0) = h(5) = 0$)

$$h'(x) = 4x^3 - 27x^2 + 46x - 15$$

El siguiente paso es resolver la ecuación $h'(x) = 0$. Para resolver la ecuación anterior, podemos usar un método numérico para hallar ceros como los métodos de **Regula-Falsi** o **Newton-Rapson.**

Usando el primer método anterior y la técnica de la deflación, podemos hallar los ceros del polinomio anterior:

$$x_1 = 0.42575862,\ \ x_2 = 2.0704646 \text{ y } x_3 = 4.2537492$$

Como los tres ceros se encuentran en el interior del intervalo $[0, 5]$.

Las imágenes de los ceros anteriores para la función $g(x) = |h(x)|$ es $g(x) = |x(x-1)(x-3)(x-5)|$ son los siguientes

$$\begin{aligned} g(0.42575862) &= 2.8788980 \\ g(2.07046460) &= 6.0353825 \\ g(4.25374920) &= 12.949453 \end{aligned}$$

Entonces $\max_{x\in[0,5]} |x(x-1)(x-3)(x-5)| = 12.949453$.

El error cometido al interpolar $f(x)$ en los nodos anteriores para $x \in [0, 5]$ se puede acotar por:

$$|f(x) - P_3(x)| \leq \frac{\pi^3(8+5\pi)}{384} \cdot 12.949453 \approx 24.7892891$$

La cota del error es muy grande ya que la longitud del intervalo es grande (5) hemos considerado sólo 4 nodos. Si aumentamos el número de nodos, el error disminuirá.

De todas formas, es una cota muy "pesimista". Basta observar el gráfico de la función $f(x)$ y del polinomio interpolador $P_3(x)$ observar que el error no es mayor que 4 unidades.

Si hubiéramos acotado la derivada $f^{(4)}(x)$ hallando el valor máximo dentro del intervalo $[0, 5]$ hubiésemos obtenido una cota más fina.

Ejercicio 7:

Crear un *programa* en **Python** para calcular el error de Interpolación del ejemplo realizado.

4.4 Método de Neville

Muchas de las aplicaciones derivadas de la **interpolación** consisten en la interpolación de **datos tabulados**, es decir, datos que tenemos en una tabla de la forma

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

En estos casos, no necesitamos conocer explícitamente la expresión del **polinomio interpolador**.

En el caso de los **datos tabulados**, al no conocer la función f que interpolamos, nos podemos aplicar la fórmula del **error en la interpolación** y no podemos saber cuál es el error cometido en un valor cualquiera x.

El método de Lagrange para hallar el **polinomio interpolador** tiene el siguiente problema:

- Imaginemos que hemos hallado el polinomio interpolador para los puntos $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$ que forman parte de la gráfica de una cierta función f.

- Con el objetivo de hallar una mejor aproximación de la función f, añadimos un nuevo punto $(x_{n+1}, f(x_{n+1}))$. Entonces para hallar el nuevo polinomio interpolador en los puntos $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n)), (x_n, f(x_{n+1}))$ usando el **Método de Lagrange**, tenemos que "empezar de nuevo" ya que los **Polinomios de Lagrange** hallados previamente para los puntos $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$ no nos sirven y hay que volver a hallar los "nuevos polinomios de Lagrange"

Lo dicho anteriormente hace que el **Método de Lagrange** no sea apropiado para hallar el **Polinomio Interpolador** para datos tabulados.

Vamos a ver un método nuevo que aprovecha el trabajo realizado los puntos $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$ con el objetivo de interpolar los puntos anteriores añadiendo un nuevo punto $(x_{n+1}, f(x_{n+1}))$. Dicho método se llama **Método de Neville.**

En primer lugar necesitamos unas definiciones previas:

Definición 2:

Sea f la función definida en los valores $x_0, x_1, \ldots x_n$. Sean $m_1, m_2, \ldots, m_k, k$ enteros diferentes, con $m_i \in \{0, 1, \ldots, n\}$ para $i = 1, \ldots, k$. El **Polinomio Interpolador de Lagrange** que interpola los puntos $\left(x_{m_1}, f\left(x_{m_1}\right)\right), \left(x_{m_2}, f\left(x_{m_2}\right)\right), \ldots, \left(x_{m_k}, f\left(x_{m_k}\right)\right)$ se denota por $P_{m_1, m_2, \ldots, m_k}(x)$

Ejemplo 5:

Consideremos los puntos $x_0 = 0$, $x_1 = 1$, $x_2 = 3$ y $x_3 = 5$ y la función $f(x) = x \sin\left(\frac{\pi}{2} x\right)$. Los puntos a interpolar son los siguientes: $(0, 0)$, $(1, 1)$, $(3, -3)$ y $(5, 5)$. Hallar el **Polinomio Interpolador** $P_{1,3}(x)$ que sería el polinomio interpolador en los puntos $x_1 = 1$ y $x_3 = 5$.

Los polinomios de Lagrange asociados a los nodos anteriores son los siguientes:

$$L_{1,0}(x) = \frac{x-5}{1-5} = -\frac{1}{4}(x-5)$$
$$L_{1,1}(x) = \frac{x-1}{5-1} = \frac{1}{4}(x-1)$$

El polinomio interpolador será pues:

$$P_{1,3}(x) = 1 \cdot \left(-\frac{1}{4}(x-5)\right) + 5 \cdot \left(\frac{1}{4}(x-1)\right) = x$$

Ejercicio 8:

Con los datos del ejemplo anterior encontrar el **Polinomio Interpolador** $P_{0,2,3}(x)$ que sería el polinomio interpolador en los puntos $x_0 = 0$, $x_2 = 3$ y $x_3 = 5$.

