

Problemas Análisis numérico

Álvarez Sánchez, Emanuel
alvarez_emanuel@javeriana.edu.co

Afanador Ochoa, Juan José Burgos Melo, Diego Andrés
j.afanador@javeriana.edu.co *burgosmd@javeriana.edu.co*

Barrera Martinez, Nicolai
barreram.n@javeriana.edu.co

February 2020

1 Evaluación de un polinomio

Evaluar polinomios o funciones en general tiene muchos problemas, a un para el software profesional. Comose requiere poder evaluar el polinomio en las raíces encontradas, es necesario dedicarle un momento a los detalles. Sea $p(x) = a_0x^n + a_1x^{n-1} + \dots + a_n$ un polinomio entonces, uno de los problemas que se enfrentanes evaluar el polinomio en valor dado x_0 de la manera mas eficiente. Un metodo visto en Horner

$$\begin{aligned} b_0 &= a_0 \\ b_k &= a_k + b_{k-1}x_0 (k = 1, \dots, n-1) \\ b_n &= P(x_0) \end{aligned}$$

Figure 1: Metodo Horner

1.1 Opción 1: Implementación del método de Horner para evaluar $f'(x)$

- Iteraciones del algoritmo de Horner

Sea

$$P(x) = a_0 x^N + a_1 x^{N-1} + \dots + a_{N-1} x + a_N .$$

Si

$$d_0 = a_0 \quad \text{y} \quad d_k = a_k + d_{k-1} x_0 ,$$

para $k = 1, 2, \dots, N-1, N$, entonces

$$d_N = P(x_0) .$$

Figure 2: Iteraciones Horner

Además, si

$$Q(x) = d_0 x^{N-1} + d_1 x^{N-2} + \dots + d_{N-2} x + d_{N-1} ,$$

entonces

$$P(x) = (x - x_0) Q(x) + d_N .$$

Figure 3: Iteraciones Horner 2

- Ventaja del método

$$P(x) = (x - x_0) Q(x) + d_N ,$$

donde

$$Q(x) = d_0 x^{N-1} + d_1 x^{N-2} + \dots + d_{N-2} x + d_{N-1} ,$$

diferenciando con respecto a x da

$$P'(x) = Q(x) + (x - x_0) Q'(x)$$

y

$$P'(x_0) = Q(x_0) .$$

Figure 4: Iteraciones Horner 2

- Implementacion en Python

```
#retorna la lista de dk
def honer(lista,valorX):
    coeficientes=lista
    #d=[0,0,0,0,0,0]
    d=[0,0,0,0,0]
    #d=prueba(lista)

    for i in range(0,len(coeficientes)):

        d[i]=d[i-1]*valorX+coeficientes[i]
    return d
```

Figure 5: Implementacion Python

Por otra parte se implementa la siguiente función que permite completar el método, teniendo como parámetros una lista que corresponde a los dk y el valorX en el que se quiere evaluar el polinomio Finalmente se implementa la

```
#retorna el valor de P(x0) segun los coeficientes
def honer1(lista,valorX):
    coeficientes=lista
    resultado=0

    for i in range(0,len(coeficientes)-1):
        resultado=resultado*valorX+coeficientes[i]
    return resultado
```

Figure 6: Implementacion Python 2

funcion derivada

```
#Derivada P'(x0)=Q(x0)
def derivada(lista, valorX):
    return honer1(honer(lista,valorX),valorX)
```

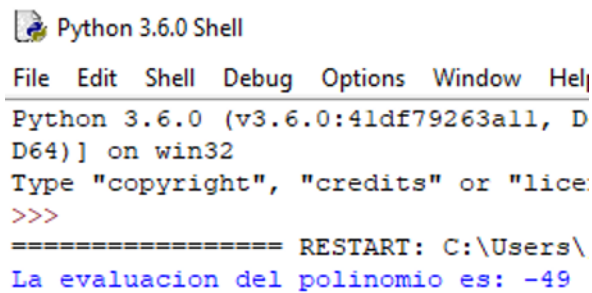
Figure 7: Implementacion Python 3

Esta función recibe como parámetros la lista de coeficientes del polinomio a evaluar ordenados de izquierda a derecha según el grado de dicho polinomio. A continuación se realiza un ejemplo:

