Problem Set 1 - ECON 880

Spring 2022 - University of Kansas

Minh Cao, Gunawan

February 2, 2022

Problem 1

(1a)

In this question, we will use the Multivariable Honer's Algorithm:

- Step 1: Let call our original multivariable polynomial $f(x, y) = M_1 + M_2 + M_3 + M_4 + M_5$ for M_i is the monomial.

- Step 2: For each monomial M_i, we count, for each variable x_i , the number of monomials M_i that contain this variable. We then select a variable with the largest value of M_i, and represent the original polynomial as $f(x, y) = A_0 + A_1 x_i$. Where A_0 is the monomial that not contain x_i, and A_i is the sum of all the terms ${M_k}' = \frac{M_k}{x_i}$ corresponding to all the monomialsM_k that do contain x_i.

- Step 3: Repeat the scheme.

Using the scheme above.
Let $M_1 = 8351y^8, M_2 = 572x^2y^4, M_3 = -2x^4, M_4 = 2x^6, M_5 = -x^8$
It is clear that, in the first step, x appears 4 times, and y appears 2 times. So per the scheme, we pick x for our factor.

$$f(x, y) = 83521y^8 + (572y^4 - 2x^2 + 2x^4 - x^6)x^2$$

Now: $A_0 = 83521y^8$ and $A_1 = 572y^4 - 2x^2 + 2x^4 - x^6$. Repeat the step until we get:

$$f(x, y) = 83521y^8 + x^2(572y^4 + x^2(-2 - x^2))$$

Result: For naive approach $f(x, y) = -2.1778 * 10^{40}$
Multivariable Honer approach: $f(x, y) = -2.1778 * 10^{40}$
So we get the same result for both method.

(1b)

We estimate the order of magnitude (number of digits) of

$$83521y^8, \tag{1}$$

where $y = 2298912$. We can rewrite the expression as follows:

$$\begin{aligned} 83521y^8 &= 83521 \cdot 2298912^8 \\ &= 8.3521 \cdot 10^4 \cdot (2.298912 \cdot 10^6)^8 \end{aligned}$$

Then, the digits can be estimated by using the following formula

$$\begin{aligned} 1 + \lfloor \log_{10}(83521 \cdot 2298912^8) \rfloor &= 1 + \lfloor \log_{10}(83521 \cdot 2298912^8) \rfloor \\ &= 1 + \lfloor \log_{10}(8.3521 \cdot 10^4 \cdot (2.298912 \cdot 10^6)^8) \rfloor \\ &= 1 + \lfloor \log_{10} 8.3521 + \log_{10} 10^4 + \log_{10} 2.298912^8 + \log_{10}(10^6)^8 \rfloor \\ &= 56 \end{aligned}$$

Now, we compute the last digits of that term, first by plugging in only the last digit of y into expression (1): our calculation gives 21,312,256 as a result. However, printing out the complete term using the command fprintf$('\%.0f\backslash n', 83251 * y^8)$, we obtain the following 56-digit number:

$$64, 948, 367, 112, 155, 225, 808, 364, 994, 454, 077, 172, 753, 899, 719, 772, 317, 155, 328.$$

We see that both results have different last digit ($6 \neq 8$).
Comment: We can conclude that the "guess" method maybe wrong because the number of digit of the term is 56 less than the maximum that a 64 bit system can represent. Theoretically, the solution of the computer must be true.

(1c)

Yes, the 64 bit can find the correct value since the number of digit is less than the maximum number of digit the system can represent.
The orders of magnitude of other terms are at most 56.

