CSCI 2202 Python for Scientists Lab E 18 March 2021
Integration - 2

In this lab:
(i) Vectorization: Applied to the Mid-point and Trapezoidal rules
(ii) Mote-Carlo Integration

(1) **Vectorization:**
Long loops in Python can be very slow for complicated implementations. The loops can be sped up using *vectorization*. Vectorization replaces the loop with an optimized, pre-compiled code, written in low-level language like C is used to perform mathematical operations over a sequence of data (like a `numpy` array).
Example: We will use the `time` module in Python to time loops and compare it to the timing of vectorized code. In this code, we are simply multiplying two lists in a for loop and then using `numpy` multiplying two `numpy` arrays using *vectorization*

```
import time
import numpy as np


def mult_lists(li_a, li_b):
for i in range(len(li_a)):
li_a[i]*li_b[i]


def mult_arr(arr_a, arr_b):
arr_a*arr_b


  #Call the list multiplication by loop function
start_time1 = time.time()
mult_lists(li_a, li_b)
print(time.time() - start_time1)


  #Call the vectorized array multiplication function
start_time2 = time.time()
mult_arr(arr_a, arr_b)
print(time.time() - start_time2)
```

Use this idea to vectorize the loop in `midpoint.py`:

- Use `linspace` (from `numpy`) to compute all the points at which the `integrand` is being evaluated. (Use about a 1000 intervals.) This is an array of x values (`x = linspace(... )` will automatically create the array.
- Call `f(x)` (the integrand defined in a function). This will produce an array of the corresponding function values.
- Use the `sum` function to sum the $f(x)$ values.

• The evaluation points in the midpoint method are $x_i = a + (i + \frac{1}{2}) * \Delta x$ for $i = 1, 2, \ldots (n-1)$. This can be calculated by `np.linspace(a + h/2, b-h/2, n)`. $f(x)$ will produce all the function values in an array. Then `sum(f(x))` will sum up the elements in the array. Finally multiply by Δx to find the answer. (Any math function you need, use the version from numpy.)
Use the above to compute $\int_0^2 e^{x^2} dx$.
Evaluate the integral with the vectorized and the un-vectorized `midpoint` code. Report the time taken for each.

• Repeat the same for the trapezoidal method: Here the only complication is that the the sum has different weights for the end-points: ($f(a)\&f(b)$ have weight 1, while all the other terms have weight 2). A simple way is to `sum` as above but hen subtract off the function evaluated at the end-points: `sum(f(x)) - f(a) -f(b)`.
Use the above to compute $\int_0^2 e^{x^2} dx$.
Evaluate the integral with the vectorized and the un-vectorized `trapezoidal` code. Report the time taken for each.

(2) **Monte-Carlo Integration**

The problems below are Monte Carlo (MC) approximations to integrals. They are similar to the MC approximation to π, covered in lab 6. *The only change is that* you will need to multiply the ratio of "# Hits" to "Total # throws" (N) by the area (or volume) of the box formed around the function being integrated. This is because the earlier simulations were done over a unit area. *(a)* below details the approach.

(a) Compute $\int_{1/2}^{3/2} \sqrt{x}dx$, using the Monte Carlo technique. Generate random x, y values in the range: $\frac{1}{2} \leq x < \frac{3}{2}$ and $0 \leq y < 2$. The upper bound of y, the value 2, needs to be large enough to include the max. $y-$extent of $f(x) = \sqrt{x}$ (*this is one of the weaknesses of the MC method for integrals - this upper bound needs to be guessed/figured out.*). Run your simulation for $N = 10000$ *and* 100000. Have your program print the approximate value of the integral and the relative error. The exact answer is 0.98904261.

(b) Using MC simulation, find the area between the two curves: $y = x^2$ & $y = 6 - x$ and the x and y axes (i.e.for $x \geq 0$ & $y \geq 0$). It will pay to sketch the two curves first before attempting the simulation.

(c) Use MC simulation to find the volume of the part of the ellipsoid: $\frac{x^2}{2} + \frac{y^2}{4} + \frac{z^2}{8} \leq 16$, that lies in the first octant (*i.e.* $x > 0,\ y > 0,\ z > 0$).