Polinomio a evaluar $p(x) = 2x^4 - 3x^2 + 3x - 4$ en $x_0 = -2$

```
print("La evaluacion del polinomio es: "+str(derivada([2,0,-3,3,-4],-2)))
```

Figure 8: Ejecucion Python



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
Python 3.6.0 (v3.6.0:41df79263a11, D
D64) on win32
Type "copyright", "credits" or "lice
>>>
===== RESTART: C:\Users\
La evaluacion del polinomio es: -49
```

Figure 9: ejecucion Python 2

1.2 Opción 2: Implementacion del metodo de Horner para numero complejos

Funciones implementadas en R Ejemplo: Resultado:

```
#retorna el valor de P(x0) segun los coeficientes
def horner1(lista,valorX):
    coeficientes=lista
    resultado=0

    for i in range(0,len(coeficientes)-1):
        resultado=resultado*valorX+coeficientes[i]
    return resultado
```

Figure 10: complejos Python

```
#Ejemplo para el polinomio (1+i)x^3+2=f(x), con x0=1-i

coef=[1+cmath.sqrt(-1),0,0,2]
valr=1-cmath.sqrt(-1)
```

Figure 11: ejemplo complejos Python

```
print("La evaluacion del polinomio es: "+str(honer1(coef,valr)))
```

Figure 12: ejemplo complejos Python 2

```
La evaluacion del polinomio es: (2-2j)
```

Figure 13: resultado complejos Python

2 Optima aproximacion polinomica

La implementaci on de punto flotante de una funcion en un intervalo a menudo se reduce a una aproximacion polinomica, siendo el polinomio tipicamente proporcionado por el algoritmo Remez. Sin embargo, la evaluacion de coma flotante de un polinomio de Remez a veces conduce a cancelaciones catastroficas. El algoritmo Remez es una metodolog ia para localizar la aproximacion racional minimax a una funcion. Las cancelaciones que tambien hay que considerar en el caso de del metodo de Horner, suceden cuando algunos de los coeficientes polinomicos son de magnitud muy pequena con respecto a otros. En este caso, es mejor forzar estos coeficientes a cero, lo que tambien reduce el recuento de operaciones. Esta tecnica, usada clasicamente para funciones pares o impares, puede generalizarse a una clase de funciones mucho mas grandes. Aplicar esta tecnica para $f(x) = \sin(x)$ en el intervalo $[-\pi/64, \pi/64]$ con una precision deseada doble. Para cada caso, evalúe la aproximacion polinomica de la funcion, el error relativo y el numero de operaciones necesarias

2.1 Opción 1: Aplicacion de una aproximacion de Taylor

```
x=sy.Symbol('x')
f=sin(x)
```

Figure 14: Taylor

```
# aproximacion de taylor
def taylor(funcion,x0,n):
    i=0
    p=0
    inf=math.pi/64
    sup=math.pi/64
    while i<=n:
        p=p+(funcion.diff(x,i).subs(x,x0))/(math.factorial(i))*(x-x0)**i
        i+=1
    return p
```

Figure 15: Taylor 2

Se puede apreciar que la instruccion `funcion.diff(x,i)` permite encontrar la i -esima derivada de la funcion que se inserta por parametro. Finalmente, esta funcion retorna una cadena en la cual se observa la descomposicion de la funcion $\sin(x)$ en el intervalo estipulado, como se muestra a continuacion: Se puede apreciar que la instruccion `funcion.diff(x,i)` permite encontrar la i -esima derivada de la funcion que se inserta por parametro. Finalmente, esta funcion retorna una cadena en la cual se observa la descomposicion de la funcion $\sin(x)$ en el intervalo estipulado, como se muestra a continuacion: Esta es la aproximacion de la funcion seno a un polinomio de grado 5, para otro grado se modifica el valor de n en el parametro de la funcion

```
(x - 5)**4*sin(5)/24 - (x - 5)**3*cos(5)/6 - (x - 5)**2*sin(5)/2 + (x - 5)*cos(5) + sin(5)
```

Figure 16: Taylor 3

2.2 Opción 2: Implementacion metodo de Remez

- Inicializacion del metodo

