Interpolación

Objetivos

1. Programar el método de interpolación de Gregory-Newton con diferencias hacia delante y hacia atrás.
2. Programar el método de interpolación de Gauss con diferencias centradas hacia delante y hacia atrás
3. Programar el método de interpolación de Stirling con diferencias centradas promediadas.

Interpolación de Gregory-Newton

Diferencias hacia delante

La interpolación de Gregory-Newton con diferencias hacia delante a través de $n+1$ puntos en un intervalo $[a,b]$ está dada por la función $P_n(x)$ con centros en $x_0, x_1, \ldots, x_{n-1}$ equidistantes con distancia h y dónde:

$$h = x_{i+1} - x_i \tag{1}$$

es:

$$P_n(x) = f(x_0) + \Delta f(x_0)(s) + \frac{\Delta^2 f(x_0)}{2!}(s)(s-1) + \cdots + \frac{\Delta^n f(x_0)}{n!}(s)(s-1)\ldots(s-n+1) \tag{2}$$

dónde el operador de diferencias hacia delante es:

$$\Delta f(x_i) = f(x_{i+1}) - f(x_i) \tag{3}$$

$$\Delta^m f(x_i) = \Delta^{m-1} f(x_{i+1}) - \Delta^{m-1} f(x_i) \tag{4}$$

y el punto a evaluar x puede ser representado como $x = x_0 + sh$ con $s \in \mathbb{R}$

Algoritmo

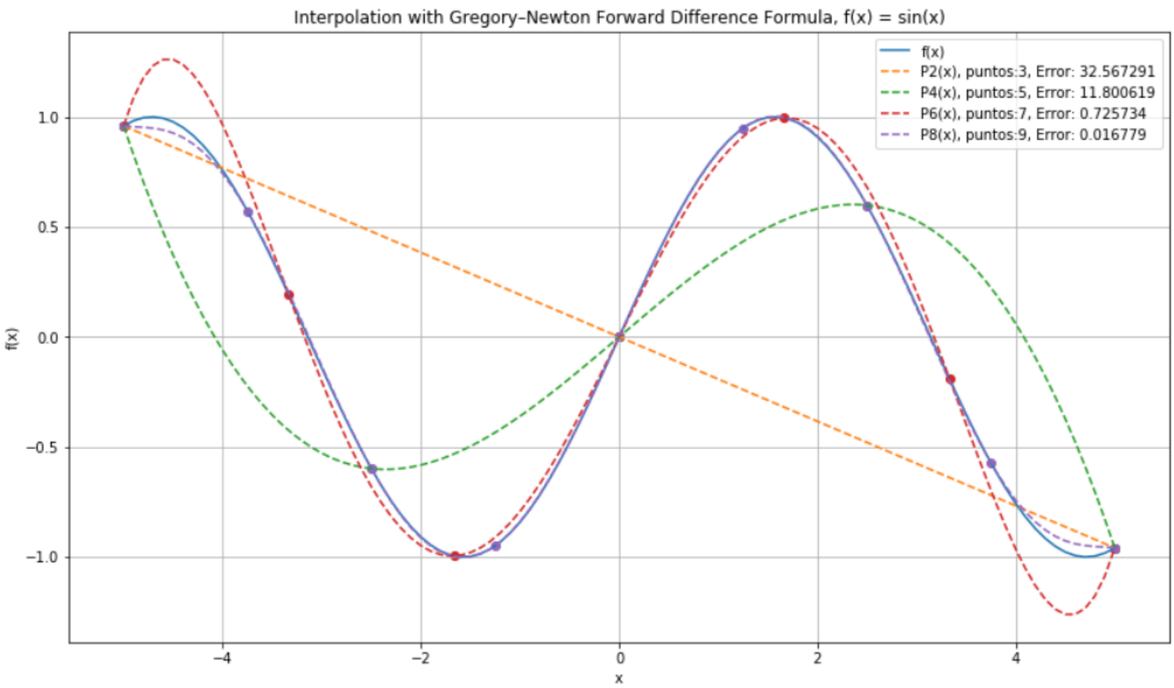
```
n <- Grado del polinomio
x[n+1] <- vector con las coordenadas en x de los n+1 puntos
y[n+1] <- vector con las coordenadas en y de los n+1 puntos

def get_forward_differences(x, y, n):
    '''Retorna los coeficientes de las diferencias
       usados para la evaluación del polinomio'''
    diff_table = [list(y), list(y)]
    diffs = [y[0]]
    size = n
    row = 0
    for i in range(n-1):
        index = 0
        for j in range(size - 1):
```

```python
            diff_table[(row+1)%2][index] = (diff_table[row][j+1] - diff_table[row][j])
            index += 1
        diffs.append(diff_table[(row+1)%2][0])
        size -= 1
        row = (row + 1) % 2

    return diffs

def gregory_newton_fw_evaluation(coefs, size, s, it = 0, fact=1):
    '''Retorna la evaluación del polinomio en un punto dado'''
    if size == 1:
        return coefs[0] / fact
    return coefs[0]/fact + (s - it) * gregory_newton_fw_evaluation(coefs[1:], size - 1, s, it+1, fact * (it+1))

