Ejercicio 16 (kappa again), 17 y 18

February 4, 2018

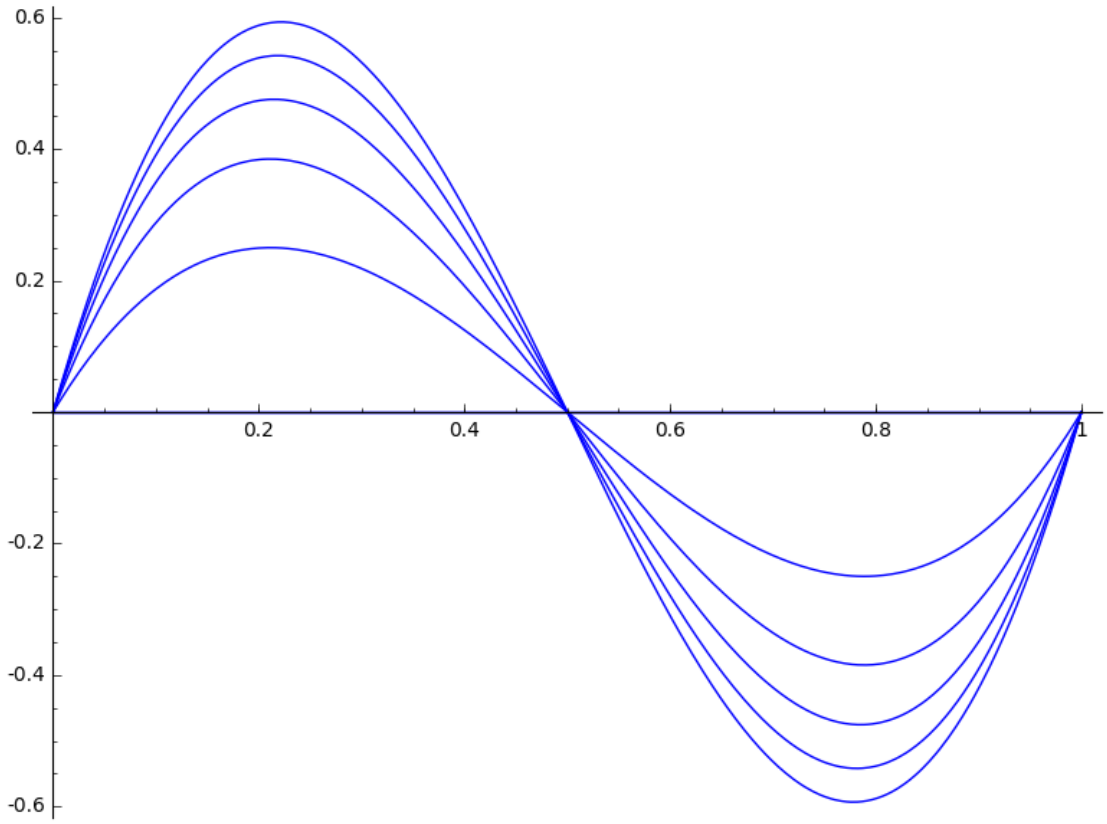
Ejercicio 17. 1. Define una función de Sage que dependa de un entero n y una función f y devuelva B n,f (x). 2. Experimenta, mediante gráficas, con diversas funciones f , continuas en [0, 1] y sus aproximaciones. Por ejemplo, podemos tomar f (x) = sen(2x), f (x) = sen(4x), etc. 3. Trata dar una definición razonable que precise en qué sentido B n,f (x) aproxima a f (x) globalmente en el intervalo [0, 1].

```
In [14]: def berstein(n,f):
             res = 0
             for i in xsrange(0,n+1):
                 res = res + binomial(n,i)*f(i/n)*((1-x)^(n-i))*x^i
             return res
```

```
In [16]: sum([plot(berstein(i,sin(2*pi*x)),0,1) for i in xsrange(1,8)])
```

```
/usr/local/SageMath/local/lib/python2.7/site-packages/sage/repl/ipython_kernel/__main__.py:1: I
See http://trac.sagemath.org/5930 for details.
  from ipykernel.kernelapp import IPKernelApp
```

Out[16]:



3. Podemos ver que en el intervalo [0,1] queda definida la forma que tendrá la función durante todo R

Ejercicio 18. El número e se puede obtener, entre otras, de una de las siguientes maneras

Este ejercicio trata de calcular valores aproximados del número e usando cada una de las expresiones anteriores y de estudiar el coste computacional de los cálculos, usando, por ejemplo. time o timeit, tratando de responder a la pregunta natural: £Cuál es el mejor método?

```
In [19]: def met1(n):
             return ((1 + (1/n))^n).n()
         def met2(n):
             res = 0
             for i in xsrange(n+1):
                 res = res + (1/factorial(i))
             return res.n()
         def met3(n):
             parte1 = (n^n)/((n-1)^(n-1))
             parte2 = ((n-1)^n-1)/((n-2)^(n-2))
             return (parte1 - parte2).n()

In [20]: %%time
         met1(1000)
```

```
CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 1.69 ms


Out[20]: 2.71692393223589

In [21]: %%time
         met2(1000)

CPU times: user 100 ms, sys: 4 ms, total: 104 ms
Wall time: 104 ms


Out[21]: 2.71828182845905

In [22]: %%time
         met3(1000)

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 1.94 ms


Out[22]: -2.70877316541795e6
```

Podemos ver que los límites son más rápidos que la serie infinita