Los polinomios de Lagrange asociados a los nodos anteriores son los siguientes:

$$L_{2,0}(x) = \frac{(x-3)(x-5)}{(0-3)(0-5)} = \frac{1}{15}\left(x^2 - 8x + 1\right)$$
$$L_{2,1}(x) = \frac{(x-0)(x-5)}{(3-0)(3-5)} = -\frac{1}{6}\left(x^2 - 5x\right)$$
$$L_{2,2}(x) = \frac{(x-0)(x-3)}{(5-0)(5-3)} = \frac{1}{10}\left(x^2 - 3x\right)$$

El polinomio interpolador será

$$
\begin{aligned}
P_{0,2,3}(x) &= 0 \cdot L_{2,0}(x) - 3 \cdot L_{2,1}(x) + 5 \cdot L_{2,2}(x) \\
& 0 \cdot \left(\frac{1}{15}\left(x^2 - 8x + 1\right) \right) - 3\left(-\frac{1}{6}\left(x^2 - 5x\right) \right) + 5\left(\frac{1}{10}\left(x^2 - 3x\right) \right) \\
&= x^2 - 4x
\end{aligned}
$$

4.4.1 Polinomio Interpolador

El siguiente resultado nos dice cómo hallar el valor del **Polinomio Interpolador** en un punto x a partir de los **Polinomios Interpoladores de Lagrange** introducidos anteriormente:

Proposición

Sean $x_0, x_1, \ldots, x_n$, $n+1$ nodos, sea f una función y sea $P_n(x)$ el **Polinomio Interpolador** de la función f en los nodos anteriores.

Sean x_i y x_j dos nodos cualquiera con $i \neq j$. Sean $P_{0,1,\ldots,i-1,i+1,\ldots,n}(x)$ y $P_{0,1,\ldots,j-1,j+1,\ldots,n}(x)$ los **Polinomios Interpoladores de Lagrange** en los nodos $x_0, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ y $x_0, x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n$ respectivamente.

Entonces el polinomio interpolador $P_n(x)$ que interpola todos los nodos $x_0, x_1, \ldots, x_n$ puede escribirse de la forma siguiente:

$$
P_n(x) = \frac{(x - x_j)\, P_{0,1,\ldots,j-1,j+1,\ldots,n}(x) - (x - x_i)\, P_{0,1,\ldots,i-1,i+1,\ldots,n}(x)}{x_i - x_j}
$$

Demostración

Para simplificar la notación escribiremos $Q_i(x) = P_{0,1,\ldots,i-1,i+1,\ldots,n}(x)$ y $Q_j(x) = P_{0,1,\ldots,j-1,j+1,\ldots,n}(x)$.

Notemos que $Q_i(x_j) = f(x_j)$ y $Q_j(x_i)$ si $i \neq j$.

Veamos que $P_n(x_k) = f(x_k)$ para cualquier valor de k. Distinguimos los casos siguientes:

❍ $k \neq i, j$, en este caso

$$\begin{aligned} P_n\left(x_k\right) &= \frac{\left(x_k - x_j\right) Q_j\left(x_k\right) - \left(x_k - x_i\right) Q_i\left(x_k\right)}{x_i - x_j} \\ &= \frac{\left(x_k - x_j\right) f\left(x_k\right) - \left(x_k - x_i\right) f\left(x_k\right)}{x_i - x_j} \\ &= \frac{f\left(x_k\right)\left(x_k - x_j - x_k + x_i\right)}{x_i - x_j} = f\left(x_k\right) \end{aligned}$$

❍ $k = i$, en este caso

$$\begin{aligned} P_n\left(x_i\right) &= \frac{\left(x_i - x_j\right) Q_j\left(x_i\right) - \left(x_i - x_i\right) Q_i\left(x_i\right)}{x_i - x_j} \\ &= \frac{\left(x_i - x_j\right) f\left(x_i\right)}{x_i - x_j} = f\left(x_i\right) \end{aligned}$$

$k = j$, en este caso

$$\begin{aligned} P_n\left(x_j\right) &= \frac{\left(x_j - x_j\right) Q_j\left(x_j\right) - \left(x_j - x_i\right) Q_i\left(x_j\right)}{x_i - x_j} \\ &= \frac{-\left(x_j - x_i\right) f\left(x_j\right)}{x_i - x_j} = \frac{\left(x_i - x_j\right) f\left(x_j\right)}{\left(x_i - x_j\right)} \\ &= f\left(x_j\right) \end{aligned}$$

El **Método de Neville** nos permite hallar los distintos polinomios de interpolación de forma recursiva.