```

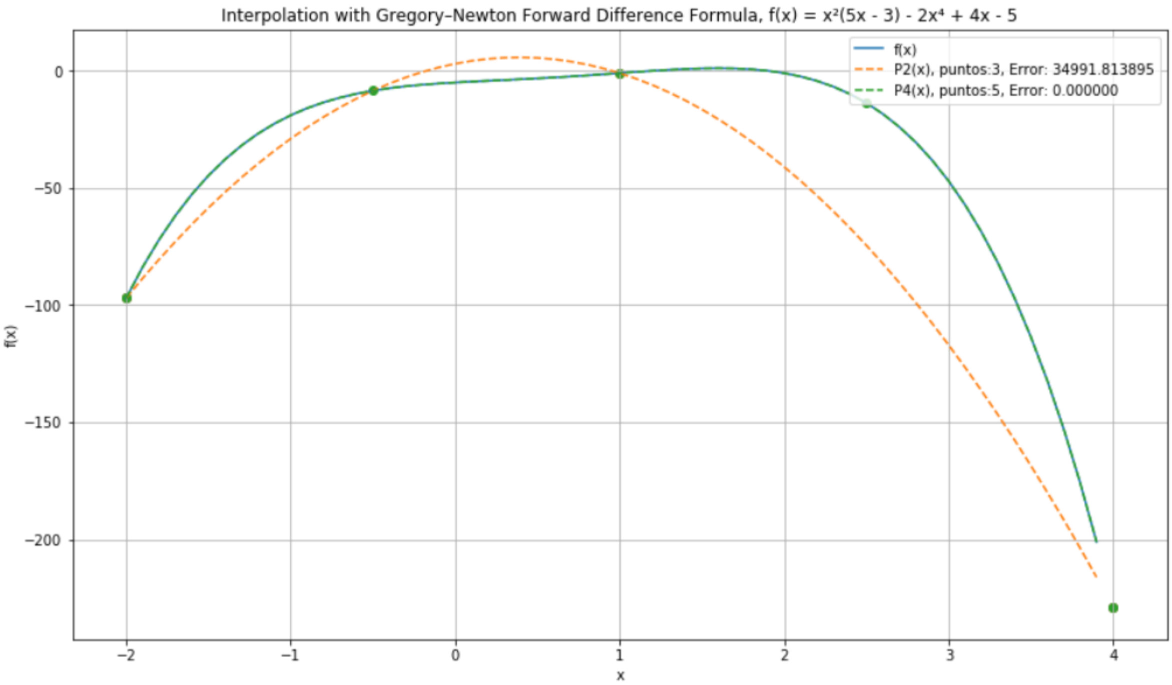
Resultados

Se presentan los resultados obtenidos para diferentes aproximaciones a las siguientes funciones:

- Función $f(x) = sin(x)$,
- Rango de evaluación: $(-5, 5)$
- Número de puntos para interpolación: 3, 5, 7 y 9



- Función: $f(x) = x^2(5X - 3) - 2X^4 + 4X - 5$
- Rango de evaluación: $(-2, 4)$
- Número de puntos para interpolación: 3, y 5



Para el cálculo del error se utilizó el Error Cuadrático Medio con puntos en el intervalo en incrementos de 0.1. Los puntos son equidistantes en el intervalo de cada gráfica.

Diferencias hacia atrás

La función $P_n(x)$ que interpola los puntos antes descritos usando diferencias hacia atrás es:

$$P_n(x) = f(x_n) + \nabla f(x_n)(s) + \frac{\nabla^2 f(x_n)}{2!}(s)(s+1) + \nabla + \frac{\nabla^n f(x_n)}{n!}(s)(s+1)\dots(s+n-1) \tag{5}$$

dónde el operador de diferencias hacia atrás es:

