Física Numérica

Tarea #1

D. A. Vázquez Gutiérrez

Escuela Superior de Física y Matemáticas, Instituto Politécnico Nacional,Unidad Profesional .Adolfo López Mateos", Zacatenco, Edificio 9, Col. San Pedro Zacatenco, C.P. 07730 del. Gustavo A. Madero, Ciudad de México, México

email: dvazquezg1600@alumno.ipn.mx

12 de marzo de 2024

Resumen

1. Limites de Underflow y Overflow

1.1. Overflow

El overflow ocurre cuando el resultado de una operación aritmética es un numero que excede el rango de representación permitido para el tipo de datos especifico, por lo tanto lo que entenderemos como *Limite de overflow* sera el numero del máximo valor representable.

Una idea inicial para encontrarlo es la auscultación directa del valor, esto por medio de la función *for*:

```python

"""
@author: D.A. Vazquez Gutierrez
"""
N=20000
overflow=1
for n in range(1, N): #queremos que inicie en 1 la cuenta .
    overflow*=2
    print("|%2d"%n,"\t\t","|%2.5e"%overflow)
```

Código 1: Auscultación del Overflow en Python.

Remarcamos el uso de `"%2d" % n` , con lo que especificamos el ancho en el que se imprimirá el numero , d indicando que unicamente queremos imprimir números enteros; Por otro lado , al ser los números para llegar a Overflow muy grandes, vale mucho la pena el utilizar notación científica, por lo que utilizamos `"%2.5e" % overflow`, donde con e especificamos el uso de notación científica.

Esto nos arroja como resultado el contenido de la Figura 1 , donde nuestro programa presenta el error `OverflowError: int too large to convert to float` después de la iteración 1023

Dado que queremos un programa que no arroje errores y que a la vez nos diga el numero de posibles iteraciones en factor de dos para obtener un limite de overflow , asi como el valor del limite del overflow , pensamos en un codigo diferente :

```python
"""

@author: D.A. Vazquez Gutierrez
"""

```

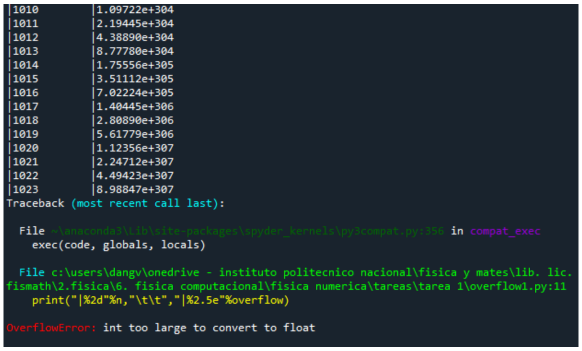

Figura 1: Compilación del Código 1. Nos arroja un error de Overflow en la iteracion 1023

```
N=2000 #para un N lo suficientemente
    grande

while True:
    try:
        overflow=1    #valor inicial del
    overflow
        s=0          #Contamos los pasos
        for n in range(N):

              overflow*=2
              s+=1


        print("El valor del limite de
    overflow es:","%e"%(overflow), "con la
     iteracion","%2d"%s,) #Si esto arroja
    un error de Overflow, lo manda al "
    except"
        break

    except OverflowError:   #Si se llega a
     un overflow significa que el numero N
     que elegimos es demaciado grande
        N=N-1                   #Disminuimos
    N
```

Código 2: Overflow en Python sin OverflowError.

En este código , a diferencia del primero , tenemos el dato exacto de la iteración y del valor del limite de overflow , siendo estos datos los únicos arrojados por el primero.

Para esto se utilizaron las funciones `try` y `except` para manejar los casos en que se tenia error , y a través de modificar el numero de iteraciones del `for`, encontrar el valor igual o equivalente al encontrado en el primer programa.

Como podemos ver en la Figura 2. Obtenemos los mismos resultados que en el primer programa , con 1023 iteraciones, para llegar a un limite de overflow:

$$l_{ofw} = 8{,}988466 \times 10^{307} \tag{1}$$

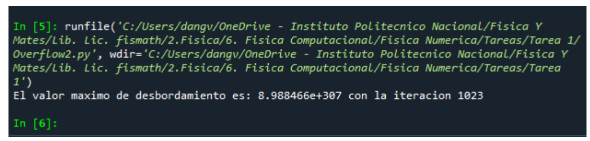

Figura 2: Compilación del Código 2. Unicamente tenemos la iteración y el valor del limite de overflow

Un error que tenemos en este código es la elección ”trivial”de la variable N , siendo esta el numero máximo inicial de iteraciones para encontrar el overflow.