```
HUSL_BLUE = (0.23299120924703914, 0.639586552066035, 0.9260706093977744)
Sin_HI = float.fromhex('0x1.62e42fee00000p-1')
Sin_LOW = float.fromhex('0x1.a39ef35793c76p-33')
L1 = float.fromhex('0x1.5555555555593p-1')
L2 = float.fromhex('0x1.999999997fa04p-2')
L3 = float.fromhex('0x1.2492494229359p-2')
L4 = float.fromhex('0x1.c71c51d8e78afp-3')
L5 = float.fromhex('0x1.7466496cb03dep-3')
L6 = float.fromhex('0x1.39a09d078c69fp-3')
L7 = float.fromhex('0x1.2f112df3e5244p-3')
SQRT2_HALF = float.fromhex('0x1.6a09e667f3bcdp-1')
CTX = mpmath.MPContext()
CTX.prec = 200
```

Figure 17: Remez

- Implementacion algortimo de Remez

```
def sin_mpf(x):
    return CTX.sin(CTX.mpf(x))

def sin_ieee754(x):
    fl, ki = np.frexp(x)
    if fl < SQRT2_HALF:
        fl *= 2
        ki -= 1
    f = fl - 1
    k = float(ki)

    s = f / (2 + f)
    s2 = s * s
    s4 = s2 * s2
    # Terms with odd powers of s^2.
    t1 = s2 * (L1 + s4 * (L3 + s4 * (L5 + s4 * L7)))
    # Terms with even powers of s^2.
    t2 = s4 * (L2 + s4 * (L4 + s4 * L6))
    R = t1 + t2
    hfsq = 0.5 * f * f
    return k * Sin_HI - ((hfsq - (s * (hfsq + R) + k * Sin_LOW)) - f)
```

Figure 18: Remez 2

- Algoritmo para visualizar errores y grabarlos en una grafica

```
def main():
    num_points = 2**14
    x_vals = np.linspace(0, np.exp(2.5), num_points)
    x_vals = x_vals[1:]
    log_rel_errors = []
    for x_val in x_vals:
        sin_val = sin_ieee754(x_val)
        sin_hp_val = sin_mpf(x_val)
        log_rel_errors.append(abs(sin_val - sin_hp_val) / abs(sin_hp_val))

    plt.plot(x_vals, log_rel_errors, color=HUSL_BLUE)
    filename = 'log_high_precision_relative_error.png'
    plt.savefig(filename, bbox_inches='tight')
    print('Saved ' + filename)

if __name__ == '__main__':
    main()
```

Figure 19: Remez 3

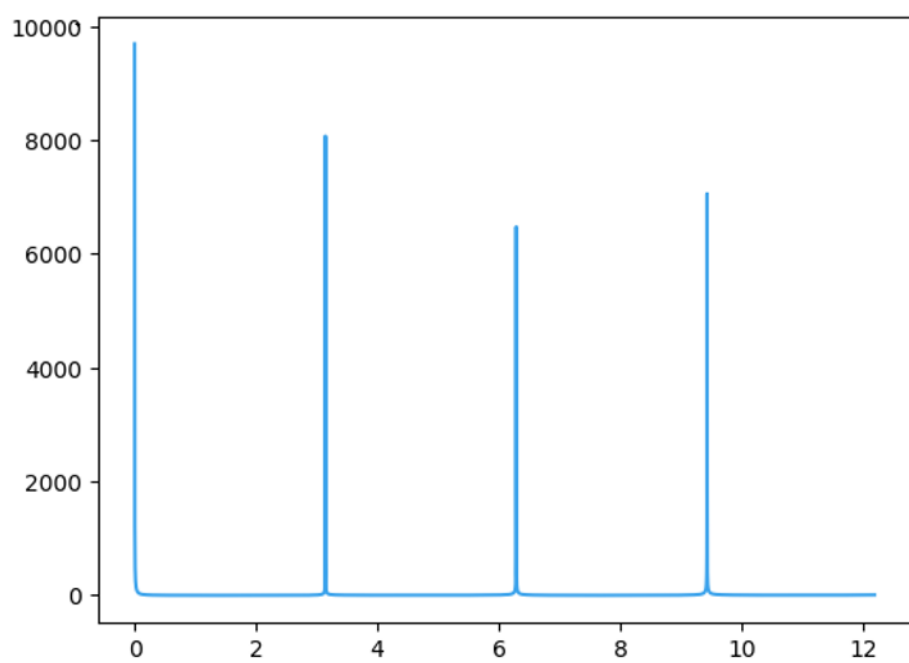


Figure 20: Grafica Remez