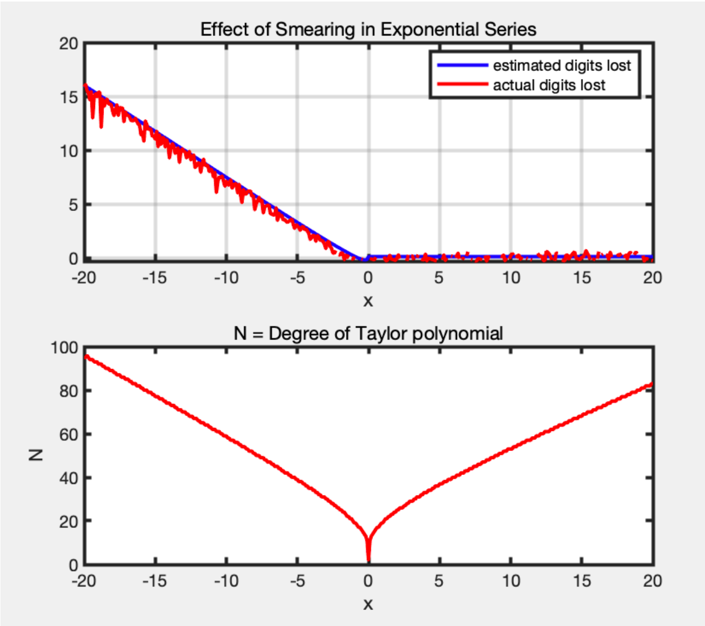
Report of MACM 316 Computing Assignment #1

Delin Chen 301376176

C1:



C2:

n	y_n
0	0.09531018
1	0.046898202
2	0.03101798
3	0.023153529
4	0.01846471
5	0.015352901
6	0.013137658
7	0.011480561
8	0.010194391
9	0.009167203
10	0.008327966
11	0.007629436
12	0.007038976
13	0.006533316
14	0.006095415
15	0.005712514
16	0.005374864
17	0.005074893
18	0.004806628
19	0.004565296
20	0.004347036

Reason why numbers differ from those in Table 1.1 is that the error in y_n at each step is damped out in the next step, so our numbers are getting more and more accurate than the numbers in Table 1.1.

C3:

Exact value s = 1.07667404746858

If we use the series directly, then we need 3161787 terms. However, if we compute s by alternative term provided, it only requires 147 terms.

Code for C1:

```
%EXPSERIESERROR
%
%Script to compute the estimated number of (decimal) digits lost, and the
%  actual number of digits lost when computing exp(x) via its
%  Mclaurin series
%
%  Plot results
%  Uses Matlab function "expseries"
%
x = [-20:.1:20]; % test points <---
exactv = exp(x); % exact values
ys=[]; edl=[]; nn=[];  % vectors to store outputs
for j=1:length(x),
    [y,edloss,k]=expseries(x(j));
    ys=[ys y]; edl=[edl edloss]; nn=[nn k];
end
%Find the coding bug in the 2 lines below <---
relerr=abs((ys-exactv)./exactv); %relative error
digitslost = (log(relerr/eps)/log(10));
subplot(2,1,1);
plot(x,edl,'b',x,digitslost,'r');
title('Effect of Smearing in Exponential Series');
xlabel('x');
legend('estimated digits lost','actual digits lost');
grid on;
subplot(2,1,2);
%Plot of "nn" goes here  <---
plot(x,nn,'r');
xlabel('x');
ylabel('N');
title('N = Degree of Taylor polynomial');

plotpubl(2);
```

Code for C2:

```
clc
clear
for n = 0:20
fun = @(x,n) (x.^n)./(x+10);
q(n+1) = integral(@(x) fun(x,n),0,1);
fprintf('%.9f\n',q(n+1));
end
```

Code for C3:

Using original series:

```
clc
clear
sum = 0;
sum0 = 0;
n = 1;
while true
    sum = sum0 + (1/(n^2+1));
    error=abs(sum-sum0);
    if error<10^-13
        break
    end
    n = n+1;
    sum0 = sum;
end
fprintf('Sum is %.14f\nThere are %d terms needed.\n',sum,n);
```

Using alternative series:

```
clc
clear
sum = 0;
sum0 = 0;
n = 1;
while true
    sum = sum0 + 1/(n^2+1)-1/n^2+1/n^4;
    if abs(sum-sum0)<10^-13
        break
    end
    n = n+1;
    sum0 = sum;
end
sum = sum+ pi.^2/6 - pi.^4/90;
fprintf('Sum is %.14f\nThere are %d terms needed.\n',sum,n);
```