Por ejemplo , si se eligiera un N , no lo suficientemente grande , el resultado obtenido en el programa seria totalmente erróneo , lo ideal entonces seria el comenzar desde un numero pequeño , donde no se tenga un overflow , e ir escalando hasta topar con el overflow; seria una solución de abajo hacia arriba y no de arriba hacia abajo como aquí se hizo.

1.2. Underflow

El Underflow ocurre cuando el resultado de una operacion aritmética es un numero que es mas pequeño que el valor mínimo representable para un tipo de dato , en nuestro caso , para un dato `float` . Para lograr esto sin la necesidad de tener que imprimir en pantalla todas las iteraciones, usamos la función `while` :

```

"""

@author: D.A. Vazquez Gutierrez

"""

underflow=1
llimite=1
```

```python
s=0
while underflow!=0:  #nuestra condicion de parada sera cuando el valor sea indistinguible del cero.
    s+=1
    limite=underflow  #guardamos el valor previo a esta iteracion,por si llegara a haber underflow .
    underflow/=2

print("El valor de limite  de underflow es:","%e"%(limite), "con la iteracion","%2d"%(s-1),)
```

Código 3: Underflow en Python .

Compilando este código obtenemos que el valor del underflow dentro del factor de 2 , en la iteración 1074 , como se ve en la figura 3, así como un limite de underflow de:

$$l_{ufw} = 4{,}940656 \times 10^{-324} \tag{2}$$

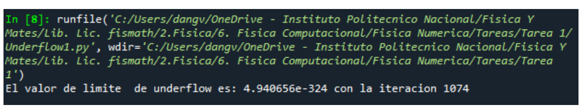

Figura 3: Compilación del Código 3. Underflow

Con este programa no hubo mayor problema ni hubo necesidad de utilizar funciones no vistas en clase.

Bajo las condiciones del propio código 3, sabemos que el valor marcado como limite de underflow al ser dividido por un factor de 2 en alguna potencia , este sera indistinguible de 0 para la maquina.

2. Precisión Maquina.

En clase, definimos la precisión maquina como el máximo número positivo que puede sumarse al numero almacenado sin cambiar el valor almacenado:

$$1_c + \epsilon_m = 1_c \tag{3}$$

Donde ϵ_m es el valor de la precisión maquina. Así como esta escrito en la ecuación es fácil ver que podemos determinarlo utilizando una función `while` , condicionándola con (3).

```python

# Determina aproximadamente la precision de maquina

paso=1.0
pasomas=0   #tiene el valor de 1 mas paso dividido en factores de 2.


while 1!=pasomas:  #Mientras estos sean diferentes, el ciclo se repitira indefinidamente.
    paso=paso/2      #reducimos en factor de 2 a paso
    pasomas=1+paso   #asignamos un valor a paso mas en funcion de paso

print('Presicion de maquina:',"%e"%paso,'valor del cambio',pasomas)

print('Podemos ver que la maquina no puede diferenciar entre el 1 y ',pasomas,' el cual matematicamente es diferente de 1 , pero numericamente no.')
```

Código 4: Precisión Maquina.

De esta forma , obtenemos que , por la figura 4, el valor de la presionan maquina es :

$$\epsilon_m = 1{,}11023 \times 10^{-16} \tag{4}$$

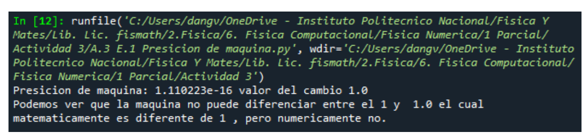

Figura 4: Compilación del Código 4. Precisión Maquina

3. Función Seno

Recordemos que la función seno podemos definiría empleando una serie infinita, la cual es :

$$
\begin{aligned}
senx &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ... \\
&= \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!}
\end{aligned} \tag{5}
$$

Para facilitar nuestro programa con una *suma inteligente* , notemos la siguiente relación entre los términos *n* y *n-1*

Por una parte :

$$a_{n-1}(x) = \frac{(-1)^{n-2} x^{2n-3}}{(2n-3)!} \tag{6}$$

$$a_n(x) = \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} \tag{7}$$

utilizando (6) y (7) en un cociente

$$\frac{a_n(x)}{a_{n-1}(x)} = \frac{-(x^2)}{(2n-2)(2n-1)}$$

Por lo tanto , tenemos la siguiente relación :

$$a_n(x) = \frac{-(x^2)}{(2n-2)(2n-1)} a_{n-1}(x) \tag{8}$$

Con esto tenemos una relación numérica con la cual hacer una función recursiva, sin embargo, como sabemos, en la formulación ideal de la serie , esta es infinita; por lo tanto nunca terminaríamos de calcular el valor "puro"de algún valor evaluado en la función seno .