$$\nabla f(x_i) = f(x_i) - f(x_{i-1}) \tag{6}$$

$$\nabla^m f(x_i) = \nabla^{m-1} f(x_i) - \nabla^{m-1} f(x_{i-1}) \tag{7}$$

Algoritmo

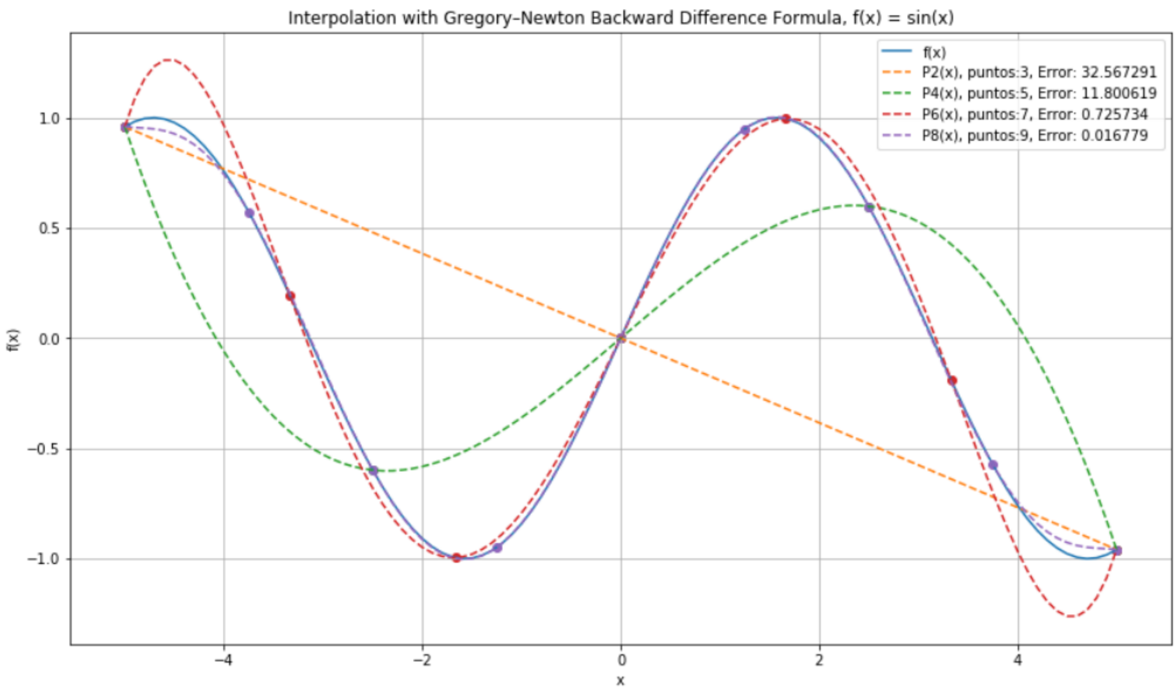
```
n <- Grado del polinomio
x[n+1] <- vector con las coordenadas en x de los n+1 puntos
y[n+1] <- vector con las coordenadas en y de los n+1 puntos

def get_backward_differences(x, y, n):
    '''Retorna los coeficientes de las diferencias
        usados para la evaluación del polinomio'''
    diff_table = [list(y), list(y)]
    diffs = [y[-1]]
    size = n
    row = 0
    for i in range(n-1):
        index = 0
        for j in range(size - 1):
            diff_table[(row+1)%2][index] = (diff_table[row][j+1] - diff_table[row][j])
            index += 1
        diffs.append(diff_table[(row+1)%2][index-1])
        size -= 1
```

```
        row = (row + 1) % 2

    return diffs

def gregory_newton_fw_evaluation(coefs, size, s, it = 0, fact=1):
    '''Retorna la evaluación del polinomio en un punto dado'''
    if size == 1:
        return coefs[0] / fact
    return coefs[0]/fact + (s + it) * gregory_newton_bw_evaluation(coefs[1:], size - 1, s, it+1, fact * (it+1))

```

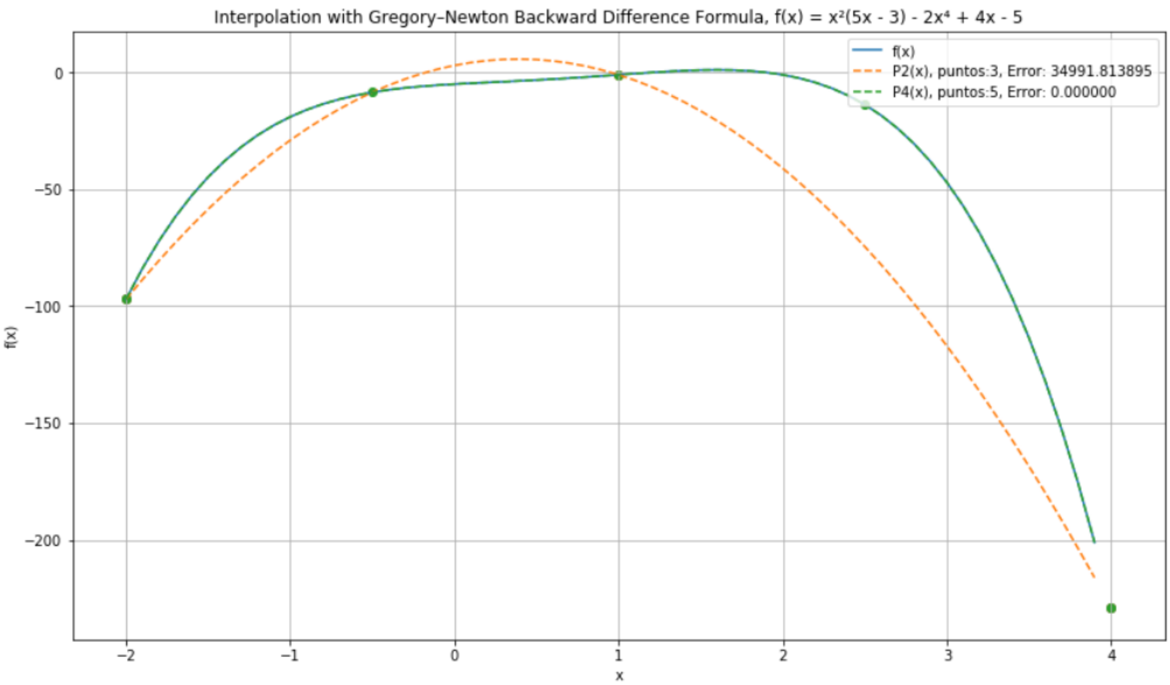
Resultados

Se presentan los resultados obtenidos para diferentes aproximaciones a las siguientes funciones:

- Función $f(x) = sin(x)$,
- Rango de evaluación: $(-5, 5)$
- Número de puntos para interpolación: 3, 5, 7 y 9



- Función: $f(x) = x^2(5X - 3) - 2X^4 + 4X - 5$
- Rango de evaluación: $(-2, 4)$
- Número de puntos para interpolación: 3, y 5



Para el cálculo del error se utilizó el Error Cuadrático Medio con puntos en el intervalo en incrementos de 0.1. Los puntos son equidistantes en el intervalo de cada gráfica.

