Ejercicio del método de Horner

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

Completa la siguiente tabla, usando el método de Horner

$$P(x) = 2 + 4x - 5x^2 + 2x^3 - 6x^4 + 8x^5 + 10x^6$$

Evalúa el polinomio $P(x)$ en:

x	P(x)
-1.5	
-0.65	
0.1	
1.4	
2.87	

Estrategia

1. Una primera solución es proporcionar al algoritmo el valor del punto a evaluar, esto funciona si son pocos puntos de la tabla.
2. La otra opción es manejar todos los datos que se van a evaluar dentro de una tabla, trabajemos de ésta manera.

Estrategia

1. Una primera solución es proporcionar al algoritmo el valor del punto a evaluar, esto funciona si son pocos puntos de la tabla.
2. La otra opción es manejar todos los datos que se van a evaluar dentro de una tabla, trabajemos de ésta manera.

Primera estrategia

```
1  n=6
2  coeficiente = [2,4,-5,2,-6,8,10]
3
4  z = eval(raw_input('Escribe el valor de x a
       evaluar en el polinomio: '))
5
6  b = coeficiente[6]
7
8  while n >0 :
9       n = n - 1
10      b = coeficiente[n] + z * b
11
12 print 'El valor del polinomio evaluado en ', z
       , ' es = ', b
```

Segunda estrategia

```
1 x0=[-1.5,  -0.65,  0.1,  1.4,  2.87]
2
3 a=[2,4,-5,2,-6,8,10]
```

Al manejar las listas, debemos de revisar con cuidado la manera en que se van a usar los elementos de ellas, ya que para el arreglo $x0$ que contiene los puntos a evaluar, debe de hacerse de izquierda a derecha, mientras que en el método de Horner, los coeficientes se usan de derecha a izquierda.

Tomamos entones primero un punto de $x0$ y se aplica el método de Horner, por lo que se usarán dos ciclos anidados.

```
1 for i in range(len(x0)):
2     Poli=0
3     for n in range(len(a)-1,-1,-1):
4         Poli=a[n] + Poli * x0[i]
5     print "Poli(%.2f) = %f" %(x0[i],Poli)
```

Salida del código

$$
\begin{aligned}
P(-1.50) &= 0.781250 \\
P(-0.65) &= -4.506831 \\
P(0.10) &= 2.351490 \\
P(1.40) &= 98.559680 \\
P(2.87) &= 6758.702451
\end{aligned}
$$

Mejoras adicionales al código

Como hemos resuelto el problema que se nos pedía

x	P(x)
-1.5	0.781250
-0.65	-4.506831
0.1	2.351490
1.4	98.559680
2.87	6758.702451

Podemos hacer unas mejoras aprovechando lo que ya vimos en el Tema 0, con funciones que nos agrupen el trabajo de las operaciones:

```
1 def metodoHorner(x):
2     metHorner = 0
3     for n in range(len(a)-1,-1,-1):
4         metHorner = a[n] + metHorner * x
5     return metHorner
```

Y como queremos comparar los valores obtenidos contra el polinomio, necesitamos otra función que evalúe el mismo, en el intervalo de puntos, para ello usamos:

