Ejercicios Aproximaci?n de PI - ALEJANDRO SANTORUM (31-01-2018)

January 31, 2018

EULER - Implementa en Sage ambas formas de aproximar (sumando (7.1) para x = 1 y sumando los valores que se obtienen para x = a y x = b) y compara la eciencia de los dos metodos.

```
In [2]: def arcotangente(x, N):
            res = 0
            for k in xsrange(0, N+1):
                res = res + ((-1)^k * ((x^(2*k+1))/(2*k+1)))
            return res
```

```
In [3]: def valor_aprox_pi(res):
            return 4*res
```

```
In [4]: def metodo(x, N):
            res1 = arcotangente(x, N)
            final_res = valor_aprox_pi(res1)
            return final_res
```

```
In [15]: metodo(1,10000).n(digits=20)
```

```
Out[15]: 3.1416926435905432135
```

```
In [14]: metodo(1,100000).n(digits=20)
```

```
Out[14]: 3.1416026534897939885
```

Se puede ver que avanza bastante lento usando $x = 1$. Vamos a probar ahora para $x = arctan(1/2)$ y $x = arctan(1/3)$

```
In [25]: metodo(arctan(1/2),10000).n(digits=20)
```

```
Out[25]: 1.7365805606736720270
```

```
In [26]: metodo(arctan(1/3),10000).n(digits=20)
```

```
Out[26]: 1.2451603407475466701
```

```
In [27]: metodo((arctan(1/3)+arctan(1/2)),10000).n(digits=20)
```

```
Out[27]: 2.6630950001134154544
```

Aparentemente, utilizando $x = arctan(1/2)$ ó $x = arctan(1/3)$ la sucesión 7.1 no se aproxima a /4 más rápido que para $x = 1$. Debe haber un error en el enunciado del ejercicio.

RAMANUJAN - Utilizando la serie de Ramanujan para obtener un valor aproximado de .

```
In [28]: def suma(N):
             res = 0
             for n in xsrange(0, N+1):
                 res = res + ((((factorial(2*n))^3)*(42*n+5))/(((factorial(n))^6)*(16^(3*n+1)))
             return res
```

```
In [46]: (1/suma(10)).n(digits=100)
```

```
Out[46]: 3.14159265358979323848149991841888029175741301542653207347233124393975960689143541718
```

```
In [47]: (1/suma(100)).n(digits=100)
```

```
Out[47]: 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862
```

```
In [48]: pi.n(digits=100)
```

```
Out[48]: 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862
```

No es difícil ver que su aproximación es bastante buena, y cuanto más crece N mejor es la aproximación. Vamos a intentar estimar cuantas cifras correctas de pi son calculadas dependiendo del valor de N.

```
In [78]: def n_cifras_correctas(N):
             cpi = str(pi.n(digits=500))
             res = (1/suma(N)).n(digits=500)
             cap = str(res)

             ncc=0
             for i in xsrange(0, 501):
                 if(cpi[i] != cap[i]):
                     break
                 ncc += 1
             return ncc
```

```
In [79]: n_cifras_correctas(10)
```

```
Out[79]: 21
```

```
In [80]: def lista_pares(N):
             L1 = [N, N+10, N+20, N+30, N+40, N+50, N+60, N+70, N+80, N+90, N+100, N+110, N+12
             L2 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
             for i in xsrange(0,21):
                 L2[i] = n_cifras_correctas(L1[i])
             LR = zip(L1, L2)
             return LR
```

```
In [81]: L = lista_pares(0) #el primer elemento de cada tupla es N y el segundo el numero de c
         L

Out[81]: [(0, 2),
          (10, 21),
          (20, 38),
          (30, 57),
          (40, 75),
          (50, 94),
          (60, 112),
          (70, 129),
          (80, 147),
          (90, 166),
          (100, 184),
          (110, 202),
          (120, 220),
          (130, 238),
          (140, 256),
          (150, 274),
          (160, 293),
          (170, 310),
          (180, 328),
          (190, 347),
          (200, 365)]
```

Se puede observar que cada vez que aumentamos N en diez unidades, el numero de cifras correctas aumenta entre 17 y 19 unidades, lo cual es bastante.