Entonces, no nos interesa tanto que sea el valor preciso e ideal de este, sino que sea lo suficientemente cercano, donde nos entrometemos con el concepto de *error absoluto*, que es la diferencia entre el valor que consideramos mas exacto , y el valor que obtendremos .

Podemos entonces considerar que un criterio para tener un error absoluto menor a una parte en 10^8 es que:

$$\left|\frac{a_N(x)}{suma(N-1)}\right| \leq 10^{-8} \tag{9}$$

Por lo tanto , usaremos este criterio como argumento para detener la función recursiva `while`

```python

"""
@author: D.A. Vazquez Gutierrez
"""

import math

pi=math.pi
#Primero definimos nuestra nueva funcion seno solamente para abs(x)=<2pi
def senopirata(x):
    a=x
    suma=a
    sumamenos=a
    amenos=0
    n=1
    while abs(a/sumamenos)>(10**(-8)): # valor del error absoluto
        amenos=a
        sumamenos=suma
        n=n+1
        a=(((-1)*x**2)*(amenos))/(((2*n)-2)*((2*n)-1))
        suma=suma+a
    return suma
g=.25
y=math.sin(g)
k=senopirata(g)
print("seno pirata",k, "seno original",y)

#En este punto ya sabemos que funciona el seno pirata, ahora  arreglemos el problema numerico para  x<2pi

def senopirataPRIME(x):
    r = x % (2 * math.pi) #uso de modulo para obtener la generalizacion de la funcion
    return senopirata(r)

t=45.63
u=math.sin(t)
l=senopirataPRIME(t)
print("seno pirata PRIME",l, "seno original",u)
```

Código 5: Primera version de funsion seno.

Creamos entonces una primera versión de la función , la primera parte unicamente funcionando entre 0 a 2 π , y observando su funcionamiento. Posteriormente utilizando el modulo entre algún numero x y 2π , generalizamos la función para cualquier numero x; tomando así en cuenta la identidad del seno tal que :

$$sen(x + 2n\pi) = sen(x)$$

Veamos que este código funciona correctamente en la figura 5, sin embargo no es lo suficientemente eficiente.

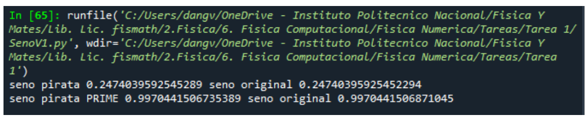

Figura 5: Compilación del Código 4. Precisión Maquina

Entonces, creamos una versión un poco mas sintetizada. Aprovechando esto ,veamos el funcionamiento de esta versión del seno para varios valores de x , así como también obtengamos el error relativo de este respecto a el valor que tendría en la biblioteca `math` .

```python

"""
@author: D.A. Vazquez Gutierrez
"""
import math
import pandas as pd

pi = math.pi

# Definimos nuestra nueva funcion seno
    para todo valor de x
def seno(x):
    r = x % (2 * pi)
    a = r
    suma = a
    sumamenos = 1
    amenos = 0
    n = 1
    while abs(a / sumamenos) > (10**(-8)):
        #tamano del error absoluto
        amenos = a
        sumamenos = suma
        n = n + 1
        a = (((-1) * r**2) * (amenos)) /
    (((2 * n) - 2) * ((2 * n) - 1))
        suma = suma + a
    return suma

#Creamos entonces una tabla para  poder
    encontrar el error relativo  entre
    senos .

h = 15     #largo del arreglo
particion = [0] * h
sumaPar = [0] * h
senoOr = [0] * h
error = [0] * h

for i in range(h):
    particion[i] = (2*pi*(1+i)) / h #Rango
     de la particion
    sumaPar[i] = seno(particion[i])
    senoOr[i] = math.sin(particion[i])
    error[i] = ((sumaPar[i] - senoOr[i]) /
     senoOr[i])

# Crear un DataFrame con los resultados
df_resultados = pd.DataFrame({
    'Particion': particion,
    'Seno Aproximado': sumaPar,
    'Seno Original': senoOr,
    'Error Relativo': error
})


df_resultados.to_excel('resultados_seno.
    xlsx', index=False) #Paso de listas de
     python a excel para despue pasarlas a
     tablas en LateX


print(df_resultados)
```

Código 6: Segunda version de funsion seno.

