Numerical Analysis HW 2

Michael Vu

August 31, 2023

Problem 1. In the case for determining the largest real value, Tom and Jim's codes produced drastically different results from one another. For my setup, I am running Windows 10, an AMD FX 8350 CPU, and my compiler is WinGW. Under these conditions, Tom's code where he multiplies the x by 2 for each iteration, the largest output is approximately 9.22×10^{18}. Whereas Jim's code where he adds x to itself for each iteration, the largest output is over twice as large, approximating 1.70×10^{38}. Jim's code also resulted in a "Note: The following floating-point exceptions are signalling: IEEE_OVERFLOW_FLAG".

I decided to run both of their codes in Linux Mint 18 using the same hardware. Jim's code resulted in the same output as Windows 10/WinGW with the same message at the end of the program. However, Tom's code result was much different. Tom's code only output up to 8388608.00. Regardless of the operating system, both codes produced very different results from one another.

Problem 2. To explain the Tom and Jim situation is not something, I truly understand. My best guess is that it has to do with the number of digits being computed, rather than the actual value itself. For example, it would seem to me that storing the data $123456 + 123456$ would require more bits than 123456×2.0, thus adding the number to itself automatically creates more space, which in turn, allows more digits.

Problem 3. After amending the codes from problem 1 for both Tom and Jim, I ran the programs and the results are as I predicted for Tom, but not what I expected for Jim. After setting the real variables to double precision, here is Tom's code:

```
program tom2
implicit none
double precision :: x
integer :: i

x = 1.0

do while (x + 1.0d0 /= x)
print*, x
x = x * 2.0d0
end do

stop
end program
```

The output from this code results in the same amount of digits as the previous code, however, more digits are displayed with the modified code. His original code resulted with a max value of $9.22337204 \times 10^{18}$ and the modified code resulted with a max value of $9.2233720368547758 \times 10^{18}$, twice as many digits, but approximately the same value. When ran in Linux Mint 18, the original max value was 8388608.00, and the max value for the modified code was 4503599627370496.0, which is significantly larger with twice as many digits. Looking at Jim's modified code:

```
program jim2
implicit none
double precision :: x
integer :: i

x = 1.0

do while (x + x /= x)
print*, x
x = x * 2.0d0
end do

stop
end
```

The output from this code results in a value significantly larger than the original code. Jim's original code had a max value of $1.70141183 \times 10^{38}$, and the modified code resulted in a max value of $8.9884656743115795 \times 10^{307}$, over 8 times larger than the original max value. When ran in Linux Mint 18, Jim's modified code matched exactly as when ran in Windows 10.