CHUDNOVSKY - Utilizaremos ahora la serie de Chudnovsky para calcular el valor aproximado de .

```
In [100]: def suma_chud(N):
              res = 0
              for n in xsrange(0, N+1):
                  a = ((-1)^n * factorial(6*n))
                  b = 545140134*n + 13591409
                  c = (factorial(3*n)*(factorial(n)^3))
                  d = (640320^(3*n))
                  res = res + ((a*b)/(c*d))
              return res

In [101]: def lim_chud(N):
              return ((426880*sqrt(10005))/suma_chud(N))

In [102]: lim_chud(1000).n(digits=500)

Out[102]: 3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862

In [109]: def n_cifras_correctas_chud(N, prec):
              cpi = str(pi.n(digits=prec))
```

```
        res = lim_chud(N).n(digits=prec)
        cap = str(res)

        ncc=0
        for i in xsrange(0, prec+1):
            if(cpi[i] != cap[i]):
                break
            ncc += 1
        return ncc
```

```
In [110]: n_cifras_correctas_chud(10, 1000)
```

```
Out[110]: 157
```

```
In [113]: def lista_pares_chud(N, prec):
              L1 = [N, N+10, N+20, N+30, N+40, N+50, N+60, N+70, N+80, N+90, N+100]
              L2 = [0,0,0,0,0,0,0,0,0,0,0]
              for i in xsrange(0,11):
                  L2[i] = n_cifras_correctas_chud(L1[i], prec)
              LR = zip(L1, L2)
              return LR
```

```
In [114]: L = lista_pares_chud(0, 1000)
          L
```

```
Out[114]: [(0, 15),
           (10, 157),
           (20, 298),
           (30, 441),
           (40, 582),
           (50, 724),
           (60, 866),
           (70, 1001),
           (80, 1001),
           (90, 1001),
           (100, 1001)]
```

Podemos concluir con que la serie de Chudnovsky converge mucho mas rapido a que la serie de Ramanujan, debido a que consigue un mayor numero de cifras correctas mucho antes.

RABINOWITZ Y WAGON - Como ya hemos venido haciendo, utilizaremos la serie de Rabinowitz y Wagon para aproximar el valor de .

```
In [116]: def suma_rw(N):
              res = 0
              for n in xsrange(0, N+1):
                  a = factorial(n)^2
                  b = 2^(n+1)
                  c = factorial(2*n+1)
                  res = res + ((a*b)/c)
              return res
```

```
In [119]: suma_rw(100).n(digits=100)

Out[119]: 3.14159265358979323846264338327936493666045842702650045699849794267216024829445373754

In [125]: def n_cifras_correctas_rw(N, prec):
              cpi = str(pi.n(digits=prec))
              res = suma_rw(N).n(digits=prec)
              cap = str(res)

              ncc=0
              for i in xsrange(0, prec+1):
                  if(cpi[i] != cap[i]):
                      break
                  ncc += 1
              return ncc

In [128]: n_cifras_correctas_rw(100, 100)

Out[128]: 32

In [132]: def lista_pares_rw(N, prec):
              L1 = [N, N+10, N+20, N+30, N+40, N+50, N+60, N+70, N+80, N+90, N+100, N+110, N+1
              L2 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
              for i in xsrange(0,21):
                  L2[i] = n_cifras_correctas_rw(L1[i], prec)
              LR = zip(L1, L2)
              return LR

In [133]: L = lista_pares_rw(0, 1000)
          L

Out[133]: [(0, 0),
           (10, 5),
           (20, 8),
           (30, 11),
           (40, 14),
           (50, 17),
           (60, 20),
           (70, 23),
           (80, 26),
           (90, 29),
           (100, 32),
           (110, 35),
           (120, 38),
           (130, 41),
           (140, 44),
           (150, 47),
           (160, 50),
           (170, 53),
```

```
    (180, 56),
    (190, 59),
    (200, 63)]
```

Despues de analizar la serie de Rabinowitz y Wagon podemos sostener que es la peor en comparacion con las series de Ramanujan y Chudnovsky.