Ej 1 y 3 de pi

February 4, 2018

```
In [3]: #Ejercicio 1
```

1. Utilizar la sucesión p n para calcular aproximaciones a 2. Como debemos usar un valor aproximado de para poder calcular p n este método es una tontera. Veremos métodos mejores.

```
In [26]: def arquiemedes(N):
             return n(2.0*N*sin(pi/N))

In [32]: for i in xsrange(990,1001):
             print i, arquiemedes(i), arquiemedes(i)-n(2*pi)

990 6.28317476190833 -0.0000105452712544363
991 6.28317478317966 -0.0000105239999221851
992 6.28317480438670 -0.0000105027928842816
993 6.28317482552970 -0.0000104816498849303
994 6.28317484660892 -0.0000104605706647831
995 6.28317486762462 -0.0000104395549680447
996 6.28317488857704 -0.0000104186025415842
997 6.28317490946646 -0.0000103977131287181
998 6.28317493029311 -0.0000103768864780918
999 6.28317495105725 -0.0000103561223383508
1000 6.28317497175913 -0.0000103354204590289
```

2. £Por qué es convergente la sucesión p n ? Contestar requiere un argumento matemático.

Para realizar este apartado simplemente tenemos que ver el límite de pn = 2nsen(pi/n). Aplicando L'hopital no es dificil ver que el límite cuando pn tiende a infinito es 2pi.

3. £Cómo podemos ver en las aproximaciones f(x) numéricas que la sucesión es de Cauchy? Dado que es convergente como hemos visto antes podemos deducir que es de cauchy. Además si recurrimos a su definición, vemos que si restamos los valores que tenemos, la diferencia es cada vez menor y siempre existirá un epsilon que cumpla la condición.

```
In [ ]: #Ejercicio 3
```

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 eficiencia de los dos métodos.

```
In [91]: def tay_arctan1(x,n):
             res = 0
             for i in xsrange (1,n+1):
                 res = res + ((-1) ^ (i+1))* (x^(2*i-1)/(2*i -1))
             return res

In [92]: %time
         for i in sxrange (990, 1001):
             print i,n(tay_arctan1(1,i)), n(pi/4) - n(tay_arctan1(1,i))

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 9.06 µs
990 0.785145638209336 0.000252525188112140
991 0.785650433767135 -0.000252270369686891
992 0.785146147332440 0.000252016065008021
993 0.785649925669972 -0.000251762272523548
994 0.785146654406761 0.000251508990687377
995 0.785649419615408 -0.000251256217960183
996 0.785147159444640 0.000251003952808304
997 0.785648915591153 -0.000250752193704518
998 0.785147662458321 0.000250500939127596
999 0.785648413585011 -0.000250250187562528
1000 0.785148163459948 0.000249999937500078


In [93]: %time
         for i in sxrange (990, 1001):
             print i,n(tay_arctan1(1/2,i)+tay_arctan1(1/3,i)), n(pi/4) - n(tay_arctan1(1/2,i)+t

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 15 µs
990 0.785398163397448 0.000000000000000
991 0.785398163397448 0.000000000000000
992 0.785398163397448 0.000000000000000
993 0.785398163397448 0.000000000000000
994 0.785398163397448 0.000000000000000
995 0.785398163397448 0.000000000000000
996 0.785398163397448 0.000000000000000
997 0.785398163397448 0.000000000000000
998 0.785398163397448 0.000000000000000
999 0.785398163397448 0.000000000000000
1000 0.785398163397448 0.000000000000000
```

Podemos ver que usando lo dicho en el enunciado (arctan(1/2) y arctan(1/3)) se aproxima mejor a pi/4, pero tarda algo más que usando taylor en 1.