❍ Los **Polinomios Interpoladores de Lagrange** de grado 0 serían:

$$P_0\left(x\right) = f\left(x_0\right), P_1\left(x\right) = f\left(x_1\right), \ldots, P_n\left(x\right) = f\left(x_n\right),$$

en general $P_i = f\left(x_i\right),\ i)0, \ldots, n$

❍ La proposición anterior nos da los **Polinomios Interpoladores de Lagrange** de grado 1:

$$\begin{aligned} P_{0,1}\left(x\right) &= \frac{\left(x - x_0\right) P_1\left(x\right) - \left(x - x_1\right) P_0\left(x\right)}{x_1 - x_0} \\ P_{1,2}\left(x\right) &= \frac{\left(x - x_1\right) P_2\left(x\right) - \left(x - x_2\right) P_1\left(x\right)}{x_2 - x_1} \\ &\vdots \\ P_{n-1,n}\left(x\right) &= \frac{\left(x - x_{n-1}\right) P_n - \left(x - x_n\right) P_{n-1}}{x_n - x_{n-1}} \end{aligned}$$

❍ Usando la misma técnica podemos obtener los **Polinomios Interpoladores de Lagrange** de grado 2:

$$P_{0,1,2}(x) = \frac{(x-x_0)\,P_{1,2}(x) - (x-x_2)\,P_{0,1}(x)}{x_2 - x_0}$$

$$P_{1,2,3}(x) = \frac{(x-x_1)\,P_{2,3}(x) - (x-x_3)\,P_{1,2}(x)}{x_3 - x_1}$$

$$\vdots$$

$$P_{n-2,n-1,n}(x) = \frac{(x-x_{n-2})\,P_{n-1,n}(x) - (x-x_n)\,P_{n-2,n-1}(x)}{x_n - x_{n-2}}$$

Los polinomios generados anteriormente se pueden escribir en forma de tabla de la siguiente manera:

x_i	P_i	$P_{i-1,i}$	$P_{i-2,i-1,i}$	$P_{i-3,i-2,i-1,i}$	$\cdots$
x_0	P_0				
x_1	P_1	$P_{0,1}$			
x_2	P_2	$P_{1,2}$	$P_{0,1,2}$		
x_3	P_3	$P_{2,3}$	$P_{1,2,3}$	$P_{0,1,2,3}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

La manera de indicar los **Polinomios de Lagrange Interpoladores** es un poco tediosa ya que el número de subíndices va aumentando el número de nodos a interpolar.

Como los subíndices son consecutivos, basta tener en cuenta el subíndice del nodo en que empieza la interpolación y el número de nodos que se interpola.

Por ejemplo, para indicar el **Polinomio Interpolador de Lagrange** $P_{2,3,4}$ bastaría "guardar" el valor 4, es decir, el último nodo que se interpola x_4 y el número 3 que es el número de nodos que se interpola ya que con estos dos valores ya quedaría claro que los nodos a interpolar serían x_2, x_3 y x_4.

4.4.2 Simplificación de la Notación

Por dicho motivo, introduciremos una nueva notación para indicar los **Polinomios Interpoladores de Lagrange**: para indicar el polinomio $P_{i-j,i-j+1,\ldots,i}$ definimos:

$$Q_{i,j} := P_{i-j,i-j+1,\ldots,i}$$

La tabla anterior sería en la nueva notación:

x_i	$Q_{i,0}$	$Q_{i,1}$	$Q_{i,2}$	$Q_{i,3}$	$\cdots$
x_0	$Q_{0,0}$				
x_1	$Q_{1,0}$	$Q_{1,1}$			
x_2	$Q_{2,0}$	$Q_{2,1}$	$Q_{2,2}$		
x_3	$Q_{3,0}$	$Q_{3,1}$	$Q_{3,2}$	$Q_{3,3}$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Ejemplo

La tabla de **Polinomios Interpoladores de Lagrange** para los datos del ejemplo sería el siguiente

x_i	$Q_{i,0}$	$Q_{i,1}$	$Q_{i,2}$
$x_0 = 0$	$Q_{0,0} = 0$		
$x_1 = 1$	$Q_{1,0} = 1$	$Q_{1,1} = \frac{(x-0)1-(x-1)0}{1-0} = x$	
$x_2 = 3$	$Q_{2,0} = -3$	$Q_{2,1} = \frac{(x-1)(-3)-(x-3)(1)}{3-1}$ $Q_{2,1} = -2x+3$	$Q_{2,2} = \frac{(x-0)(-2x+3)-(x-3)(x)}{3-0}$ $Q_{2,2} = -x^2+2x$
$x_3 = 5$	$Q_{3,0} = 5$	$Q_{3,1} = \frac{(x-3)(5)-(x-5)(-3)}{5-3}$ $Q_{3,1} = 4x-15$	$Q_{3,2} = \frac{(x-1)(4x-15)-(x-5)(-2x+3)}{5-1}$ $Q_{3,2} = 1.5x^2-8x+7.5$

Cuadro 4.1: Tabla de los Polinomios Interpoladores de Lagrange

Calculemos

$$\begin{aligned} Q_{3,3}(x) &= \frac{(x-0)\left(1.5x^2-8x+7.5\right)-(x-5)\left(-x^2+2x\right)}{5-0} \\ &= 0.5x^3-3x^2+3.5x \end{aligned}$$

Ejercicio 9:

Crear un programa en Python que calcule de forma recursiva el **Polinomio Interpolador de Lagrange** usando el **Método de Neville.** Además, realice una gráfica del Polinomio encontrado y la función aproximada $f(x) = x\sin\left(\frac{\pi}{2}x\right)$

Los **Polinomios Interpoladores de Lagrange** rara vez se calculan.

Lo que habitualmente se hace es usar el algoritmo anterior para evaluar el valor de $P_n(x)$ para un valor determinado x.

De esta manera, la tabla de los $Q_{i,j}$ no son polinomios sino valores.

De hecho, el valor de $Q_{i,j}$ es el valor de polinomio interpolador en los valores $(x_0, y_0), \ldots, (x_i, y_i)$ en el punto x.