Problem 2

In this exercise, we write an algorithm to determine the relative speeds of addition, multiplication, division, exponentiation, and the logarithmic function of our computer. The computer uses Intel(R) Core(TM) i5-1035G4 CPU @ 1.10GHz, 1.50 GHz with 64-bit operating system and 8.00 GB installed RAM. We proceed by generating two matrices A and B of size $10^4 \times 10^4$ using the rand() function in Matlab. Then, we use them in element-wise operations of addition ($A + B$), multiplication ($A. * B$), division ($A./B$), exponentiation ($A.^B$), as well as the logarithmic function ($\log(A)$). The statements `tic` and `toc` are used around them to measure the computation time. This procedure is iterated 100 times, and the average computation time for each operation is computed. The results are as follows:

- Average computation time for variable initiation is 1.54030 seconds
- Average computation time for addition is 0.12073 seconds

- Average computation time for multiplication is 0.12083 seconds
- Average computation time for division is 0.11823 seconds
- Average computation time for exponentiation is 3.70586 seconds
- Average computation time for log function is 0.91528 seconds

Problem 3

In order to find our machine ε, we follow Ken Judd's definition[†] by writing a while loop to subtract (resp. add) progressively smaller numbers ϵ from (resp. to) 1 until the condition $1+\epsilon > 1 > 1-\epsilon$ is no longer satisfied. Repeat the exercise using 0.001 and 1000 instead of 1. We verify our results by comparing them with the epsilons delivered by the built-in Matlab function `eps(x)`, and the exponents with `log2(eps(x))`. The results are shown on Table 1.

x	$\varepsilon(x)$ - decimal	$\varepsilon(x)$ - exponential
0.001	2.16840434497101e-19	2^{-62}
1	2.22044604925031e-16	2^{-52}
1000	1.13686837721616e-13	2^{-43}

Table 1: Machine ε for diverse values of x

Comment: As x increases, the machine $\varepsilon(x)$ also does increase. This is to be expected, since $\varepsilon(x) \approx x\varepsilon(1)$. Thus, $\varepsilon(1000) > \varepsilon(1) > \varepsilon(0.001)$.

Problem 4

We wrote a loop to evaluate the convergence of the following sequences:

(4a) $x_k = \sum_{k=1}^{n} \frac{1}{2^n}$, where $\lim_{k\to\infty} x_k = 1$

(4b) $y_k = \sum_{k=1}^{n} \frac{1}{n}$, where $\lim_{k\to\infty} y_k = \infty$

We use absolute and relative convergence criteria, and tolerance distance $\delta \in \{10^{-2}, 10^{-4}, 10^{-6}\}$ as stopping rule. We limit the maximum number of iterations to 100,000. The number of iterations before convergence, as well the final guess are reported on Table 2 and 3 for x_k and y_k, respectively.

δ	10^{-2}	10^{-4}	10^{-6}
no. of iteration - absolute criteria	7	14	20
final guess - absolute criteria	0.992187500000000	0.999938964843750	0.999999046325684
no. of iteration - relative criteria	7	14	20
final guess - relative criteria	0.992187500000000	0.999938964843750	0.999999046325684

Table 2: Convergence for x_k

Comment: Table 2 shows that both absolute and relative convergence criteria lead to the same number of iterations for the sequence x_k. As tolerance distance δ lowers, the final guess for x_k

[†]Kenneth L. Judd, 1998. "Numerical Methods in Economics," MIT Press Books, The MIT Press, p.30

δ	10^{-2}	10^{-4}	10^{-6}
no. of iteration - absolute criteria	100	10,000	100,000
final guess - absolute criteria	5.18737751763962	9.78760603604435	12.0901461298633
no. of iteration - relative criteria	100	10,000	100,000
final guess - relative criteria	5.18737751763962	9.78760603604435	12.0901461298633

Table 3: Convergence for y_k

approaches its true limit 1. Table 3 also shows that both absolute and relative criteria lead to the same number of iterations for the sequence y_k. Since y_k is a divergent sequence, the final guess will be higher (with no upper bound), the more we iterate. By lowering tolerance distance δ, the algorithm needs to iterate longer in order to satisfy the stopping rule, which in turn results in higher final guess. Since there is no upper bound for y_k, we can always make the tolerance distance δ even lower, and obtain even higher final guess for y_k.