Homework 1

Ddongwook - Math 19B

Due: Jan 18th, 2024

1. My choice of programming language for this class will be fortran.

2. Exercises 12-16 from Fortran Tutorial by P. Garaud:

    12. The code returns a line to the terminal which reads the matrix by going down each column first and then moving to the column to the right. There is also a value returned which is not supposed to be in the printed matrix. It reads -7.822 * 10^{33}. NOTE: the values which are incorrect are random, on a different run there were two incorrect values and they did not have the same value as the first.

    13. The code now returns the matrix in a grid pattern corresponding to the indicies of the array. Hence the (i,j) element of the array prints in the (i,j) cell of the printed grid. There are still incorrect values in the output. Often occuring in the first column of the matrix.

    14. I chose to zero all three matrices before any values were entered into a and b. From three different runs of the code, this seems to have corrected the bug which returned incorrect values in the matrix.

    15. After running both codes, I can see the difference between the intrinsic operators and the manual ones. Comparing the run times, I can see that performing the matrix addition manually more than triples the runtime for the code. The intrinsic operation completes in 0.71 seconds while the manual operation completes in 2.3 seconds.

    16. The code seems to be running properly at first glance. The output is 1.571... which is very close to pi/2. The way the problem is stated implies that there should be an error in the code somewhere. I am going to try an adjacent period of cos to see if an issue appears.Then I will try to include multiple periods (this should break the code since there would be multiple roots). Then I will try a simple cubic function and maybe an exponential decay problem. Looking at the adjacent root, this algorithm computed it correctly, getting the first 4 digits accurately. I will now

try multiple roots. With multiple roots, the code returns that there are either none or multiple roots in the interval, and computes only one of the roots. I could modify it to compute both roots. After attempting this with a different function, $f(x) = x^3 - 1$, no discrepancies can be foundd other than the code caan only approximate the value. The output was .9995 which is very very close to 1. After widening the interval so that the code would have a harder time finding it, it still obtained a correct answer to 2 decimal places. Maybe if the root is in the midpoint of the interval it is easier, I will make it a third of the way through the interval. Even then it gave an answer only very slightly differnt from the last and only exhibits a 3% error. It is unclear where this code should be faulty. I will try one more function, tan(x). Testing tan(x) and intentionally placing the discontinuity in the interval, I was able to have the code find the discontinuity rather than a root without an error message. Extending the interval to include the discontinuity and a root, the code returned the root rather than the discontinuity. This might be by chance or an artifact of the algorithm design. Either way, the method of bisection would not be able to differentiate between an infinite discontuity which changes sign and a root of a function. I cannot find an error in the code, so I will not change it.

3. The results of my code are odd but not for a reason I currently understand. For a threshold of 10^{-4} the code took 3 iterations to complete and returned a difference of $.52632 \cdot 10^{-5}$. For a threshold of 10^{-8} the code took 5 iterations to complete and returned a difference of $.81295 \cdot 10^{-8}$. For a threshold of 10^{-12} the code took 8 iterations to complete and returned a difference of $.82023 \cdot 10^{-12}$. For a threshold of 10^{-16} the code took 11 iterations to complete and returned a difference of $.00000 \cdot 10^{00}$. The last output which is correlated to the lowest threshold, it makes sense that the difference is registered at 0. This would be because double precision variables in fortran go out to only 16 decimal places. Thus, when the threshold is so low, fortran can no longer distinguish the variables. The bug in my code which I cannot understand is that as soon as I take away the print statments from the while loop, the code breaks. It seems that the issue is part of the while loop. If there is no interaction with the terminal inside the loop, none of the arithmatic is actually performed. LATER, the bug was later found to be an issue with the line length limit of the compiler and an error in a number type (raising the integer 16 to a negative power rather than the real 16 to a negative power).

4. Academic Integrity Statement

   "As a student at UC Santa Cruz, I hold myself to a high standard of academic integrity. By signing this statement, I affirm my commitment to honor the UC Santa Cruz Code of Student Conduct and to encourage other students to do the same. I pledge that the work I submit on homework and exams will be my own. I will not solicit help from anyone, I will not consult unauthorized sources, and I will

not provide unauthorized help to others. I am aware that failure to abide by these commitments may result in sanc- tions ranging from no credit for the work in question to failing the class, to suspension from the university." Signature: Dante Buhl