Primero $|x| \leq 2\pi$ con una precisión de 10^{-8}, despues $|x| \leq 2\pi$ con una precisión de 10^{-16}, $|x| \geq 2\pi$ con una precisión de 10^{-8} y por ultimo $|x| \geq 2\pi$ con una precisión de 10^{-16} . Estas tablas se encuentras al final de este documento debido al que ocupan mucho mas espacio.

Podemos contrastar las primeras dos con las ultimas dos tablas en que esta claro que la propiedad en la que el seno funciona para todo valor de x es valida, o al lo es cuando el valor de x no es lo suficientemente cercano a cero; ya que como podemos ver , entre el cuadro 1 y 2 , , contrastando el ultimo valor , que corresponde a la partición con valor de 2π , hace que sea seno igual cero.

Sin embargo al intentar ser muy precisa la maquina , esta tiene valores muy pequeños que ser contrastados , lo que hace que el error relativo fácilmente se dispare al estar cerca , tanto de cero como de 2π.Pero ademas de ese caso muy especifico , cuando se trata de medir el error relativo este si tiende a disminuir significativamente mientras mas cercanos mas pequeño sea el error absoluto.

Particion	Seno Aproximado	Seno Original	Error Relativo
0,418 879 02	0,406 736 643	0,406 736 643	$4{,}285\,72 \times 10^{-12}$
0,837 758 041	0,743 144 825	0,743 144 825	$-2{,}156\,43 \times 10^{-11}$
1,256 637 061	0,951 056 516	0,951 056 516	$2{,}459\,94 \times 10^{-11}$
1,675 516 082	0,994 521 895	0,994 521 895	$-1{,}812\,54 \times 10^{-11}$
2,094 395 102	0,866 025 404	0,866 025 404	$1{,}182\,97 \times 10^{-11}$
2,513 274 123	0,587 785 252	0,587 785 252	$-8{,}357\,86 \times 10^{-12}$
2,932 153 143	0,207 911 691	0,207 911 691	$1{,}020\,33 \times 10^{-11}$
3,351 032 164	−0,207 911 691	−0,207 911 691	$4{,}113\,57 \times 10^{-12}$
3,769 911 184	−0,587 785 252	−0,587 785 252	$2{,}752\,21 \times 10^{-11}$
4,188 790 205	−0,866 025 404	−0,866 025 404	$2{,}589\,89 \times 10^{-10}$
4,607 669 225	−0,994 521 895	−0,994 521 895	$-7{,}381\,82 \times 10^{-11}$
5,026 548 246	−0,951 056 516	−0,951 056 516	$2{,}514\,34 \times 10^{-11}$
5,445 427 266	−0,743 144 826	−0,743 144 825	$3{,}263\,14 \times 10^{-10}$
5,864 306 287	−0,406 736 643	−0,406 736 643	$-1{,}889\,76 \times 10^{-10}$
6,283 185 307	$2{,}1433 \times 10^{-15}$	$-1{,}133\,11 \times 10^{-15}$	−2,891 527 831

Cuadro 1: Resultados de la aproximación del seno y error relativo.$|x| \leq 2\pi$ con una presicion de 10^{-8}

Particion	Seno Aproximado	Seno Original	Error Relativo
0,418 879 02	0,406 736 643	0,406 736 643	0
0,837 758 041	0,743 144 825	0,743 144 825	$1{,}493\,95 \times 10^{-16}$
1,256 637 061	0,951 056 516	0,951 056 516	0
1,675 516 082	0,994 521 895	0,994 521 895	0
2,094 395 102	0,866 025 404	0,866 025 404	$2{,}563\,95 \times 10^{-16}$
2,513 274 123	0,587 785 252	0,587 785 252	$-1{,}888\,82 \times 10^{-16}$
2,932 153 143	0,207 911 691	0,207 911 691	$2{,}669\,94 \times 10^{-16}$
3,351 032 164	−0,207 911 691	−0,207 911 691	$1{,}735\,46 \times 10^{-15}$
3,769 911 184	−0,587 785 252	−0,587 785 252	$-3{,}777\,65 \times 10^{-16}$
4,188 790 205	−0,866 025 404	−0,866 025 404	$-7{,}691\,85 \times 10^{-16}$
4,607 669 225	−0,994 521 895	−0,994 521 895	$-1{,}897\,78 \times 10^{-15}$
5,026 548 246	−0,951 056 516	−0,951 056 516	$1{,}167\,36 \times 10^{-15}$
5,445 427 266	−0,743 144 825	−0,743 144 825	$1{,}942\,14 \times 10^{-15}$
5,864 306 287	−0,406 736 643	−0,406 736 643	$3{,}002\,55 \times 10^{-15}$
6,283 185 307	$2{,}1433 \times 10^{-15}$	$-1{,}133\,11 \times 10^{-15}$	−2,891 527 831