Por tango, $Q_{n,n}(x)$ es el valor de $P_n(x)$

Ejemplo 6:

Calcular $P_n(2)$ del ejemplo realizado en la tabla 4.1

- $Q_{1,1}(2) = \frac{(2-0)(1)-(2-1)(0)}{1-0} = 2$
- $Q_{2,1}(2) = \frac{(2-1)(-3)-(2-3)(1)}{3-1} = -1$
- $Q_{3,1}(2) = \frac{(2-3)(5)-(2-5)(-3)}{5-3} = -7$
- $Q_{2,2}(2) = \frac{(2-0)(-1)-(2-3)(2)}{3-0} = 0$
- $Q_{3,2}(2) = \frac{(2-1)(-7)-(2-5)(-1)}{5-1} = -\frac{5}{2}$
- $Q_{3,3}(2) = \frac{(2-0)\left(-\frac{5}{2}\right)-(2-5)(0)}{5-0} = -1$

x_i	$Q_{i,0}$	$Q_{i,1}$	$Q_{i,2}$	$Q_{i,3}$
0	0			
1	1	$Q_{1,1}(2) = 2$		
3	-3	$Q_{2,1}(2) = -1$	$Q_{2,2}(2) = 0$	
5	5	$Q_{3,1}(2) = -7$	$Q_{2,3}(2) = -\frac{5}{2}$	$Q_{3,3}(x) = -1$

Cuadro 4.2: Evaluación de los Polinomios Interpoladores de Lagrange para $x = 2$

Ejemplo 7:

Añadir un nuevo punto a los datos del ejemplo anterior, por ejemplo el punto $(6, 0)$.

Para hallar dicho valor podemos aprovechar la tabla anterior 4.2. Solo tenemos que calcular una nueva fila para hallar el valor del polinomio interpolador en $x = 2$.

- $Q_{4,1}(2) = \frac{(2-5)(0)-(2-6)(5)}{6-1} = 20$

- $Q_{4,2}(2) = \frac{(2-3)(20)-(2-6)(-7)}{6-3} = -16$
- $Q_{4,3}(2) = \frac{(2-1)(-16)-(2-6)\left(-\frac{5}{2}\right)}{6-1} = -\frac{26}{5}$
- $Q_{4,4}(2) = \frac{(2-0)\left(-\frac{26}{5}\right)-(2-6)(-1)}{6-0} = -\frac{12}{5}$

x_i	$Q_{i,0}$	$Q_{i,1}$	$Q_{i,2}$	$Q_{i,3}$	$Q_{i,4}$
0	0				
1	1	$Q_{1,1}(2) = 2$			
3	-3	$Q_{2,1}(2) = -1$	$Q_{2,2}(2) = 0$		
5	5	$Q_{3,1}(2) = -7$	$Q_{2,3}(2) = -\frac{5}{2}$	$Q_{3,3}(2) = -1$	
6	0	$Q_{4,1}(2) = 20$	$Q_{4,1}(2) = -16$	$Q_{4,3}(2) = -\frac{26}{5}$	$Q_{4,4}(2) = -\frac{12}{5}$

Cuadro 4.3: Evaluación de los Polinomios Interpoladores de Lagrange para $x = 2$

Observación

Realizando los cálculos anteriores, tenemos que el valor del nuevo polinomio interpolador para $x = 2$ vale -2.4. Para hallar dicho valor, no hace falta empezar de nuevo sino que hemos usado los cálculos de la tabla con los nodos “anteriores”. Esta es la ventaja del **Método de Neville**, es decir, aprovechar los cálculos realizados al añadir un nuevo punto a interpolar.

4.4.3 Pseudocódigo del Método de Neville

```
procedure TMetodoNeville.calcular;
var
  x: array[0..n] of Real;
  y: array[0..n] of Real;
  Q: array[0..n, 0..n] of Real;
begin
  { Crear un ciclo de i=0,...,n para crear los
    polinomios de orden 0 }
  for i:=1 to n do
  begin
```

```
    Q[i,0] = f(x[i]);
  end;

  for i:=1 to n
  begin
    for j:=1 to i do
    begin
      { Calculamos los valores Q_{i,j} de la Tabla}
      Q[i,j] := ((x-x[i-j])*Q[i,j-1]-(x-x[i])*Q[i-1,j
        -1])/(x[i] - x[i-j]);
    end;
  end;

  Console.log('Método de Neville', 'Tabla', Q);
end;
```

Ejercicio 10:

Crear un *programa* en **Python** que calcule la tabla del **Método de Neville**. Se deben ingresar los nodos a interpolar y el valor de x que se quiere evaluar.

4.5 Método de Newton

El **Método de Neville** anteriormente descrito se usa básicamente para hallar el valor del **Polinomio Interpolador** $P_n(x)$ en un valor concreto x.

El método de las **Diferencias Divididas** permite hallar el **Polinomio Interpolador** $P_n(x)$ de la función f en los nodos $x_0, x_1, \ldots, x_n$ expresando $P_n(x)$ de la forma siguiente:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1})$$

es decir es escribimos el polinomio interpolador como combinación lineal de los elementos de la *base siguiente del espacio vectorial de polinomios de grado* n:

$$1, (x - x_0), (x - x_0)(x - x_1), \ldots, (x - x_0)\cdots(x - x_{n-1})$$

Calculemos los primeros coeficientes a_i.

