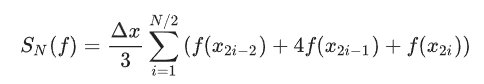
[Simpson's rule](https://en.wikipedia.org/wiki/Simpson%27s_rule)

 uses a quadratic polynomial on each subinterval of a partition to approximate the function **f(x)** and to compute the definite integral. This is an improvement over the trapezoid rule which approximates **f(x)**by a straight line on each subinterval of a partition.

The formula for Simpson's rule is



where N is an even number of subintervals of [a,b], Δx=(b−a)/N and xi=a+iΔx.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## ***Error Formula***

**Theorem** Let **SN(f)** denote Simpson's rule

**SN(f)=Δx3∑i=1N/2(f(x2i−2)+4f(x2i−1)+f(x2i))**

where N is an *even* number of subintervals of **[a,b], Δx=(b−a)/N and xi=a+iΔx.** The error bound is

**ENS(f)=| ∫abf(x)dx−SN(f) |≤(b−a)^5/180N^4K^4**

where **| f(4)(x) |≤K4 for all x∈[a,b].**

## ***Implementation***

Let's write a function called simps which takes input parameters **f, a, b** and N and returns the approximation **SN(f).** Furthermore, let's assign a default value **N=50.**

**def** **simps**(f,a,b,N=50):

'''Approximate the integral of f(x) from a to b by Simpson's rule.

Simpson's rule approximates the integral \int\_a^b f(x) dx by the sum:

(dx/3) \sum\_{k=1}^{N/2} (f(x\_{2i-2} + 4f(x\_{2i-1}) + f(x\_{2i}))

where x\_i = a + i\*dx and dx = (b - a)/N.

Parameters

----------

f : function

Vectorized function of a single variable

a , b : numbers

Interval of integration [a,b]

N : (even) integer

Number of subintervals of [a,b]

Returns

-------

float

Approximation of the integral of f(x) from a to b using

Simpson's rule with N subintervals of equal length.

Examples

--------

>>> simps(lambda x : 3\*x\*\*2,0,1,10)

1.0

'''

**if** N % 2 == 1:

**raise** ValueError("N must be an even integer.")

dx = (b-a)/N

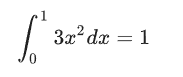
x = np.linspace(a,b,N+1)

y = f(x)

S = dx/3 \* np.sum(y[0:-1:2] + 4\*y[1::2] + y[2::2])

**return** S

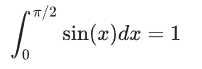
Let's test our function on integrals for which we know the exact value. For example, we know



simps**(lambda** **x : 3\*x\*\*2,0,1,10)**

1.0

Test our function again with the integral



simps(np.sin,0,np.pi/2,100)

1.0000000003382361

## **scipy.integrate.simps**

The SciPy subpackage scipy.integrate contains several functions for approximating definite integrals and numerically solving differential equations. Let's import the subpackage under the name spi.

**import** scipy.integrate **as** spi

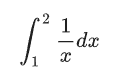
The function scipy.integrate.simps computes the approximation of a definite by Simpson's rule. Consulting the documentation, we see that all we need to do it supply arrays of x and yvalues for the integrand and scipy.integrate.simps returns the approximation of the integral using Simpson's rule.

-----------------------------------------------------------------------------------------------------------

## ***Examples***

### **Approximate ln(2)**

Find a value N which guarantees that Simpson's rule approximation SN(f) of the integral

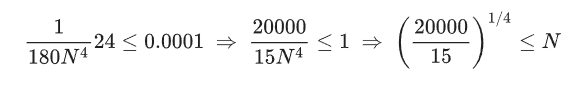


satisfies **ENS(f)≤0.0001**.

Compute



therefore



Compute

(20000/15)\*\*0.25

6.042750794713537

Compute Simpson's rule with **N=8** (the smallest even integer greater than 6.04)

approximation = simps**(lambda x : 1/x,1,2,8)**

print(approximation)

0.693154530655

We could also use the function scipy.integrate.simps to compute the exact same result

**N = 8; a = 1; b = 2;**

**x = np.linspace(a,b,N+1)**

**y = 1/x**

approximation = spi.simps(y,x)

print(approximation)

0.693154530655

Verify that



np.abs(np.log(2) - approximation) <= 0.0001

True

## **Exercises**

Under construction