Cuadro 2: Resultados de la aproximación del seno y error relativo.$|x| \leq 2\pi$ con una presicion de 10^{-16}

Particion	Seno Aproximado	Seno Original	Error Relativo
8	0,989 358 247	0,989 358 247	$-2{,}755\,15 \times 10^{-11}$
16	−0,287 903 317	−0,287 903 317	$5{,}456\,19 \times 10^{-12}$
24	−0,905 578 362	−0,905 578 362	$5{,}3429 \times 10^{-11}$
32	0,551 426 681	0,551 426 681	$-2{,}659\,65 \times 10^{-13}$
40	0,745 113 161	0,745 113 16	$8{,}189\,16 \times 10^{-11}$
48	−0,768 254 661	−0,768 254 661	$1{,}031\,32 \times 10^{-10}$
56	−0,521 551 002	−0,521 551 002	$-7{,}375\,06 \times 10^{-11}$
64	0,920 026 038	0,920 026 038	$8{,}512\,03 \times 10^{-12}$
72	0,253 823 363	0,253 823 363	$5{,}747\,22 \times 10^{-12}$
80	−0,993 888 654	−0,993 888 654	$-7{,}1359 \times 10^{-11}$
88	0,035 398 303	0,035 398 303	$9{,}703\,16 \times 10^{-14}$
96	0,983 587 745	0,983 587 745	$-3{,}919\,64 \times 10^{-11}$
104	−0,321 622 403	−0,321 622 403	$6{,}318\,61 \times 10^{-12}$
112	−0,889 995 604	−0,889 995 604	$6{,}628\,45 \times 10^{-11}$
120	0,580 611 184	0,580 611 184	$-5{,}399\,95 \times 10^{-13}$

Cuadro 3: Resultados de la aproximación del seno y error relativo para otra serie de datos.$|x| \geq 2\pi$ con una presicion de 10^{-8}

Particion	Seno Aproximado	Seno Original	Error Relativo
8	0,989 358 247	0,989 358 247	$-1{,}122\,16 \times 10^{-16}$
16	−0,287 903 317	−0,287 903 317	$2{,}506\,55 \times 10^{-15}$
24	−0,905 578 362	−0,905 578 362	$2{,}451\,96 \times 10^{-16}$
32	0,551 426 681	0,551 426 681	$1{,}812\,03 \times 10^{-15}$
40	0,745 113 16	0,745 113 16	$-1{,}192 \times 10^{-15}$
48	−0,768 254 661	−0,768 254 661	$8{,}670\,74 \times 10^{-16}$
56	−0,521 551 002	−0,521 551 002	$-7{,}6633 \times 10^{-15}$
64	0,920 026 038	0,920 026 038	$9{,}653\,84 \times 10^{-16}$
72	0,253 823 363	0,253 823 363	$-9{,}8415 \times 10^{-15}$
80	−0,993 888 654	−0,993 888 654	$3{,}351\,15 \times 10^{-16}$
88	0,035 398 303	0,035 398 303	$9{,}703\,16 \times 10^{-14}$
96	0,983 587 745	0,983 587 745	$-5{,}643\,74 \times 10^{-16}$
104	−0,321 622 403	−0,321 622 403	$1{,}1564 \times 10^{-14}$
112	−0,889 995 604	−0,889 995 604	$-2{,}744\,39 \times 10^{-15}$
120	0,580 611 184	0,580 611 184	$6{,}310\,14 \times 10^{-15}$

Cuadro 4: Resultados de la aproximación del seno y error relativo para otra serie de datos.$|x| \geq 2\pi$ con una presicion de 10^{-16}