- Como $P_n(x_0) = f(x_0)$, deducimos que $a_0 = f(x_0)$
- Como $P_n(x_1) = a_0 + a_1(x_1 - x_0) = f(x_1)$, deducimos $a_1 = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$

Para calcular los demás coeficientes a_i, necesitamos introducir las diferencias divididas de cierto orden:

Definición 3: Definición de Diferencias Divididas

- Las diferencias divididas de orden 0 valen $f[x_i] = f(x_i)$
- Las diferencias divididas de orden 1 valen $f[x_i, x_{i+1}] = \dfrac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$
- Las diferencias divididas de orden 2 valen $f[x_i, x_{i+1}, x_{i+2}] = \dfrac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i}$

En general, las diferencias divididas de orden k se definen en función de las diferencias divididas de orden $k-1$

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

4.5.1 Polinomio Interpolador

Entonces tenemos el siguiente teorema:

Teorema 5:

El polinomio interpolador $P_n(x)$ de la funcion f en los nodos $x_0, x_1, \ldots, x_n$ vale en función de las diferencias divididas:

$$\begin{aligned} P_n(x) &= f[x_0] + f[x_0, x_1](x - x_0) + \cdots \\ &\quad + f[x_0, \ldots, x_n](x - x_0)\cdots(x - x_{n-1}), \\ &= f[x_0] + \sum_{k=1}^{n} f[x_0, \ldots, x_k](x - x_0)\cdots(x_k - x_{k-1}) \end{aligned}$$

Observación

El polinomio interpolador también puede escribirse como:

$$\begin{aligned} P_n(x) &= f[x_n] + f[x_n, x_{n-1}](x - x_n) + f[x_n, x_{n-1}, x_{n-2}](x - x_n)(x - x_{n-1}) \cdots \\ &\quad + f[x_n, \ldots, x_0](x - x_n)(x - x_{n-1}) \cdots (x - x_1), \\ &= f[x_n] + \sum_{k=n-1}^{0} f[x_n, \ldots, x_k](x - x_n)(x - x_{n-1}) \cdots (x - x_{k+1}) \\ &= f[x_n] + \sum_{k=0}^{n-1} f[x_k, \ldots, x_n](x - x_{k+1}) \cdots (x - x_{n-1})(x - x_n) \end{aligned}$$

- Orden 1
  - Para $x_0,\ f[x_0, x_1] = \dfrac{f(x_1) - f(x_0)}{x_1 - x_0}$
  - Para $x_1,\ f[x_1, x_2] = \dfrac{f(x_2) - f(x_1)}{x_2 - x_1}$
  - Para $x_2,\ f[x_2, x_3] = \dfrac{f(x_3) - f(x_2)}{x_3 - x_2}$
- Orden 2
  - Para $x_1,\ f[x_0, x_1, x_2] = \dfrac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$
  - Para $x_2,\ f[x_1, x_2, x_3] = \dfrac{f[x_2, x_3] - f[x_1, x_3]}{x_3 - x_1}$
- Orden 3
  - Para $x_1,\ f[x_0, x_1, x_2, x_3] = \dfrac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0}$
  - Para $x_2,\ f[x_1, x_2, x_3, x_4] = \dfrac{f[x_2, x_3, x_4] - f[x_1, x_3, x_4]}{x_4 - x_1}$

4.5.2 Pseudocódigo

- Damos los valores de los nodos x_i y sus valores a interpolar $f(x_i)$ para $i = 0, 1, \ldots, n$
- Input $x_0, x_1, \ldots, x_n$
- Input $f(x_0), f(x_1), \ldots, f(x_n)$

- Para $i = 0, \ldots, n$
  - $F_{i,0} = f(x_i)$ (Definimos $F_{i,j} = f[x_{i-j}, \ldots, x_i]$. Los coeficientes $a_i = f[x_0, \ldots, x_n]$ ser+an $F_{i,j}$. Por tanto, $F_{i,0} = f[x_i] = f(x_i)$
- Para $i = 1, \ldots, n$
  - Para $j = 1, \ldots, i$
    - Δ $F_{i,j} = \dfrac{F_{i,j-1} - F_{i-1,j-1}}{x_i - x_{i-j}}$. Calculamos $F_{i,j} = f[x_{i-j}, \ldots, x_i]$ en función de $F_{i,j-1} = f[x_{i-j+1}, \ldots, x_i]$ y $F_{i-1,j-1} = f[x_{i-j}, \ldots, x_{i-1}]$
- Imprimir $F_{0,0}, \ldots, F_{n,n}$ (Damos los coeficientes $a_0, \ldots, a_n$

Ejemplo 8:

Calculemos el polinomio interpolador en los nodos anteriores $x_0 = 0$, $x_1 = 1$, $x_2 = 3$, $x_3 = 5$ para la función

$$f(x) = x \cdot \sin\left(\frac{\pi}{2}x\right)$$

Los valores $f(x_i)$, $i = 0, 1, 2, 3$ valen $0, 1, -3, 5$.

x_i	$f(x_i)$	Orden 1	Orden 2	Orden 3
0	0			
		$f[0,1] = \dfrac{1-0}{1-0} = 1$		
1	1		$f[x_0, x_1, x_2] = \dfrac{-2-1}{3-0} = -1$	
		$f[1,3] = \dfrac{-3-1}{3-1} = -2$		$f[0,1,3,5] = \dfrac{1.5-(-1)}{5-0} = 0.5$
3	-3		$f[x_1, x_2, x_3] = \dfrac{4+2}{5-1} = 1.5$	
		$f[3,5] = \dfrac{5+3}{5-3} = 4$		
5	5			

El polinomio interpolador será:

$$\begin{aligned} P_3(x) &= 0 + 1(x-0) - 1(x-0)(x-1) + 0.5(x-0)(x-1)(x-3) \\ &= x - x(x-1) + 0.5x(x-1)(x-3) \end{aligned}$$