Interpolación de Gauss

Diferencias centradas hacia adelante

La interpolación de Gauss con diferencias centradas hacia delante a través de $n+1$ puntos en un intervalo $[a, b]$ está dada por la función $P_n(x)$ con centros en $x_0, x_1, \ldots, x_{n-1}$ equidistantes con distancia h es:

$$\begin{aligned} P_n(x) = f(x_0) + \delta f(x_{\frac{1}{2}})(s) + \frac{\delta^2 f(x_0)}{2!}(s)(s-1) + \frac{\delta^3 f(x_{\frac{1}{2}})}{3!}(s)(s-1)(s+1) + \\ \frac{\delta^4 f(x_0)}{4!}(s)(s-1)(s+1)(s-2) + \frac{\delta^5 f(x_{\frac{1}{2}})}{5!}(s)(s-1)(s+1)(s-2)(s+2) + \ldots \end{aligned} \tag{8}$$

dónde el operador de diferencias centradas es:

$$\delta f(x_{i+1/2}) = f(x_{i+1}) - f(x_i) \tag{9}$$

$$\begin{aligned} \delta^m f(x_{i-1/2}) = \delta^{m-1} f(x_i) - \delta^{m-1} f(x_{i-1}),\ si\ m\ es\ impar \\ \delta^m f(x_i) = \delta^{m-1} f(x_{i+1/2}) - \delta^{m-1} f(x_{i-1/2}),\ si\ m\ es\ par \end{aligned} \tag{10}$$

Algoritmo

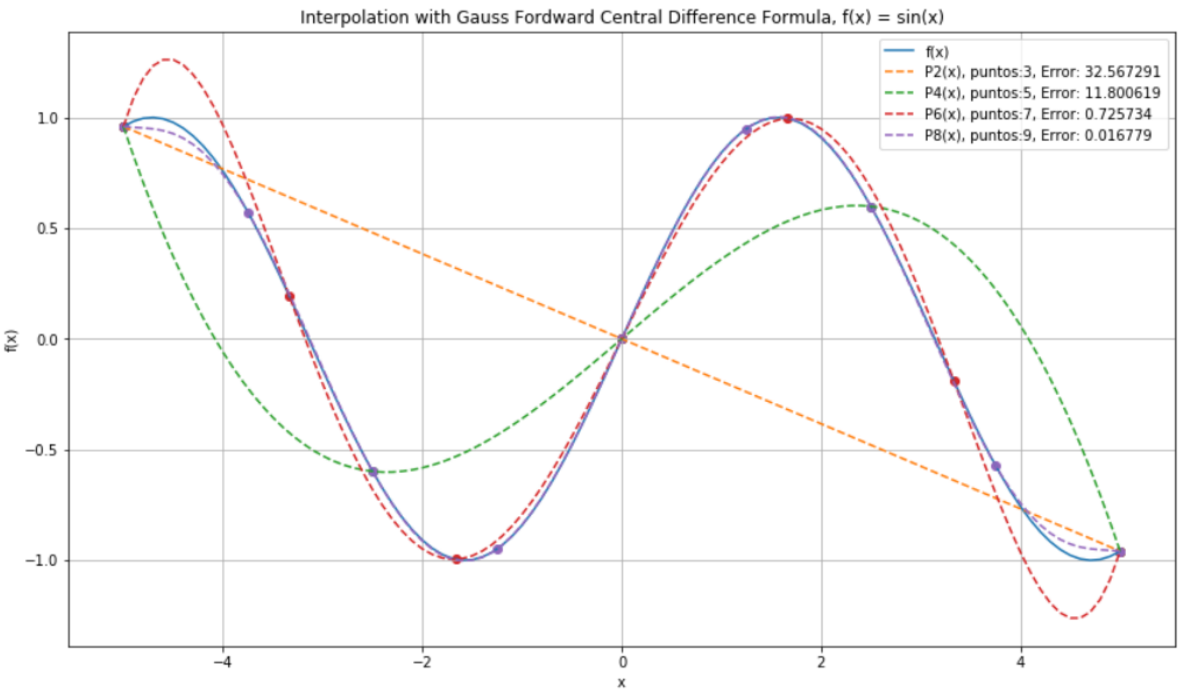
```
n <- Grado del polinomio
x[n+1] <- vector con las coordenadas en x de los n+1 puntos
y[n+1] <- vector con las coordenadas en y de los n+1 puntos

def get_forward_central_differences(x, y, n):
    '''Retorna los coeficientes de las diferencias
        usados para la evaluación del polinomio'''
    diff_table = [list(y), list(y)]
    size = n
    diffs = [y[int(size / 2)]]
    row = 0
    for i in range(n-1):
```

```python
        index = 0
        for j in range(size - 1):
            diff_table[(row+1)%2][index] = (diff_table[row][j+1] - diff_table[row][j])
            index += 1
        size -= 1
        diffs.append(diff_table[(row+1)%2][int(size / 2)])
        row = (row + 1) % 2
    return diffs

def gauss_bw_central_diff_evaluation(x, coefs, size, s):
    '''Retorna la evaluación del polinomio en un punto dado'''
    fact = 1.0
    result = coefs[0]
    s_prod = 1.0
    less = 0.0
    for i in range(size-1):
        if  i & 1:
            less += 1
            s_prod *= s - less
        else:
            s_prod *= s + less
        result += s_prod * coefs[i+1] / fact
        fact *= i+2
    return result
```

