Ryan Valby 40848407 MEEN 2450

Exercise 1: 11101101

$(-)\ 2^6\ 2^5\ 0\ 2^3\ 2^2\ 0\ 2^0$

$-\left[2^6+2^5+2^3+2^2+2^0\right] = \boxed{-109}$

Exercise 2:

		R
/2	420	
/2	210	0
/2	105	0
/2	52	1
/2	26	0
/2	13	0
/2	6	1
/2	3	0
/2	1	1
	0	1

$420 = \boxed{110100100}$

$\boxed{9 \text{ bits required}}$

Exercise 3

$$f(x) = \frac{5x}{(1-2x^2)^2}$$

a) $f(.423) = \dfrac{5(.423)}{(1-2(.423)^2)^2} = \dfrac{2.11}{(1-2(.178))^2} = \dfrac{2.11}{(1-.356)^2}$

$= \dfrac{2.11}{(.644)^2} = \dfrac{2.11}{.414} = \boxed{5.09}$ w/ 3 digit chopping

$E_T = T - A = 5.129\ldots - 5.09 = .03918\ldots$

$\varepsilon_T = \dfrac{E_T}{T} \times 100 = \dfrac{.03918\ldots}{5.129\ldots} \times 100 = \boxed{0.764\%}$

b) $f(.423) = \dfrac{5(.423)}{(1-2(.423)^2)^2} = \dfrac{2.115}{[1-2(.1789)]^2} = \dfrac{2.115}{[1-.3578]^2} = \dfrac{2.115}{(.6422)^2}$

$= \dfrac{2.115}{.4124} = \boxed{5.128}$

$E_T = T - A = 5.129\ldots - 5.128 = .00118\ldots$

$\varepsilon_T = \dfrac{E_T}{T} \times 100 = \dfrac{.00118}{5.129\ldots} \times 100 = \boxed{.0231\%}$

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 24 01:17:19 2019
Exercise 1 and 2 extra
@author: Ryan Dalby
"""
def toDecimal(num):
    '''Takes binary number in signed magnitude form as a string
    (first bit before number must be 0 or 1 to specify sign)
    and gives decimal form'''
    ans = int(num[1:], 2)
    if int(num[0]) == 1:
        return -1 * ans
    else:
        return ans

def toBinary(num):
    '''Takes decimal number and gives binary number in signed
    magnitude form'''
    if num < 0:
        return '1' + (bin(abs(num))[2:])
    else:
        return bin(num)[2:]


print(toDecimal('0110100100'))
print(toDecimal('11101101'))

print(toBinary(420))
print(toBinary(-109))
```

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 22 17:45:15 2019
HW1b Exercise 4
@author: Ryan Dalby
"""
import math

def form1(x, numTerm):
    '''Enter x value to evaluate at and number of terms to approximate to
    using formula 1, gives a matrix that for each term gives:
    [current term number, resulting value of approximation that iteration, true relative e
    actualValue = (math.e)**(-1 * x) #actual value that formula approximates
    currentResult = 0 #current result of formula corresponds to current result of sum in f
    lastResult = 0 #previous result(previous iteration) from formula
    matrix = [] #matrix will hold information about each approximation(with a given amount
    for n in range(numTerm):
        currentResult += ((-1)**n)* (x**n)/(math.factorial(n)) #formula
        Et = getEt(actualValue, currentResult)
        Ea = ((currentResult - lastResult)/currentResult) * 100
        lastResult = currentResult
        matrix.append([(n+1), currentResult, Et, Ea]) #add information to matrix about sum

    return matrix


def form2(x, numTerm):
    '''Enter x value to evaluate at and number of terms to approximate to
    using formula 2, gives a matrix that for each term gives:
    [current term number, resulting value of approximation that iteration, true relative e
    actualValue = (math.e)**(-1 * x) #actual value that formula approximates
    sumResult = 0 #current result of sum in formula
    lastResult = 0 #previous result(previous iteration) from formula
    matrix = [] #matrix will hold information about each approximation(with a given amount
    for n in range(numTerm):
        sumResult += (x**n)/(math.factorial(n)) #sum formula
        currentResult = 1 / sumResult #current reuslt of formula is 1/sumResult
        Et = getEt(actualValue, currentResult)
        Ea = ((currentResult - lastResult)/currentResult) * 100
        lastResult = currentResult
        matrix.append([(n+1), currentResult, Et, Ea]) #add information to matrix about sum

    return matrix

def getEt(T, A):
    '''Given T true value and A approximate value gives true relative error'''
    return ((T - A) / T) * 100

def printByRow(m):
    '''Prints by row for a matrix'''
    for i in m:
```

```
        print(i)

m1 = form1(5, 20)
m2 = form2(5, 20)
printByRow(m1)
printByRow(m2)
```

	Formula 1			Formula 2		
terms	value	εt (%)	εa (%)	value	εt (%)	εa (%)
1	1	-14741.31591	N/A	1	-14741.31591	N/A
2	-4	59465.26364	125	0.166666667	-2373.552652	-500
3	8.5	-126051.1852	147.0588235	0.054054054	-702.2332924	-208.3333333
4	-12.33333333	183142.8962	168.9189189	0.025423729	-277.3215909	-112.6126126
5	13.70833333	-203349.7056	189.9696049	0.015296367	-127.0182166	-66.20762712
6	-12.33333333	183142.8962	211.1486486	0.010938924	-62.34803184	-39.83428936
7	9.368055556	-138934.272	231.6530764	0.008840322	-31.20200694	-23.73898511
8	-6.132936508	91120.84817	252.7499191	0.007774898	-15.38972015	-13.70337563
9	3.555183532	-52663.60191	272.506889	0.007230283	-7.306918083	-7.532414692
10	-1.827105379	27216.64813	294.5801032	0.006959453	-3.287438512	-3.891547345
11	0.864039076	-12723.47689	311.4609662	0.006831506	-1.388543334	-1.872889299
12	-0.359208403	5431.125394	340.5397724	0.006774891	-0.548299117	-0.835662288
13	0.150478046	-2133.292224	338.7115011	0.006751577	-0.202293563	-0.345307019
14	-0.045555204	776.0991671	430.3202153	0.006742653	-0.069847751	-0.132353367
15	0.024456671	-262.969187	286.2690254	0.006739472	-0.022630488	-0.04720658
16	0.00111938	83.38693102	-2084.841268	0.006738412	-0.0069013	-0.015728102
17	0.008412283	-24.84935587	86.69350846	0.006738081	-0.001986944	-0.004914259
18	0.006267312	6.984846157	-34.22474802	0.006737983	-0.000541637	-0.001445299
19	0.006863137	-1.857987739	8.681532094	0.006737956	-0.00014017	-0.000401466
20	0.006706341	0.469073812	-2.338028632	0.006737949	-3.45E-05	-0.000105649

ME EN 2450
Ryan Dalby u0848407
HW1b

Exercise 4:
It appears that formula 2 is a better approximation than formula 1 because both the true relative error and the approximate relative error is smaller than formula 1's errors when we go out to 20 summation terms. It can also be noted that formula 2 avoids alternating around the solution like formula 1 does which gives more difficult to interpret results before getting close to the solution.