Tal como hemos indicado, dicho polinomio también puede escribirse como:

$$P_3(x) = 2 + 4(x-5) + 1.5(x-5)(x-3) + 0.5(x-5)(x-3)(x-1)$$

La implementación puede verse en el siguiente archivo:

Archivo de Programa

El archivo de programa se encuentra en la ruta: *Falta crear el archivo*

4.5.3 Diferencias Divididas

El teorema siguiente nos da la relación entre las diferencias divididas y la función que interpolados f:

Teorema 6:

Sea $f \in \mathcal{C}^n$ una función de clase $\mathcal{C}^n$ en un intervalo $[a, b]$ que contenga los nodos a interpolar $x_0, x_1, \ldots, x_n$. Entonces existe un número $\xi \in (a, b)$ tal que

$$f[x_0, x_1, \ldots, x_n] = \frac{f^{(n)(\xi)}}{n!}$$

Demostración

Sea $g(x) = f(x) - P_n(x)$, donde $P_n(x)$ es el polinomio interpolador en los puntos $(x_i, f(x_i))$ para $i = 0, 1, \ldots, n$. Como $f(x_i) = P_n(x_i)$ para $i = 0, 1, \ldots, n$ la función g tiene $n+1$ ceros distintos en $[a, b]$. Aplicando el Teorema de Roller Generalizado, tenemos que existe un $\xi \in (a, b)$ tal que $g^{(n)}(\xi) = 0$.

Ahora bien, como $P_n(x)$ es un polinomio de grado n, su derivada n-ésima será constante de valor

$$P_n^{(n)}(x) = n!$$

coeficiente principal $n!f\left[x_0, x_1, \ldots x_n\right]$. Por tanto,

$$g^{(n)}\left(\xi\right) = f^{(n)}\left(\xi\right) - P_n^{(n)}\left(\xi\right) - n!f\left[x_0, x_1, \ldots, x_n\right] = 0$$

entonces

$$f\left[x_0, x_1, \ldots, x_n\right] = \frac{f^{(n)}\left(\xi\right)}{n!},$$

como queríamos demostrar.

4.6 Nodos equiespaciados

Supongamos que los nodos son equiespaciados, esto es, que la diferencia entre dos nodos consecutivos es constante, o $x_i - x_{i-1} = h$, donde h es independiente del nodo considerado.

En este caso, podemos escribir los nodos de la siguiente manera, suponiendo que el primero x_0 es el menor de todos ellos:

$$x_i = x_0 + i \cdot h,\ i = 0, 1, \ldots, n$$

Así por ejemplo, $x_1 = x_0 + h$, $x_2 = x_0 + 2h$, y así sucesivamente hasta llegar a $x_n = x_0 + n \cdot h$.

Vamos a hallar una expresión del polinomio interpolador en los nodos anteriores para una determinada función f en una forma simplificada.

Recordemos que el polinomio interpolador se escribía como:

$$P_n\left(x\right) = f\left[x_0\right] + \sum_{k=1}^{n} f\left[x_0, \ldots, x_k\right]\left(x - x_0\right)\cdots\left(x - x_{k-1}\right)$$

Sea x un valor cualquiera. Asociamos a este valor x un valor s tal que $x = x_0 + s \cdot h$, o, si se quiere $s = \dfrac{x - x_0}{h}$. El polinomio interpolador anterior valdrá en función de s:

$$\begin{aligned}
P_n\left(x\right) &= P_n\left(x_0 + sh\right) \\
&= f\left[x_0\right] + \sum_{k=1}^{n} f\left[x_0 + sh - x_0\right]\cdots\left(x_0 + sh - \left(x_0 + \left(k-1\right)h\right)\right), \\
&= f\left[x_0\right] + \sum_{k=1}^{n} f\left[x_0, \ldots, x_k\right] s\left(s-1\right)\cdots\left(x - k + 1\right) \cdot h^k
\end{aligned}$$

Si definimos el número binomial $\binom{s}{k}$ como $\binom{s}{k} = \frac{s\left(s-1\right)\cdots\left(x-k+1\right)}{k!}$, podemos escribir el polinomio interpolador en los nodos equiespaciados como:

$$P_n\left(x\right) = f\left[x_0\right] + \sum_{k=1}^{n} \binom{s}{k} k! h^k f\left[x_0, \ldots, x_k\right],$$

donde $s = \frac{x - x_0}{h}$.

A continuación, vamos a escribir la parte $f\left[x_0, \ldots, x_k\right]$ de una forma más compacta usando las diferencias hacia adelante introducidas en el capítulo de ceros cuando se explicó el Método de Aitken.