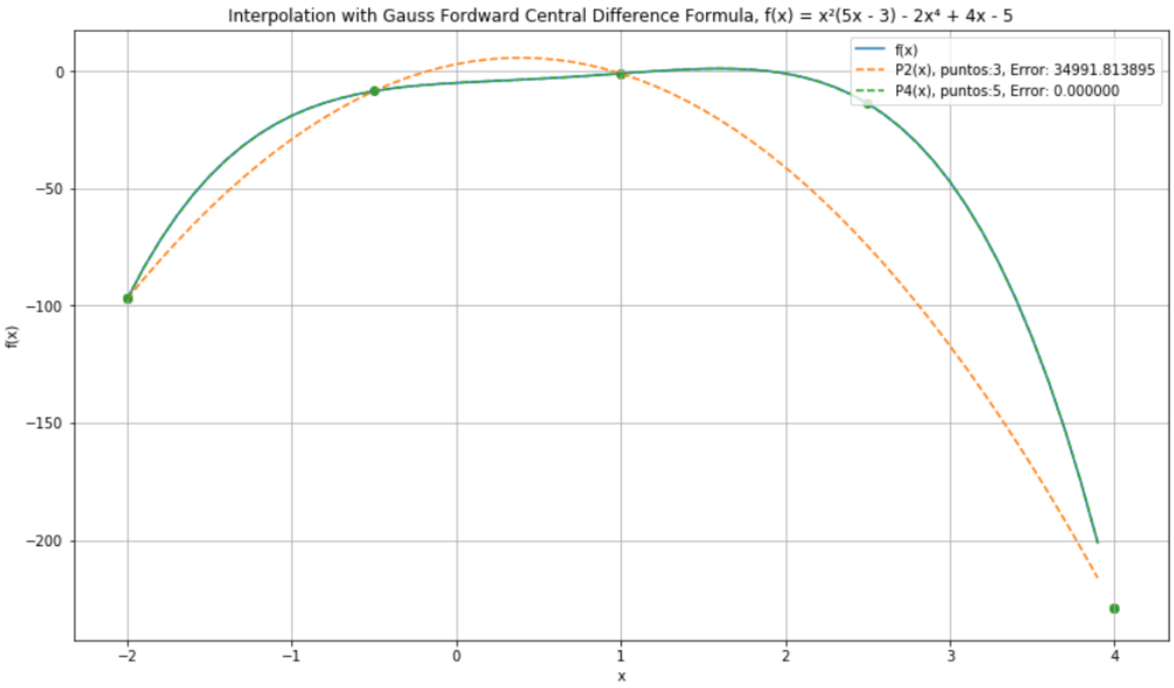
Resultados

Se presentan los resultados obtenidos para diferentes aproximaciones a las siguientes funciones:

- Función $f(x) = sin(x)$,
- Rango de evaluación: $(-5, 5)$
- Número de puntos para interpolación: 3, 5, 7 y 9



- Función: $f(x) = x^2(5X - 3) - 2X^4 + 4X - 5$
- Rango de evaluación: $(-2, 4)$
- Número de puntos para interpolación: 3, y 5



Para el cálculo del error se utilizó el Error Cuadrático Medio con puntos en el intervalo en incrementos de 0.1. Los puntos son equidistantes en el intervalo de cada gráfica.

Diferencias centradas hacia atrás

La interpolación de Gauss con diferencias centradas hacia atrás a través de los puntos anteriormente descritos está dado por:

$$P_n(x) = f(x_0) + \delta f(x_{-\frac{1}{2}})(s) + \frac{\delta^2 f(x_0)}{2!}(s)(s+1) + \frac{\delta^3 f(x_{-\frac{1}{2}})}{3!}(s)(s+1)(s-1) + \frac{\delta^4 f(x_0)}{4!}(s)(s+1)(s-1)(s+2) + \frac{\delta^5 f(x_{-\frac{1}{2}})}{5!}(s)(s+1)(s-1)(s+2)(s-2) + \dots \tag{11}$$

