ChEE 201
Fall 2005
University of Arizona
Computer Homework 5
Taylor Series Approximations

Students need to have read Computer Reading 4 and 5 along with having completed Computer HW 4 before attempting this homework.

At the end of this section, students will be able to:
1) use a computer program with a Taylor series approximation to estimate a function at a point
2) apply the approximate relative error can be used to decide when they have included enough terms in their infinite series to have come close enough to a converged answer to stop
3) understand how double and single precision can affect answers
4) compute a Taylor series expansion and approximate a function that has more than one variable

1) In the previous homework, you found an infinite series representation for ln(x) where you expanded about the point $x_i = 1$ because you know that ln(1) is zero. You then found how many terms it would take to find ln(2) using an approximate relative error of 0.01. Now, use your infinite sum equation to find how many terms you would need to have in your series to find the ln of 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, and 2.0 with an approximate relative error of 0.01. To do this, write a short VBA program that takes in xi+1 as input and reports the number of terms that you need to reach convergence. Make a table that has x_{i+1} and the number of terms it takes to get convergence for each value of x_{i+1}. For full credit, submit a hard copy of your program and your table of the number of steps it takes for each value.

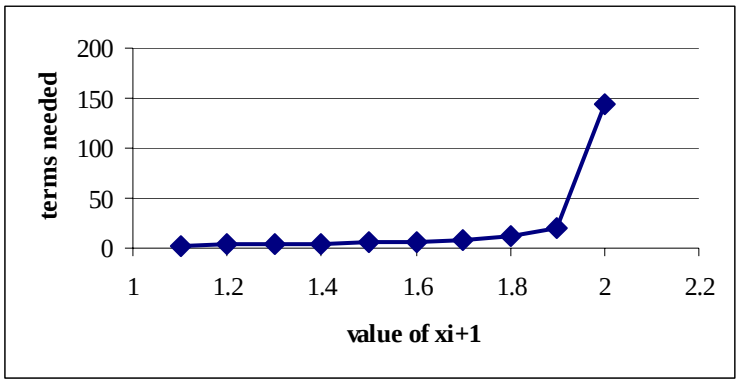
Solution: We could write a program that looks like this:

```
Option Explicit
Function logapp(n)
Dim ae As Single, b As Single, a As Single
Dim sum As Single, i As Single, term As Single
Dim stopping As Single

stopping = 10000
sum = 0
For i = 1 To stopping
   term = ((-1) ^ (i + 1)) * ((n - 1) ^ i) / i
   a = sum + term
   ae = (a - sum) / a

   sum = a
   b = i
   If Abs(ae) < 0.01 Then i = stopping
Next i

MsgBox (ae)
logapp = b
End Function
```

Notice how our programs are getting longer and a little more complex? You're building up your skills in programming and are becoming more able to accomplish tasks.

Our table of results using this program is (with a plot off to the side (students do not need to show plot):

x_{i+1}	Number of Terms
1.1	3
1.2	4
1.3	4
1.4	5
1.5	6
1.6	7
1.7	9
1.8	12
1.9	20
2	145



2) If you tried to use your program to find the ln(2.9), though, it can't with this convergence criteria. Try it with 10000 steps just to prove to yourself that this doesn't converge. The problem is that the power on the $(x_{i+1} - x_i)$ term starts to dominate over the other terms until your function explodes; you're taking 1.01^n and this grows too quickly to be reigned in by dividing by n. Your approximation will diverge for any value of x_{i+1} that is greater than 1.

Let's imagine you wanted to approximate ln(2.9). How could you do this and have your program still converge? For this problem, if x_{i+1} is greater than 2 but less than three, have the computer first approximate the ln(2) with the program you have above and then use that approximation in your new approximation to get to 2.9, rewriting your derivation for the second infinite series. In essense, you'll be doing a Taylor series approximation to get from 1 to 2 and then doing another one from 2 to 3. Use a convergence criteria of 0.0001 on the absolute relative error for the second approximation and 0.01 on the first one. Submit a hardcopy of your program and your approximate answer to ln(3). Email a copy of your Excel file saved as First Name Last Name 5.2.xls to blowers@engr.arizona.edu before 10 am on the day the program is due.

Solution: Our program will be a bit longer this time. We will start adding comments so the program is easier to understand and pick apart. First, though, notice that we have to rederive our infinite series here because we are no longer using our starting point (x_i) as 1. Here is what we get for each term before we reduce it:

$$\ln(2.9) \approx \ln(2)_{appr} + \frac{1}{x}\bigg|_2 \frac{1}{1!}(2.9-2.0)^1 - \frac{1}{x^2}\bigg|_2 \frac{1}{2!}(2.9-2.0)^2 + \cdots$$

The infinite series becomes:

$$\ln(2.9) \approx \ln(2)_{appr} + \sum_n (-1)^{n+1} \frac{1}{n!}\left(\frac{1}{2}\right)^n (0.9)^n$$