4.6.1 Diferencias hacia adelante

Definición 4: Definición de diferencias hacia adelante

Sean $x_0, x_1 \ldots, x_n$, $n+1$ nodos equiespaciados y f una función que queremos interpolar. Definimos las diferencias $\Delta^k f\left(x_i\right)$ de forma recurrente de la siguiente manera:

$$\Delta^0 f\left(x_i\right) = f\left(x_i\right), i = 0, \ldots, n, \Delta^k f\left(x_i\right) = \Delta^{k-1} f\left(x_{i+1}\right) - \Delta^{k-1} f\left(x_i\right)$$

Así por ejemplo,

$$\begin{aligned}
\Delta^0 f\left(x_0\right) &= f\left(x_0\right), \\
\Delta^1 f\left(x_0\right) &= \Delta f\left(x_0\right) = f\left(x_1\right) - f\left(x_0\right), \\
\Delta^2 f\left(x_0\right) &= \Delta f\left(x_1\right) - \Delta f\left(x_0\right) = f\left(x_2\right) - f\left(x_1\right) - \left(f\left(x_0\right) - f\left(x_0\right)\right) \\
&= f\left(x_2\right) - 2f\left(x_1\right) + f\left(x_0\right)
\end{aligned}$$

Los valores de $f\left[x_0, x_1\right]$ y $f\left[x_0, x_1, x_2\right]$ serán en función de las diferencias hacia adelante $\Delta f\left(x_0\right)$ y $\Delta^2 f\left(x_0\right)$:

$$\begin{aligned}
f\left[x_0, x_1\right] &= \frac{f\left(x_1\right) - f\left(x - x_0\right)}{x_1 - x_0} = \frac{1}{h}\Delta f\left(x_0\right), \\
f\left[x_0, x_1, x_2\right] &= \frac{f\left[x_1, x_2\right] - f\left[x_0, x_1\right]}{x_2 - x_0} = \frac{\frac{1}{h}\left(\Delta f\left(x_1\right) - \Delta f\left(x_0\right)\right)}{2h} \\
&= \frac{1}{2h^2}\Delta^2 f\left(x_0\right)
\end{aligned}$$

En general, puede demostrarse por inducción que:

$$f\left[x_0, x_1, \ldots, x_k\right] = \frac{1}{k!h^k}\Delta^k f\left(x_0\right)$$

Usando la expresión anterior, el polinomio de interpolación de la función f en los nodos equiespaciados se puede escribir como:

$$P_n\left(x\right) = f\left[x_0\right] + \sum_{k=1}^{n} \binom{s}{k} \Delta^k f\left(x_0\right)$$

4.6.2 Diferencias hacia atrás

Vamos a hacer lo mismo pero en lugar de introducir diferencias hacia adelante, introduciremos las diferencias hacia atrás:

Definición 5: Definición de diferencias hacia atras

Sean $x_0, x_1, \ldots, x_n$, $n+1$ nodos equiespaciados y f una función que queremos interpolar. Definimos las diferencias $\nabla^k f\left(x_i\right)$ de forma recurrente de la siguiente manera:

$$\begin{aligned}\nabla^0 f\left(x_i\right) &= f\left(x_i\right),\ i = 0, \ldots, n,\\ \nabla^n f\left(x_i\right) &= \nabla^{k-1} f\left(x_i\right) - \nabla^{k-1} f\left(x_{i-1}\right)\end{aligned}$$

Los valores de $f\left[x_n, x_{n-1}\right]$ y $f\left[x_n, x_{n-1}, x_{n-2}\right]$ serán en función de las diferencias hacia atrás $\nabla f\left(x_n\right)$ y $\nabla^2 f\left(x_n\right)$:

$$\begin{aligned}f\left[x_n, x_{n-1}\right] &= \frac{f\left(x_{n-1}\right) - f\left(x_n\right)}{x_{n-1} - x_n} = \frac{1}{h}\nabla f\left(x_n\right),\\ f\left[x_n, x_{n-1}, x_{n-2}\right] &= \frac{f\left[x_{n-1}, x_{n-2}\right] - f\left[x_n, x_{n-1}\right]}{x_{n-2} - x_n}\\ &= \frac{\frac{1}{h}\left(\nabla f\left(x_{n-1}\right) - \nabla f\left(x_n\right)\right)}{-2h}\\ &= \frac{1}{2h^2}\nabla^2 f\left(x_n\right)\end{aligned}$$

En general, puede demostrarse por inducción que:

$$f\left[x_n, x_{n-1}, \ldots, x_k\right] = \frac{1}{(n-k)!h^{n-k}}\nabla^{n-k} f\left(x_n\right)$$

Recordemos que el polinomio de interpolación de la función f en los nodos equiespaciados se podía expresar como:

$$P_n(x) = f[x_n] + \sum_{k=0}^{n-1} f[x_k, \ldots, x_n](x - x_{k+1}) \cdots (x -_{xn-1})(x - x_n)$$

Dado un valor x, sea s tal que $x = x_n - sh$, donde $h = x_i - x_{i-1}$. Entonces $x_i = x_n - (n-i)h$ y $x - x_i = x_n - sh - (x_n - (n-i)h) = (n-i-s)h$

La productoria

$$\prod_{k,n} = (x - x_{k+1}) \cdots (x - x_{n-1})(x - x_n)$$

se puede expresar como

$$\begin{aligned}
\prod_{k,n} &= (b - (k+1) - s) \cdots (n - (n-1) - s)(n - n - s) h^{n-k} \\
&= (-1)^{n-k} h^{n-k} s(s-1) \cdots (x - n + k + 1) \\
&= (-1)^{n-k} h^{n-k} (n-k)! \binom{s}{n-k}
\end{aligned}$$

El polinomio interpolador será

$$\begin{aligned}
P_n(x) &= f[x_n] + \sum_{k=0}^{n-1} f[x_k, \ldots x_n](-1)^{n-k} h^{n-k} (n-k)! \binom{s}{n-k} \\
&= f[x_n] + \sum_{k=0}^{n-1} (-1)^{n-k} \binom{s}{n-k} \nabla^{n-k} f(x_n) \\
&= f[x_n] + \sum_{k=1}^{n} (-1)^{k} \binom{s}{k} \nabla^{k} f(x_n)
\end{aligned}$$

Ejemplo 9:

Consideremos la función $f(x) = \dfrac{1}{1 + \cos^2\left(\frac{\pi}{2}x\right)}$ y $n+1$ nodos equiespaciados en el intervalo $[-1, 1]$:

$$x_0 = -1, x_1 = -1 + \frac{2}{n}, \ldots x_n = 1$$

en general $x_i = -1 + \dfrac{2^i}{n}, i = 0, 1, \ldots n$

Consideremos por ejemplo $n = 5$. Los nodos serán -1, -0.6, -0.2, 0.2, 0.6, 1

Hallemos el polinomio interpolador usando diferencias hacia adelante.

