1 Problems

1. The mean value theorem states that given any continuous, differentiable function \( f \) on a closed interval \([a, b]\) that there exists a point \( c \) in \([a, b]\) such that the derivative of \( f \) at \( c \) is equal to the average rate of change from \( a \) to \( b \), i.e. that:

\[
 f'(c) = \frac{f(b) - f(a)}{b - a}.
\]

To find such a point for \( f = \frac{1}{x+1} \) on the interval \([0, 1]\) note \( f(1) = \frac{1}{2} \) and \( f(0) = 1 \) so \( \frac{f(b) - f(a)}{b - a} = -\frac{1}{2} \). Then note \( f'(x) = -\frac{1}{(x+1)^2} \). Solving this equation for \( f'(x) = -\frac{1}{2} \) yields \( x = \pm \sqrt{2} - 1 \), only one of which is in \([0, 1]\) hence our answer is \( \sqrt{2} - 1 \).

2. To find the second degree Taylor Polynomial for \( e^{x^2} \) at \( x = 0 \) we must evaluate the sum:

\[
 \sum_{n=1}^{2} \frac{f^n(0)}{n!} x^n
\]

First note \( f(0) = 1 \), \( f'(x)|_{x=0} = 2e^{x^2} \) \( x=0 \) = 0 and \( f''(x)|_{x=0} = 4x^2e^{x^2} + 2e^{x^2} |_{x=0} = 2 \). Then the second degree Taylor Polynomial can be written as:

\[
 1 + 0(x) + \frac{2}{2!}(x^2) = 1 + x^2.
\]

3. (a) We can find the forth degree Taylor approximation for \( \ln(x) \) around \( x = 1 \) the same way we found the Taylor Polynomial in the previous problem. Note
\( f'(x) = \frac{1}{x}, f''(x) = -\frac{1}{x^2}, f'''(x) = \frac{2}{x^3}, \) and \( f^{iv} = -\frac{6}{x^4}. \) The 4th degree Taylor approximation can be then written as:

\[
\hat{f}(x) = (x - 1) - \frac{1}{2}(x - 1)^2 + \frac{1}{3}(x - 1)^3 - \frac{1}{4}(x - 1)^4
\]

(b) We can now use this approximation to estimate \( \ln(.9) \approx \hat{f}(.9) = .1054 \) and \( \ln(1.1) \approx \hat{f}(1.1) = .0953 \)

(c) We can use the next term in the Taylor expansion for \( \ln(x) \) to estimate the errors in the approximations found in part (b). Then we define \( \text{errorestimate}(x) = \frac{1}{5}(x - 1)^5. \) Then evaluating we find \( \text{errorestimate}(.9) = -2.0 \times 10^{-6} \) and \( \text{errorestimate}(1.1) = 2 \times 10^{-6}. \)

(d) Checking our approximation with a calculator we find:

\[
\hat{f}(.9) = \ln(.9) = 2.183 \times 10^{-6}
\]
\[
\hat{f}(1.1) = \ln(1.1) = 1.847 \times 10^{-6}
\]

4. Plots of \( \text{mynsin.m}, n(x) \) and errors. Note the differences in the scale of the relative errors as we change the number of points, this is happens because for \( x_i \) close to \( \pi \) or \( 2\pi, \sin(x_i) \approx 0 \) so we end up dividing by a number very close to zero when we calculate the relative error.
2 Codes

Code for 4

```matlab
%PS1N4

% generate data
numpts = 1001;
x = linspace(0,2*pi,numpts);
x1 = linspace(0,2*pi,numpts-1);
y = sin(x);
y1 = sin(x1);
myy1 = x(1:end-1);
its1 = myy1;
abserror1 = myy1;
relerror1 = myy1;

myy2 = x;
its2 = x;
abserror2 = x;
relerror2 = x;

for i=1:length(x)-1
    [myy1(i),its1(i)] = mysin(x1(i));
```

```
abserror1(i) = abs((myy1(i)-y1(i)));
relerror1(i) = abs((myy1(i)-y1(i))/y1(i));
end

for i=1:length(x)
    [myy2(i),its2(i)] = mysin(x(i));
    abserror2(i) = abs((myy2(i)-y(i)));
    relerror2(i) = abs((myy2(i)-y(i))/y(i));
end

%plots
%
figure
suptitle(['Numpts = ' num2str(numpts-1)])
subplot(2,2,1)
plot(x1,myy1)
xlabel('x')
ylabel('y')
title('mysin.m')

subplot(2,2,2)
plot(x1,its1)
xlabel('x')
ylabel('n(x)')
title('n(x) vs x')

subplot(2,2,3)
plot(x1,abserror1)
xlabel('x')
ylabel('error')
title('Absolute Error')

subplot(2,2,4)
plot(x1(1:end-1),relerror1(1:end-1))
xlabel('x')
ylabel('error')
title('Relative Error')

%2
figure
suptitle(['Numpts = ' num2str(numpts)])
subplot(2,2,1)
plot(x,myy2)
xlabel('x')
ylabel('y')
title('mysin.m')

subplot(2,2,2)
plot(x,its2)
xlabel('x')
ylabel('n(x)')
title('n(x) vs x')

subplot(2,2,3)
plot(x,abserror2)
xlabel('x')
ylabel('error')
title('Absolute Error')

subplot(2,2,4)
plot(x(1:end-1),relerror2(1:end-1))
xlabel('x')
ylabel('error')
title('Relative Error')

function [ out , nTerms] = mysin(x)

% allocate space
out = 0;
term = 1;
nMax = 100;%set maximum number of iterations

%loop to evaluate taylor epansion
for its = 0:nMax
    term = ((-1).^its.*x.^(2*its+1))/factorial(2*its+1);
    if( abs(term) < 1.e-10 )
        break %exit for loop when next term is smaller than 10^-10
    end
    out = out + term;
end
nTerms = its+1;