Lab 1 -- MTH 351 -- John Waczak

Table of Contents

.... 1
1) Consider the Taylor expansion for ln(1-x), ln((1+x)/(1-x)) 1
2.) 3


```
clear all;
format long e;
```

1) Consider the Taylor expansion for ln(1-x), ln((1+x)/(1-x))

see function files for calculation of taylor expansions and relative error. taylor1 corresponds to the first function and taylor2 to the second. a) To get ln(1.9) an x value of -0.9 must be used. b) Write a script in Matlab to demonstrate how many terms are necessary to achieve ten digits of accuracy.

```
xT1 = log(1.9);
xA1 = taylor1(-0.9,1);
n1 = 1;
while relative(xT1, xA1) > (5*10^(-10-1))
    n1 = n1 + 1 ;
    xA1 = taylor1(-0.9, n1) ;
end
n1
xT1
xA1
```

```
n1 =

   174


xT1 =

     6.418538861723947e-01


xA1 =

     6.418538861427578e-01
```

c) Do the same as (a) for series two.

```
val = (1.9-1)/(1+1.9)
```

```
% ^ that is the value for x to get ln(1.9)


val =

     3.103448275862069e-01
```

d) Do the same as (b) for series two

```
xT2 = log(1.9);
xA2 = taylor2(val, 1);
n2 = 1;
while relative(xT2, xA2)>(5*10^(-10-1))
    n2 = n2 + 1;
    xA2 = taylor2(val, n2);
end
n2
xT2
xA2


n2 =

     9


xT2 =

     6.418538861723947e-01


xA2 =

     6.418538861468703e-01
```

e) which is more efficient? we can measure the time to compute using the tic,toc commands however it would appear 2 should be faster as it took 9 iterations versus 174.

```
tic
    taylor1(-0.9, n1)
toc

tic
    taylor2(val, n2)
toc


ans =

     6.418538861427578e-01

Elapsed time is 0.001170 seconds.
```

```
ans =

     6.418538861468703e-01

Elapsed time is 0.000383 seconds.
```

This timing command reflects my suspicion that (2) is the more efficient method.

2.)

Write a script in Matlab to create a table of values (similar to Table 2.7) obtained by evaluating a given function as it is written, and also as a reformulation designed to eliminate loss-of-significance errors. Choose x from 10^{-1} to 10^{-20} decreasing by a factor of 0.1. a) see f1 and fixed_f1 function files. This function was fixed by multiplying top and bottom by the conjugate.

```
fprintf('\n\nTable:\n\n');
fprintf('x \t f(x) \t fixed f(x) \n');
for i=1:20
    x=10^(-i);
    fprintf('%g \t %0.15f \t %0.15f\n',x,f1(x), fixed_f1(x))
end
```

```
Table:

x    f(x)    fixed f(x)
0.1    0.248456731316584    0.248456731316587
0.01    0.249843945007866    0.249843945007857
0.001    0.249984376953005    0.249984376952820
0.0001    0.249998437520382    0.249998437519531
1e-05    0.249999843759952    0.249999843750195
1e-06    0.249999984269778    0.249999984375002
1e-07    0.249999998480632    0.249999998437500
1e-08    0.249999976276172    0.249999999843750
1e-09    0.250000020685093    0.249999999984375
1e-10    0.250000020685093    0.249999999998437
1e-11    0.249977816224600    0.249999999999844
1e-12    0.250022225145585    0.249999999999984
1e-13    0.248689957516035    0.249999999999998
1e-14    0.222044604925031    0.250000000000000
1e-15    0.000000000000000    0.250000000000000
1e-16    0.000000000000000    0.250000000000000
1e-17    0.000000000000000    0.250000000000000
1e-18    0.000000000000000    0.250000000000000
1e-19    0.000000000000000    0.250000000000000
1e-20    0.000000000000000    0.250000000000000
```

b) see f2 and fixed_f2 function files. This function was fixed by expanding using a Taylor polynomial for exp(-x)

```
fprintf('\n\nTable:\n\n');
fprintf('x \t f(x) \t fixed f(x) \n');
for i=1:20
```

```
    x=10^(-i);
    fprintf('%g \t %0.15f \t %0.15f\n',x,f2(x), fixed_f2(x))
end
```

```
Table:

x   f(x)   fixed f(x)
0.1   0.951625819640405   0.951666666666667
0.01   0.995016625083189   0.995016666666667
0.001   0.999500166624978   0.999500166666667
0.0001   0.999950001666638   0.999950001666667
1e-05   0.999995000017240   0.999995000016667
1e-06   0.999999499984305   0.999999500000167
1e-07   0.999999949513608   0.999999950000002
1e-08   0.999999993922529   0.999999995000000
1e-09   0.999999971718068   0.999999999500000
1e-10   1.000000082740371   0.999999999950000
1e-11   1.000000082740371   0.999999999995000
1e-12   0.999977878279878   0.999999999999500
1e-13   1.000310945187266   0.999999999999950
1e-14   0.999200722162641   0.999999999999995
1e-15   0.999200722162641   0.999999999999999
1e-16   1.110223024625157   1.000000000000000
1e-17   0.000000000000000   1.000000000000000
1e-18   0.000000000000000   1.000000000000000
1e-19   0.000000000000000   1.000000000000000
1e-20   0.000000000000000   1.000000000000000
```

This loss of significance error is occuring whenever we subtract two numbers that are nearly the same. When we fix the functions by rewriting them to remove the subtraction, we get rid of the loss of significance error.

Published with MATLAB® R2017b