Roll No: 20PH20014
Name: Jessica John Britto

Computational Physics Lab Report-1

Aim:

1. Encode an algorithm to determine the smallest positive number that can be represented on the computer you are using and run it. Do this for both single precision and double precision floating point numbers.
2. Evaluate the expression $y = (x^2 + 1.0)^{0.5} - 1$ in two ways given below -
   a. $y = (x^2 + 1.0)^{0.5} - 1$ A
   b. $y = \frac{x^2}{(x^2+1.0)^{0.5}+1.0}$ B

   for small values of x such as $0.1,\ 0.01,\ 0.001,\ 10^{-4}$ and so on, and evaluate the fractions deviation $\frac{A-B}{B}$. Plot the fraction deviation as a function of x on a log-log plot. Discuss which method is expected to be superior.

Tools Used: Jupyter Notebook, Python, NumPy, Pandas, Matplotlib.

Theory:

1. Single precision is a format for the representation of floating-point numbers. It occupies 32 bits in computer memory, where 23 bits are used for mantissa and 8 bits for exponent.
2. Double precision In double precision, 64 bits are used to represent a floating-point number, where 52 bits are used for mantissa and 11 bits are used for exponent.
3. Machine Epsilon is the smallest number of EPS (epsilon) such that (1 + EPS) is not equal to 1.
4. Machine Epsilon is a machine-dependent floating point value that provides an upper bound on the relative error due to rounding in floating point arithmetic. Mathematically, for each floating point type, it is equivalent to the difference between 1.0 and the smallest representable value that is greater than 1.0.
5. In Python3, the information is available in **sys.float_info**, which corresponds to float.h in C99.

Observations:

For problem-1:

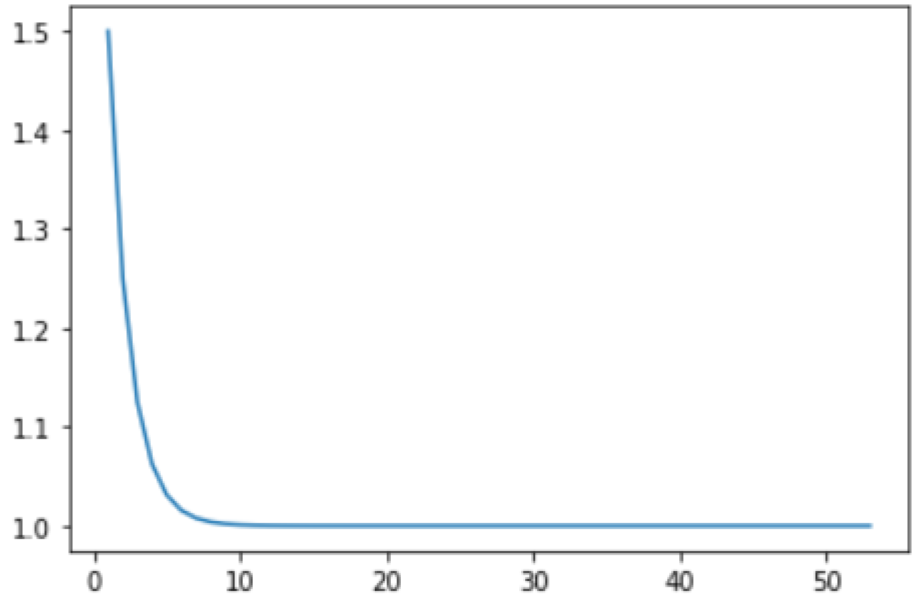
Algorithm for finding the Machine epsilon is given in the form of pseudocode below -

```
input s <--- 1.0
for k=1,2,3,...,100 do
      s <--- 0.5 s
      t <--- s + 1.0
      if t <= 1.0 then
            s <---  2.0  s
            output k-1, s
            stop
      endif
end
```

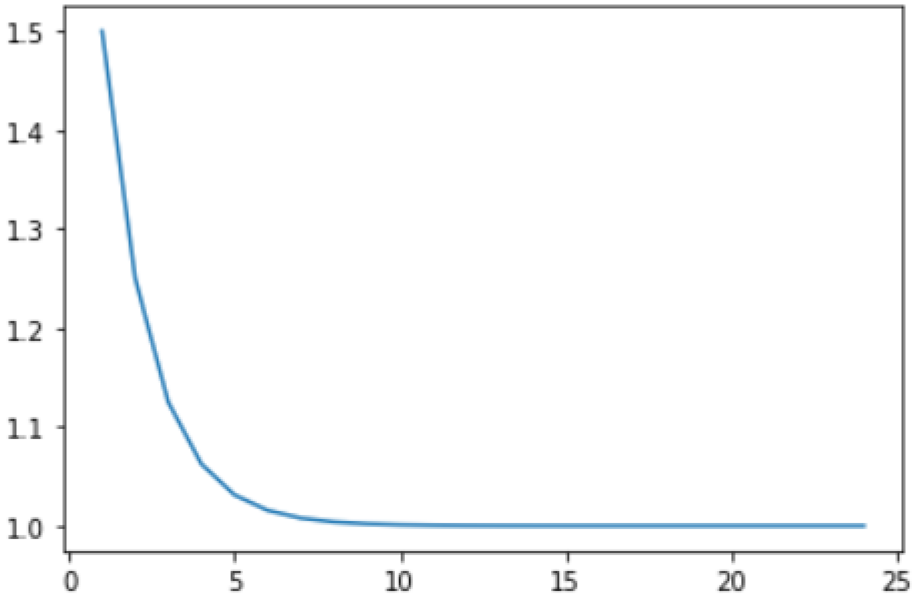
As shown in the pseudocode, firstly we initialize s to 1.0 where s is a variable that is used for storing the machine epsilon value. With each iteration, the value of s decreases by a factor of (0.5). At some stage during the iteration, it will become zero once its value cannot be stored by the system as its smaller than the number of decimals the system can represent a floating point number. At this stage, t equals to 1, [as s is zero], then the for-loop breaks, and (k-1) stores the value of the iteration in which the least value of s can be stored by the system.