Los valores de $\Delta^k f(x_0) = \Delta^k f(-1)$, para $k = 1, 2, 3, 4, 5$ son los siguientes:

- $k = 1$

$$\begin{aligned}
\Delta f(-1) &= f(-0.6) - f(-1) = 0.743228 - 1 = -0.2567772 \\
\Delta(-0.6) &= f(-0.2) - f(-0.6) = 0.5250699 - 0.7432228 = -0.218153 \\
\Delta(-0.2) &= f(0.2) - f(-0.2) = 0.5250699 - 0.5250699 = 0 \\
\Delta(0.2) &= f(0.6) - f(0.2) = 0.7432228 - 0.520699 = 0.2567772 \\
\Delta(0.6) &= f(1) - f(0.6) = 1 - 0.7432228 = 0.2567772
\end{aligned}$$

- $k = 2$

$$\begin{aligned}
\Delta^2 f(-1) &= \Delta f(-0.6) - \Delta f(-1) = -0.218153 - (-0.2567772) = 0.0386242 \\
\Delta^2 f(-0.6) &= \Delta f(-0.2) - \Delta f(-0.6) = 0 - (-0.218153) = 0.218153 \\
\Delta^2 f(-0.2) &= \Delta f(0.2) - \Delta f(-0.2) = 0.518153 - 0 = 0.218153 \\
\Delta^2 f(0.2) &= \Delta f(0.6) - \Delta f(0.2) = 0.2567772 - (0.218153) = 0.0356242
\end{aligned}$$

- $k = 3$

$$\begin{aligned}
\Delta^3 f(-1) &= \Delta^2 f(-0.6) - \Delta^2 f(-1) = 0.218153 - 0.0386242 = 0.1795288 \\
\Delta^3 f(-0.6) &= \Delta^2 f(-0.2) - \Delta^2 f(-0.6) = 0.218153 - 0.218153 = 0 \\
\Delta^3 f(-0.2) &= \Delta^2 f(0.2) - \Delta^2 f(-0.2) = 0.0386242 - 0.218153 = -0.1795288
\end{aligned}$$

- $k = 4$

$$\begin{aligned}
\Delta^4 f(-1) &= \Delta^3 f(-0.6) - \Delta^3 f(-1) = 0 - 0.1795288 = -0.1795288 \\
\Delta^4 f(-0.6) &= \Delta^3 f(-0.2) - \Delta^3 f(-0.6) = -0.1795288 - 0 = -0.1795288
\end{aligned}$$

- $k = 5$

$$\Delta^5 f(-1) = \Delta^4 f(-0.6) - \Delta^4 f(-1) = -0.1795288 - (-0.1795288) = 0$$

Los valores $\binom{s}{k}$ para $k = 1, 2, 3, 4, 5$ valen:

$$\begin{aligned}
\binom{s}{1} &= s, \\
\binom{s}{2} &= \frac{s\left(s-1\right)}{2}, \\
\binom{s}{3} &= \frac{s\left(s-1\right)\left(s-2\right)}{6}, \\
\binom{s}{4} &= \frac{s\left(s-1\right)\left(s-2\right)\left(s-3\right)}{24} \\
\binom{s}{4} &= \frac{s\left(s-1\right)\left(s-2\right)\left(s-3\right)\left(s-4\right)}{120}
\end{aligned}$$

El polinomio interpolador será:

$$\begin{aligned}
P_5\left(x\right) = {} & f\left(-1\right) + s \cdot \Delta f\left(-1\right) + \frac{s\left(s-1\right)}{2} \cdot \Delta^2 f\left(-1\right) + \frac{s\left(s-1\right)\left(s-2\right)}{6} \cdot \Delta^3 f\left(-1\right) \\
& + \frac{s\left(s-1\right)\left(s-2\right)\left(s-3\right)}{24} \cdot \Delta^4 f\left(-1\right) + \frac{s\left(s-1\right)\left(s-2\right)\left(s-3\right)\left(s-4\right)}{120} \cdot \Delta^5 f\left(-1\right) \\
\\
& = 1 - 0.2567772s + \frac{0.0386242}{2} s\left(s-1\right) + \frac{0.17925822}{6} s\left(s-1\right)\left(s-2\right) \\
& - \frac{0.1795288}{24} s\left(s-1\right)\left(s-2\right)\left(s-3\right) + \frac{0}{120} s\left(s-1\right)\left(s-2\right)\left(s-3\right)\left(s-4\right) \\
\\
& = 1 - 0.2567772s + 0.0193121s\left(s-1\right) + 0.0299215s\left(s-1\right)\left(s-2\right) \\
& - 0.0074804s\left(s-1\right)\left(s-2\right)\left(s-3\right)
\end{aligned}$$

donde $s = \dfrac{x-\left(-1\right)}{0.4} = 2.5\left(x+1\right)$