Algoritmo

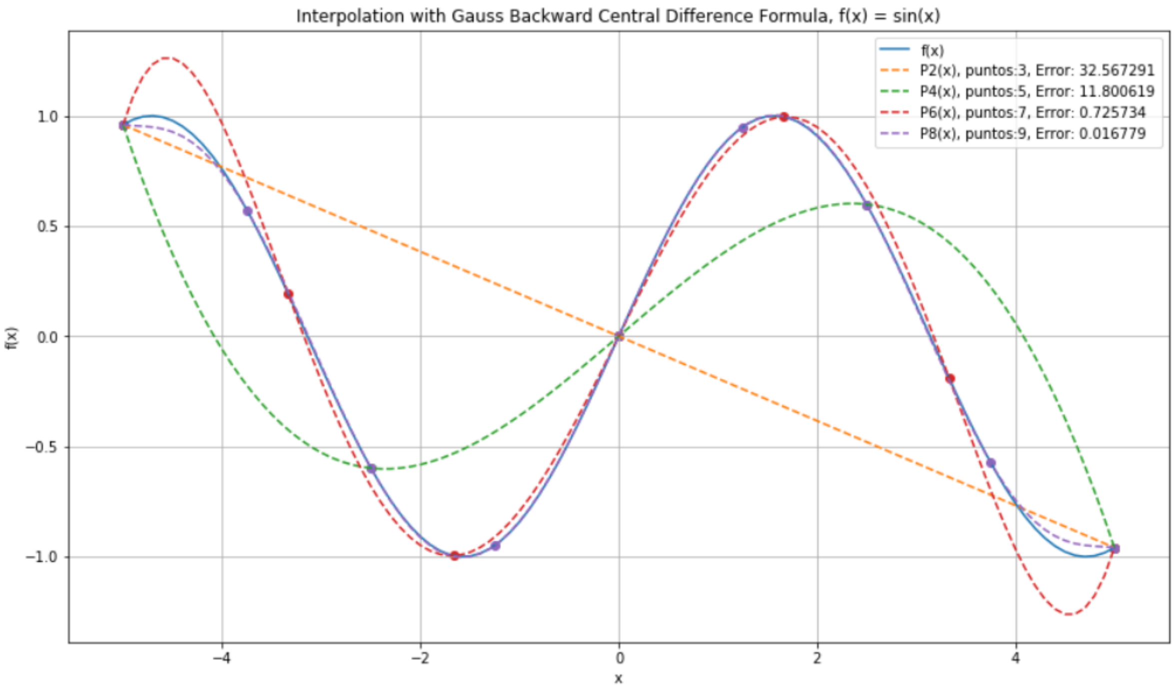
```
n <- Grado del polinomio
x[n+1] <- vector con las coordenadas en x de los n+1 puntos
y[n+1] <- vector con las coordenadas en y de los n+1 puntos

def get_backward_central_differences(x, y, n):
    '''Retorna los coeficientes de las diferencias
        usados para la evaluación del polinomio'''
    diff_table = [list(y), list(y)]
    size = n
    diffs = [y[int((size-1) / 2)]]
    row = 0
    for i in range(n-1):
        index = 0
        for j in range(size - 1):
            diff_table[(row+1)%2][index] = (diff_table[row][j+1] - diff_table[row][j])
            index += 1
        size -= 1
        diffs.append(diff_table[(row+1)%2][int((size-1) / 2)])
        row = (row + 1) % 2
    return diffs
```

```python

def gauss_bw_central_diff_evaluation(x, coefs, size, s):
    '''Retorna la evaluación del polinomio en un punto dado'''
    fact = 1.0
    result = coefs[0]
    s_prod = 1.0
    less = 0.0
    for i in range(size-1):
        if  i & 1:
            less += 1
            s_prod *= s + less
        else:
            s_prod *= s - less
        result += s_prod * coefs[i+1] / fact
        fact *= i+2
    return result
```

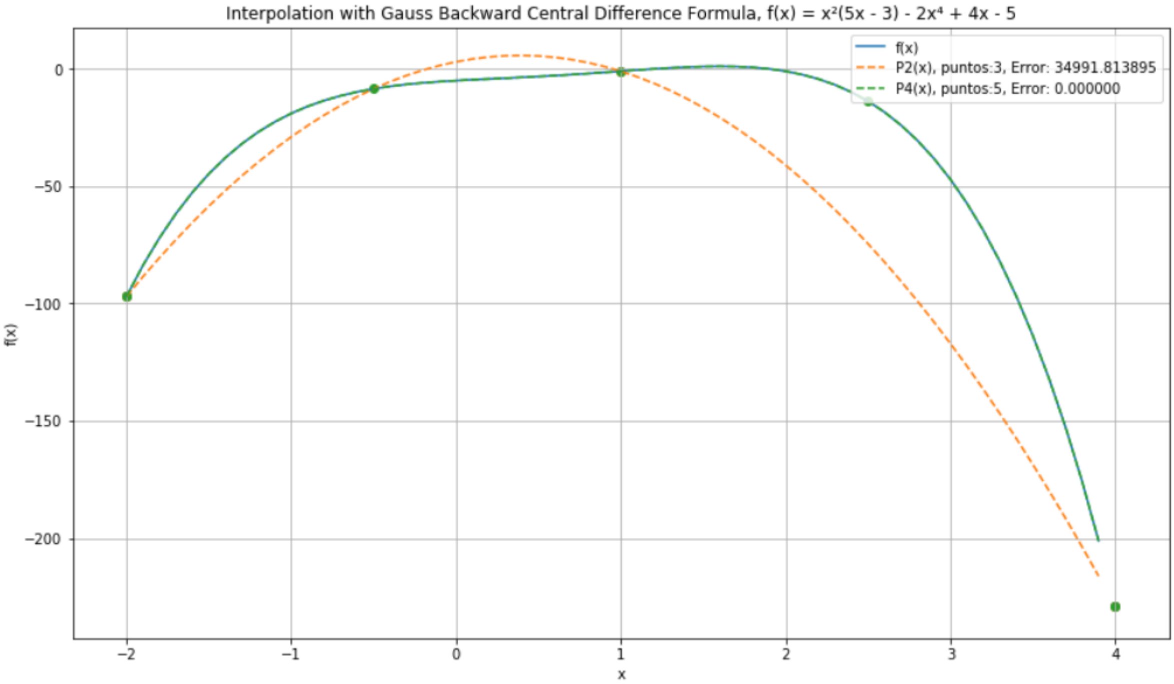
Resultados

Se presentan los resultados obtenidos para diferentes aproximaciones a las siguientes funciones:

