311 – Numerical Computations
Lab 11: Numerical differentiation and Integration

A)Background:

- Linspace:

```
import numpy as np


def f (x):
    return x+2
x=np.linspace(1,4,4)
print(f(x))                        #valid in NumPy
print(sum(f(x)))
```

output:

[3. 4. 5. 6.]

18.0

=================================

- List Slicing:

```
L=[4,14,7,3,1,9, 8,2]
print(sum(L[2:5]))    #prints 11  (7+3+1)
print(sum(L[:5]))     #prints 29 (4+14+7+3+1)
print(sum(L[4:]))     #prints 20 (1+9+8+2)
print(sum(L[2:6:2]))  #prints 8 (7+1)
```

==

B) Numerical Differentiation

Remember

Differentiation formulas:

central: f(a+h) - f(a-h))/ (2h)
forward: f(a+h) - f(a))/h
backward: f(a) - f(a-h))/h

```
from scipy.misc import derivative
def f(x):
    return x*x
d = derivative(f, 1.0, dx=0.01)
print (d)
```

Remark: Scipy uses Central formula

C)Numerical Integration:

Background: What is an elementary function?

an elementary function is a function of a single variable that is defined as taking sums, products, and compositions of finitely many polynomial, rational, trigonometric, hyperbolic, and exponential functions, including possibly their inverse functions.

Theorem: The antiderivatives of certain elementary functions cannot themselves be expressed as elementary functions.

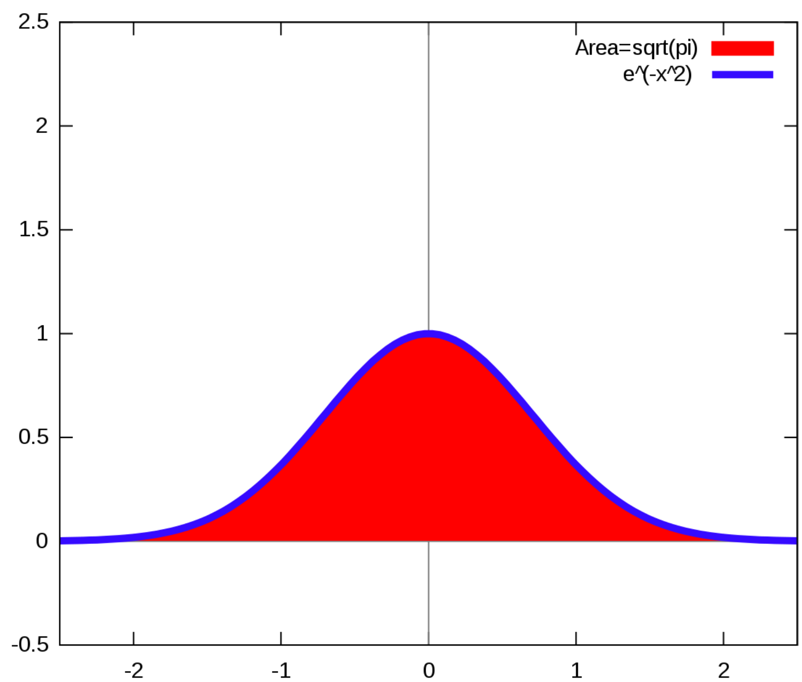
Want to read more about this theorem?

Liouville's theorem (differential algebra) - Wikipedia

Or Liouville's theorem on functions with elementary integrals. (projecteuclid.org)

One of the famous functions that has no antiderivative: e^{-x^2}

(Conclusion: We need Numerical Integration for different reasons, the above discussion is one of them)



- **Trapezoidal rule:** $= \frac{h}{2}\left(y_0 + 2y_1 + 2y_2 + \ldots + 2y_{n-1} + y_n\right)$

Compute $\int_0^{\pi} \sin(x)\, dx$ using trapezoidal rule with n=10 equally spaced intervals:

a) Using Scipy b) Directly using the formula

```
import numpy as np
from scipy.integrate import trapz
a = 0
b = np.pi
n = 10
#--------------------------- a -----------------------------
x = np.linspace(a, b, n+1)
f = np.sin(x)                              #try   print(f)
print(trapz(f,x))
#----------------------------- b ----------------------------
h = (b - a) / n
print((h/2)*(f[0] + 2 * sum(f[1:n]) + f[n]))
```

Another example:

```
import numpy as np

x = np.array([0, 0.5, 1, 1.5, 2])

y = [a * a for a in x]      #equivalently in NumPy you can say: y = x*x

print(y)

print(np.trapz(y, x))
```

Output: [0. 0.25 1. 2.25 4.]

2.75

- **Simpsons 1/3 rule**

Compute $\int_0^{\pi} \sin(x)\, dx$ using **1/3 Simpson's rule,**
with n=10 equally spaced intervals:

a) Using Scipy
b) Directly using the **1/3 Simpson's rule** formula:

$$= \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n/2-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(x_n)\right],$$

```
import numpy as np
from scipy import integrate
a = 0
b = np.pi
n = 10
x = np.linspace(a, b, n+1)                #we need n+1 points
y = np.sin(x)
print(y)
#-----------------------    a    ------------------------------
print(integrate.simpson(y, x))

#------------------------  b    -----------------------------------
h = (b - a) / (n)
print((h/3) * (y[0] + 2*sum(y[2:n-1:2]) + 4*sum(y[1:n:2]) + y[n]))
```

Output:

```
[0.00000000e+00 3.09016994e-01 5.87785252e-01 8.09016994e-01
 9.51056516e-01 1.00000000e+00 9.51056516e-01 8.09016994e-01
 5.87785252e-01 3.09016994e-01 1.22464680e-16]
2.0001095173150043
2.0001095173150043
```

Lab 11 Task:

Write a program that:

- Reads two integers (integration limits)
- Reads n, number of intervals (must be a multiple of 3, you may assume that) and applies Simpson's 3/8 rule over the n equally spaced intervals for the function f(x) = (x+7)/x (without calling any library function except np.linspace).

Simpson's 3/8 rule

$$= \frac{3h}{8}\left[f(x_0) + 3\sum_{i \neq 3k}^{n-1} f(x_i) + 2\sum_{j=1}^{n/3-1} f(x_{3j}) + f(x_n)\right]$$

Suppose n=9

f(x9)

3(f(x1) + f(x2) + f(x4) + f(x5) + f(x7) + f(8))

2(f(x3) + f(x6))

Print Final Result rounded to 5 decimal Places.

Sample I/O:

Input	Result
4 8 6	8.85237