CSC/MAT 335 - EXAM 1 PRACTICE-TEST
Dr. A.J. Pounds
Fall 2013

Name ____ **Section** ____

Show all work and draw a box around numerical answers.

1. For the following calculation:

$$\frac{(12.65)(2.588)}{(13.01)} - 2.516$$

   (a) compute the result using four digit rouding.

   (b) compute the result using four digit chopping.

   (c) is there any loss of precision in the final result? If so, how many digits are lost? If not, why not?

2. The Maclaurin series expansion of $\ln(x+1)$ truncated to second order with its associated error term is given in Eqn. 1.

$$M_2(x) = \underbrace{x - \frac{x^2}{2}}_{\text{expansion}} + \underbrace{\frac{x^3}{3(\xi(x)+1)^3}}_{\text{error term}} \tag{1}$$

(a) What is the value of $M_2(.25)$?

(b) What is the error bound in $M_2(.25)$?

(c) What is the absolute error in $M_2(.25)$ compared to $\ln(0.25+1)$?

3. You want to determine the **negative** root for the function

$$f(x) = x^3 + 2x + 10$$

using the bisection method.

(a) Identify two values that you know bound the root.

(b) Using the bounds from part (a), carry out four iterations of the bisection algorithm to determine a reduced domain in which the root exists.

(c) Restarting from one of your bounds in part (a), carry out four iterations of Newton's method to find an approximation to the root.

(d) For your equation

$$f(x) = x^3 + 2x + 10$$

i. recast the equation in the form $g(x)$ so it can be solved by the fixed point (or Picard's) method.

ii. write a function code segment (or a subroutine) that solves for the root using the fixed point (or Picard's) method to 10 times *IEEE* single precision (32 bit) machine tolerance.

4. *NVIDIA* introduced a half precision data type (16 bits total with 1 sign bit, 5 bits in the exponent, and 10 bits in the mantissa). This later became the IEEE 754 half-precision binary floating point format known as **binary16**. What is the machine number for π in this format? Please note that the most significant and least significant bits are noted with an "m" and "l" respectively.

1	10001	1000000001
	m l	m l