- Función $f(x) = sin(x)$,
- Rango de evaluación: $(-5, 5)$
- Número de puntos para interpolación: 3, 5, 7 y 9



- Función: $f(x) = x^2(5X - 3) - 2X^4 + 4X - 5$
- Rango de evaluación: $(-2, 4)$
- Número de puntos para interpolación: 3, y 5



Para el cálculo del error se utilizó el Error Cuadrático Medio con puntos en el intervalo en incrementos de 0.1. Los puntos son equidistantes en el intervalo de cada gráfica.

Interpolación de Stirling

La interpolación de Stirling con diferencias centradas a través de $n+1$ puntos en un intervalo $[a,b]$ está dada por la función $P_n(x)$ con centros en $x_0, x_1, \ldots, x_{n-1}$ equidistantes con distancia h es:

$$\begin{aligned} P_n(x) = f(x_0) + (s)\left[\frac{\delta f(x_{1/2}) + \delta f(x_{-1/2})}{2}\right] + \frac{(s^2)}{2!}\delta^2 f(x_0) \\ + \frac{(s)(s^2-1)}{3!}\left[\frac{\delta^3 f(x_{1/2}) + \delta^3 f(x_{-1/2})}{2}\right] + \frac{(s^2)(s^2-1)}{4!}\delta^4 f(x_0) \\ + \frac{(s)(s^2-1)(s^2-2^2)}{5!}\left[\frac{\delta^5 f(x_{1/2}) + \delta^5 f(x_{-1/2})}{2}\right] + \ldots \end{aligned} \tag{12}$$

dónde el operador de diferencias centradas es el mismo definido anteriormente.

Algoritmo

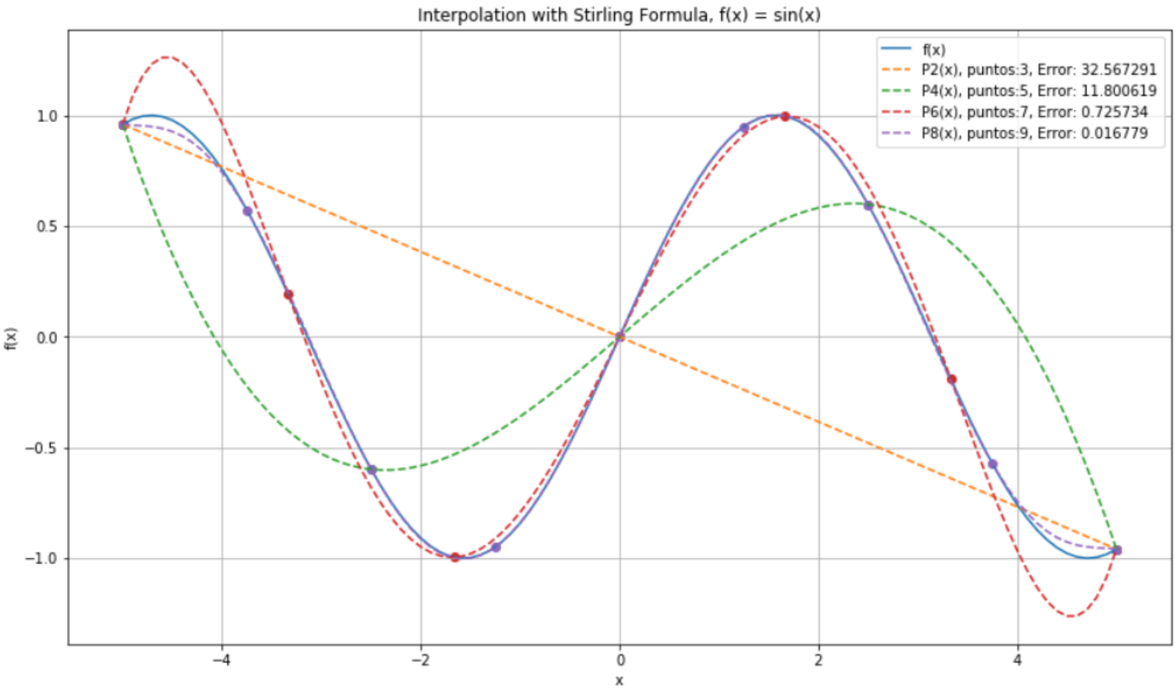
```
n <- Grado del polinomio
x[n+1] <- vector con las coordenadas en x de los n+1 puntos
y[n+1] <- vector con las coordenadas en y de los n+1 puntos

def get_central_differences(x, y, n):
    '''Retorna los coeficientes de las diferencias
       usados para la evaluación del polinomio'''
    fw_diffs = get_forward_central_differences(x, y, n)
    bw_diffs = get_backward_central_differences(x, y, n)
    return fw_diffs, bw_diffs

def stirling_evaluation(x, fw_coefs, bw_coefs, size, s):
    '''Retorna la evaluación del polinomio en un punto dado'''
    fact = 1.0
    result = fw_coefs[0]
    s_prod = 1.0
    less = 0.0
    for i in range(size-1):
```

```
        if i & 1:
            result +=  s * s * s_prod * bw_coefs[i+1] / fact
        else:
            result += s * s_prod * (bw_coefs[i+1] + fw_coefs[i+1]) / (2 * fact)
        fact *= i+2
        if  i & 1:
            less += 1
            s_prod *= s*s - less * less
    return result
```

