Homework 2

Section 1.2

# Problem 1.

Compute the absolute error and relative error in approximation of by .

Absolute error:

Relative error:

# Problem 13.

Use four-digit rounding arithmetic and the formulas (1.1), (1.2), and (1.3) to find the most accurate approximation to the roots of the following quadratic equations. Compute he absolute errors and relative errors.

Doing the 1.1 method first, solve the following:

Equation 1.1:

Now solve for the discriminant:

Using the method of 1.2 and 1.3:

Equation 1.2:

Equation 1.3:

Solving with computer, we get the roots to be:

Using the closest value:

# Problem 15.

Using the 64-bit long real format to find the decimal equivalent of the following floating point machine number:

Sign:

Exponent:

Mantissa:

Section 1.3

# Problem 1

1. Use three-digit chopping arithmetic to compute the sum first by and then by . Which method is more accurate, and why?

|  |  |  |  |
| --- | --- | --- | --- |
| Fraction | Decimal | Adding Down | Adding Up |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Using Wolfram Alpha, we get that the correct answer is:

The reason why adding the numbers from smallest to biggest was more accurate than adding from biggest to smallest was because when we started from 1, as we went down and got smaller numbers, their accuracy was decreased since the large number took all the digits for itself. Going down to up, the small numbers were able to keep more of their digits through the process.

# Problem 7.

Find the rates of convergence of the following function as .

Section 2.1

# Problem 5.

Use the Bisection method to find solutions accurate within for the following problem.

Using program:

#include <iostream>

#include <cmath>

using namespace std;

//This is where the function is at.

long double function(long double x)

{

long double fx = 0;

fx = exp(x) - x \* x + 3 \* x - 2;

return fx;

}

//This is where the main program is at.

int main()

{

//The starting points are here where it goes from 0 to 1.

long double a = 0, b = 1, xn = 0, tol = 0;

tol = 1 / pow(10, 3);

bool flag1 = false;

//This is just to know how many iterations had to be done.

int steps = 0;

//This do loop will perform the bisection method until we are within our tolerance.

do

{

//This is where the half way value is at.

xn = (a + b) / 2;

//This is where our values will be tested and adjusted as needed.

if (function(xn) < 0)

{

a = xn;

}

else if (function(xn) > 0)

{

b = xn;

}

else if (function(xn) == 0)

{

cout << "The value x = " << xn << " is a root to the polynomial e^x - x^2 + 3\*x - 2 = 0." << endl;

flag1 = true;

}

else

{

cout << "There is a problem" << endl;

flag1 = true;

}

long double c = 0;

c = b - a;

if (c <= (tol/2))

{

continue;

}

//This is the step counter.

steps = steps + 1;

}while (flag1 == false);

//This is where the number of steps is outputted.

cout << "The number of steps it took was: " << steps << endl;

return 0;

}

Doing this we get and the number of steps was 10.

# Problem 15.

Use Theorem 2.1 to find a bound for the number of iterations needed to achieve an approximation with accuracy to the solution of lying in the interval . Find an approximation to the root with this degree of accuracy.

Theorem 2.1:

Suppose that and . The Bisection method generated a sequence approximating a zero of with