Graphs-

For Double precision-

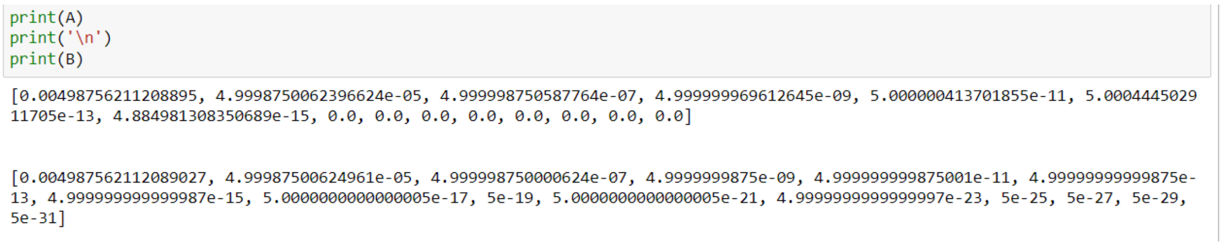


For Single precision-



For problem-2:

Method **(B)** is superior to the method **(A)** to find the given expression and this is evident from below - (where A is the list containing values a (of equation A), and B is the list containing values b (of equation B))



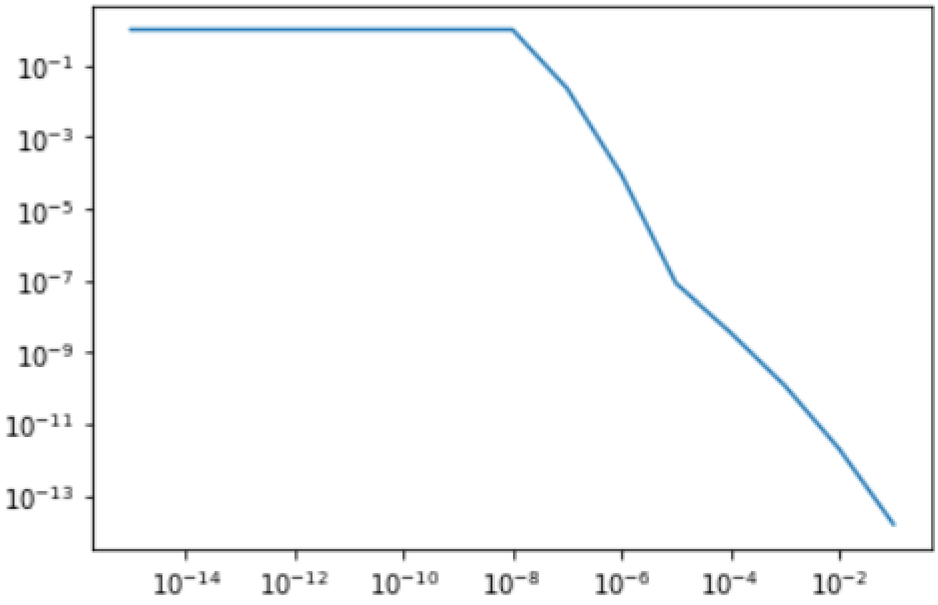
```
print(A)
print('\n')
print(B)
```

```
[0.00498756211208895, 4.9998750062396624e-05, 4.999998750587764e-07, 4.999999969612645e-09, 5.000000413701855e-11, 5.0004445029
11705e-13, 4.884981308350689e-15, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]


[0.004987562112089027, 4.99987500624961e-05, 4.999998750000624e-07, 4.9999999875e-09, 4.999999999875001e-11, 4.99999999999875e-
13, 4.999999999999987e-15, 5.0000000000000005e-17, 5e-19, 5.0000000000000005e-21, 4.9999999999999997e-23, 5e-25, 5e-27, 5e-29,
5e-31]
```

Graphs-

Plot of $\frac{y_a - y_b}{y_b}$ vs x in the log-log form-



The plot becomes constant for values of $x <= 10^{-8}$.

Results:

1. In problem-1, the iteration at which the s becomes zero for the double precision case is **52**, and the machine epsilon value is 2.220446049250313e-16, which is the same as the value obtained through using this code -
   import sys
   sys.float_info.epsilon
2. In problem-1, the iteration at which the s becomes zero for the single precision case is **23**, and the machine epsilon value is 1.1920929e-07.
3. In problem 2, method **(B)** is superior to the method **(A)** to find the given expression, as, during the iterations, the first equation **(A)** becomes zero before the second equation **(B)** during the iteration.
4. In problem-2, from the graph, its clear that $\frac{A-B}{B}$ is close to 1 for $x <= 10^{-8}$.