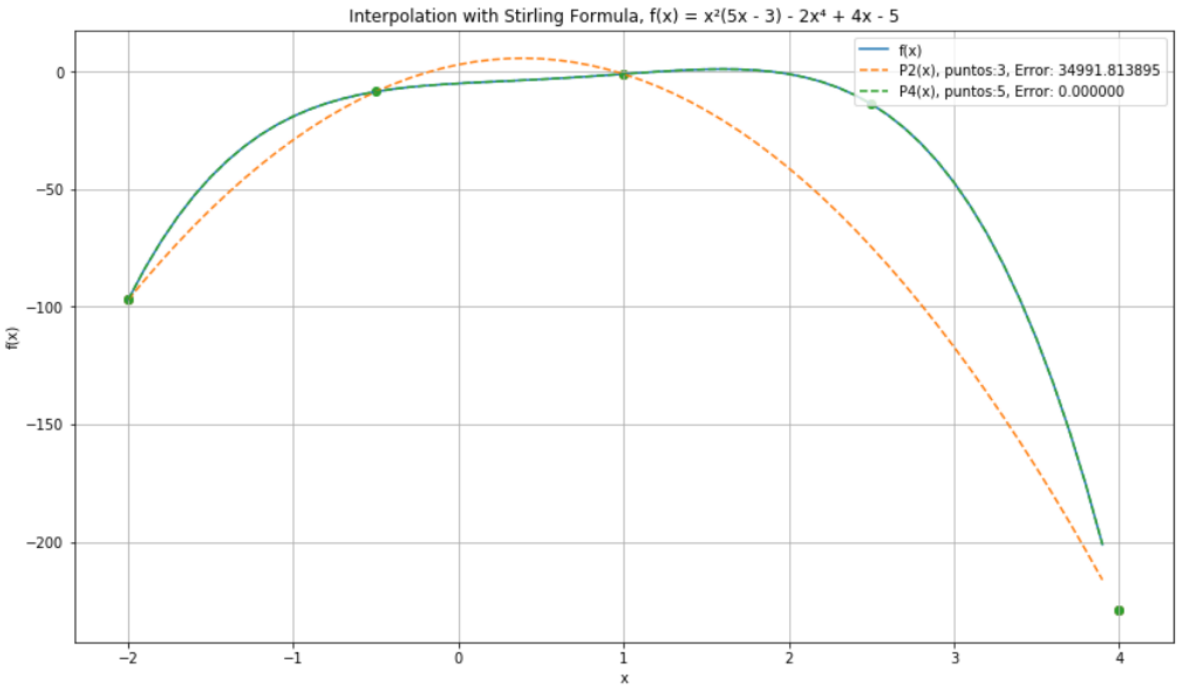
Resultados

Se presentan los resultados obtenidos para diferentes aproximaciones a las siguientes funciones:

- Función $f(x) = sin(x)$,
- Rango de evaluación: $(-5, 5)$
- Número de puntos para interpolación: 3, 5, 7 y 9



- Función: $f(x) = x^2(5X - 3) - 2X^4 + 4X - 5$
- Rango de evaluación: $(-2, 4)$
- Número de puntos para interpolación: 3, y 5



Para el cálculo del error se utilizó el Error Cuadrático Medio con puntos en el intervalo en incrementos de 0.1. Los puntos son equidistantes en el intervalo de cada gráfica.

Tiempos de ejecución

La siguiente tabla muestra los tiempos de ejecución para los algoritmos antes descritos para los procesos de interpolación de las 2 funciones anteriores y la generación de una muestra en el intervalo dado.

Algoritmo	**Tiempo Promedio**	**Repeticiones**	**Tiempo Promedio total**
Gregory-Newton Foreward	0.002000s	10000	31.204000s
Gregory-Newton Backward	0.002000s	10000	31.257000s
Gauss Foreward	0.002000s	10000	24.752000s
Gauss BackWard	0.001000s	10000	22.208000s
Stirling	0.014000s	10000	30.044000s

Conclusiones

De la tabla de tiempos anterior podemos observar que el algoritmo de Gauss es un poco más rápido que los demás, sin embargo, la implementación del algoritmo de Stirling usa dos veces la obtención de los coeficientes de las diferencias centradas hacia adelante y hacia atrás algo que podría mejorarse al obtenerse en la misma iteración haciendo que su tiempo se reduzca.

En cuanto al error de aproximación de cada método al observarlos desde los gráficos tienden al ser el mismo, es de suponerse puesto que los operadores usados para expresar cada polinomio son similares.

Durante la implementación del algoritmo de Gauss se observó un comportamiento extraño para cuando se iba a interpolar un número par de puntos haciendo que la función interpolada se mostrara desplazada a los puntos de interpolación.