MAT360 Exam 1 (Spring 2015)

Name:

Directions: Problems are not equally weighted. Show your work! Answers without justification will likely result in few points. Your written work also allows me the option of giving you partial credit in the event of an incorrect final answer (but good reasoning). Indicate clearly your answer to each problem (e.g., put a box around it). You must skip 20 points (your choice). Write "skip" clearly on those parts you skip. **Good luck!**

Problem 1. (20 pts)

Compute the following, using both three-digit round-to-even and three-digit chopping. Assume that each constant and the result of each arithmetic operation must be converted to a machine number using the appropriate float (fl) conversion.

$$\frac{\frac{34}{21} - \frac{8}{5}}{\pi - e} \approx 0.0449968$$

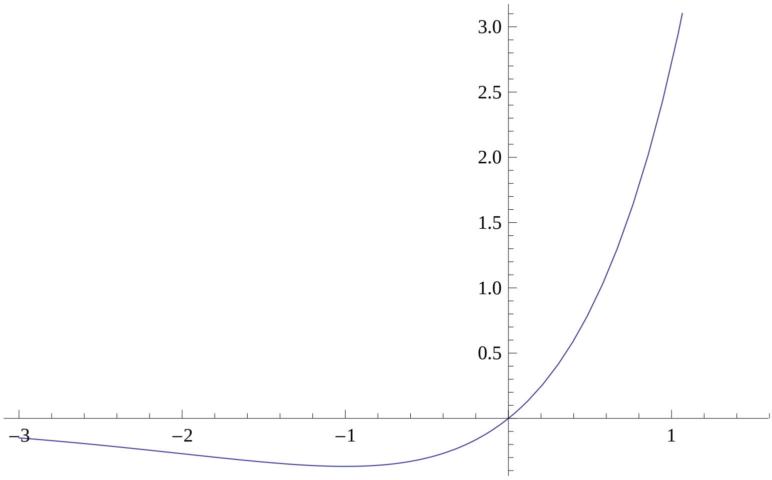
Compute absolute and relative errors and give your final answer in the following table (show your work for each below the table):

method	Approximation	Absolute error	Relative Error
3-digit chopping			
3-digit rounding			

- (10 pts) chopping:

- (10 pts) rounding:

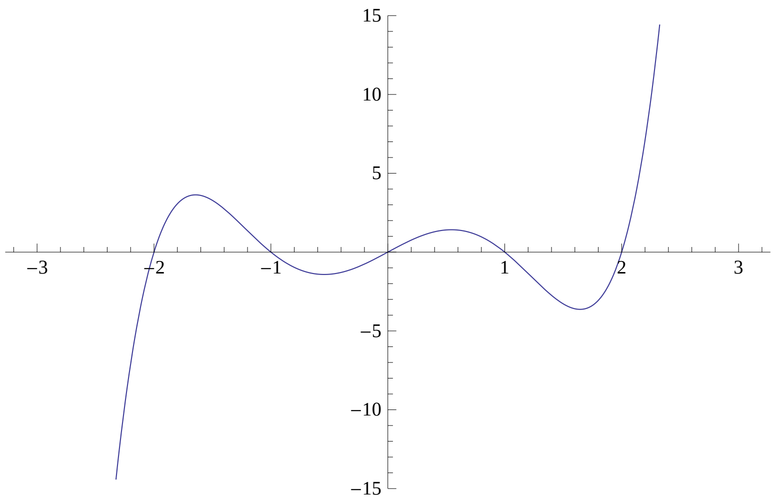
Problem 2. (20 pts) Consider the following graph of the function $f(x) = xe^x$:



a. (10 pts) Assume that $x_0 = 1$ is the first approximation to the root r. Demonstrate graphically the means by which Newton's method obtains x_1, the next approximation, and find the **exact value** of x_1.

b. (10 pts) Describe the ways in which Newton's method could fail to converge to a root in this case. Describe (or show) particular starting approximations x_0 which would lead to failure. The more you can say, the merrier.

Problem 3. (10 pts) To which root of $f(x) = (x+2)(x+1)x(x-1)(x-2)$ will bisection converge, if we start with the interval $[-4, 2.6]$? How many iterations of bisection must you see before you know that? (Justify your answer! Strong hint: do you **need** to compute f(x)?)



Problem 4. (15 pts) Recall the toy computer

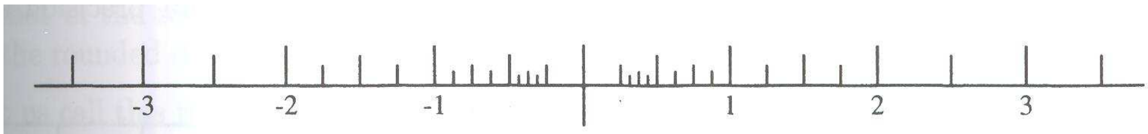


Figure 2.3: Machine numbers on a toy binary computer

a. (5 pts) Give the base, precision, and the exponent range of the computer.

b. (10 pts) Compute the product $1.2 \cdot 0.43$ using this toy computer, assuming rounding-to-even.

Problem 5: (20 pts) Short answer (5 points each):

a. Why does the author advise using $4 \cdot \tan^{-1}(1)$ as an input to software, rather than any decimal approximation to π?

b. Write 2592_{10} in base 2, and 10011011_2 in base 10.

c. For which values of x do tiny relative changes in x result in large relative changes in $e^{\frac{1}{x}}$? (You need to show **some** work.)

d. Suppose we want to compute e^a, but a is given with error: $\hat{a} = a + \epsilon_a$. Show that the propogated **relative** error in the calculation of e^a is a function only of the **absolute** error ϵ_a in a.

Problem 6: (15 pts) Consider the bivision algorithm developed in class, to replace machine division using only the other three basic arithmetic operations (*, +, -):

```
Bivision[a0_, b0_] :=
 Module[{},
  a = N[a0];
  b = N[b0];
  (* part 1 *)
  If[b < 0, a = -a; b = -b];
  (* part 2 *)
  p = 1;
  tmp = b;
  While[tmp < 1, tmp = tmp*10; p = p*10];
  (* part 3 *)
  If[b < 1,
   Return[a*BisectionDivision[b, 1, p]],
   Return[a*BisectionDivision[b, 0, 1]]
   ]
  ]
```

a. (5 pts) Describe the point of each of the three labelled parts of this algorithm, and why they're useful or necessary.

b. (10 pts) I'm interested in the calculation of p. In class it was suggested that we determine the value of p using logs, but I said that we wanted to use only the three other basic arithmetic operations. How could we calculate p using base-10 logs and the floor function?

Problem 7: (10 pts) Consider Example 2.7, p. 67:

"The calculation of $x - \pi$ is typical of what must be done to compute trigonometric functions. Assume the use of four-decimal-digit numbers. If you use $x - 3.142$, you will get a completely wrong answer for $x = 3.142$ and poor accuracy for nearby values. However, $(x - 3.141) - 0.0005927$ produces results correctly rounded to four significant digits for all x."

a. Explain the significance (and even the cleverness) of this trick.

b. π is given by 3.141592653589793 in my favorite software package. How could I implement the scheme of Example 2.7, given that 3.1415926535897932384626433832795028841971693993751 is a better approximation to π?