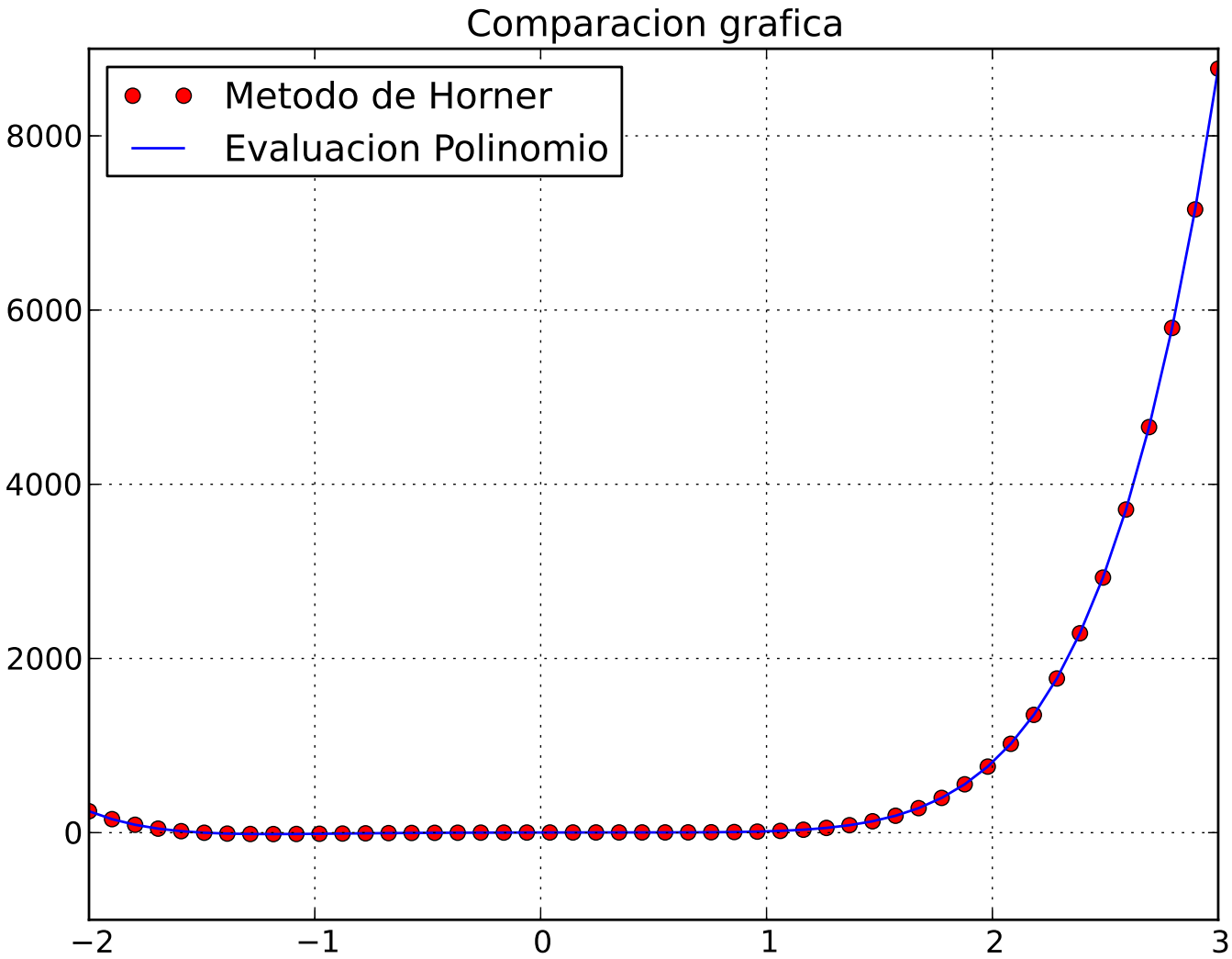
```
1 def evaluaPoli(x):
2     return 2 + x *(4 + x * (-5 + x * (2 + x
          *(-6 + x * (8 + x * 10)))))
```

Para graficar los puntos obtenidos por el método de Horner y por la evaluación directa del polinomio, vamos a ocupar el módulo de `numpy` y la librería de `mattplotlib`:

```
import matplotlib.pyplot as plt
from numpy import *
```

El código completo queda:

```
1  import matplotlib.pyplot as plt
2  from numpy import *
3
4  def metodoHorner(x):
5  ...
6
7  def evaluaPoli(x):
8  ...
9
10 x = linspace(-2.,3.)
11 plt.plot(x,metodoHorner(x),'ro')
12 plt.plot(x,evaluaPoli(x))
13 plt.grid(True)
14 plt.show()
```


Comparacion grafica
Metodo de Horner
Evaluacion Polinomio
8000
6000
4000
2000
0
−2
−1
0
1
2
3