This infinite series would go into our second loop for adding the successive approximation to our answer. With all this done, our code could look like:

```
Option Explicit
Function logapp(n)
'This function will approximate the ln of a number between 2 and 3.  Specifically,
'it is written to approximate ln(2.9)

Dim ae As Single, b As Single, a As Single
Dim sum1 As Single, i As Single, term As Single
Dim stopping As Single, sum2 As Single
Dim factory As Single, j As Single

'ae is our absolute error that will be used as our stopping criteria
'  if ae<0.01 we will stop
'b will be the number of terms needed to reach convergence
'a will be our new value of our previous value + the newest term added
```

```
'sum1 is the first summing variable to get to ln(2)
'i is the first looping variable to approximate ln(2) first
'term is the newest term computed in the loop that is used to find the new sum1 or sum2
'stopping is the maximum number of steps the program can take to avoid infinite loops
'factory will compute our factorial that we need in our revised loop for the successive approximation
'j is the counting variable for that factory loop
'sum2 is the second summing variable for the second approximation

'decision loop for when to do this approximation twice
If n > 2 And n < 3 Then

   'first loop to get find the approximate value of ln(2).
   'This will be our starting point for the next loop
   stopping = 10000
   sum1 = 0
   For i = 1 To stopping
      term = ((-1) ^ (i + 1)) * ((2 - 1) ^ i) / i
      a = sum1 + term
      ae = (a - sum1) / a

      sum1 = a
      b = i
      ' the next line checks our stopping criteria for this loop
      If Abs(ae) < 0.01 Then i = stopping
   Next i

   stopping = 10000
   'the next line sets our starting ln(2)appr value
   sum2 = sum1
   For i = 1 To stopping
      'the next four lines calculate our factorial function
      factory = 1
      For j = 1 To i
         factory = factory * j
      Next j

      term = ((-1) ^ (i + 1)) * (0.5 ^ i) * (1 / factory) * (0.9 ^ i)
      'msgbox reports out our terms to check our infinite sum and calcs so far - doesn't need to be here
      MsgBox (term)
      a = sum2 + term
      ae = (a - sum2) / a

      sum2 = a
      b = i
      If Abs(ae) < 0.0001 Then i = stopping
   Next i
   'the next msgbox reports out our final answer
   MsgBox (sum2)

Else
'return a variable that we know is not right if the condition above was not true
   b = 0
End If

'send our answer back to Excel
logapp = b
```

End Function
Whew!

3) This problem is going to be very, very short, but will address objective three above.
The infinite series:

$$f(N)=\sum_{n=1}^{N}\frac{1}{n^2}$$

converges on a value of $f(N)=\pi^2/6$ as N approaches infinity. Write a program to compute f(N) for N = 10000 by computing the sum from n = 1 to 10000. Then repeat the computation but in reverse order, that is from n = 10000 to 1 using increments of -1. Use single precision for all numbers.

Explain the difference between your results. Which number do you feel is more accurate?

Solution: Our programs would look like

Option Explicit *Function upcount(n)* *Dim i As Single, sum As Single* *sum = 0* *For i = 1 To n Step 1* *sum = sum + 1 / i ^ 2* *Next i* *upcount = sum* *End Function*	*Option Explicit* *Function downcount(n)* *Dim i As Single, sum As Single* *sum = 0* *For i = n To 1 Step -1* *sum = sum + 1 / i ^ 2* *Next i* *downcount = sum* *End Function*

With single precision we get 1.644725 when we count from 1 to 10000, but get 1.644834 when we go from 10000 to 1 instead. The difference comes because we fill up our significant figures in 32 bit too fast going 1+1/4+1/9+... If we start at the high end and count backward, we don't fill up our sigfigs as fast and can add in quite a bit more detail. I believe the higher number is more correct since it includes these extra terms.

Now, repeat the two directions again, but using double precision for all numbers. Explain what is different compared to what you had found using single precision numbers.

Solution: When we do double precision we get 1.644834 and 1.6448341 for each one. It appears that 1.644834 is the right answer. Now they are both in agreement.

4) We began a Taylor series approximation of a two variable function in class. Complete your second order approximation to the function we used: $e^{2x}y^3$ for $x_{i+1} = 1$ and $y_{i+1} = 2.1$. Report your value after the first order corrections are added and then after the second order corrections are added. What is the absolute error?

Solution: We should be orderly like we laid out in the Computer Readings so we'll set up some basic information: We start by saying that $x_{i+1} = 1$ and $y_{i+1} = 2.1$. We need to select x_i and y_i so we choose $x_i = 0$ because we know e^0, and choose $y_i = 2$ because we know $2^3 = 8$. Now we need to figure out the first derivative with respect to each variable and evaluate it at x_i,y_i:

$$f(x,y)=e^{2x}y^3\Big|_{0,2}=e^{2(0)}2^3=2^3=8$$

$$\frac{\partial f}{\partial x}\Big|_{x_i,y_i}=2e^{2x}y^3\Big|_{0,2}=2e^{2(0)}2^3=2^4=16$$

$$\frac{\partial f}{\partial y}\Big|_{x_i,y_i}=3e^{2x}y^2\Big|_{0,2}=3e^{2(0)}2^2=3(4)=12$$

Let's plug this into our overall approximation for second order and find out what we have so far:

$$f(x_{i+1},y_{i+1}) \approx 8+16\frac{1}{1!}(1-0)+12\frac{1}{1!}(2.1-2)+\left.\frac{\partial^2 f}{\partial x^2}\right|_{x_i,y_i}\frac{1}{2!}(x_{i+1}-x_i)^2+\left.\frac{\partial^2 f}{\partial y^2}\right|_{x_i,y_i}\frac{1}{2!}(y_{i+1}-y_i)^2+$$

$$\frac{2}{2!}\left(\left.\frac{\partial^2 f}{\partial x\partial y}\right|_{x_i,y_i}\right)((x_{i+1}-x_i)(y_{i+1}-y_i))\cdots$$

$$f(x_{i+1},y_{i+1}) \approx 8+16+12(0.1)=25.2+\left.\frac{\partial^2 f}{\partial x^2}\right|_{x_i,y_i}\frac{1}{2!}(x_{i+1}-x_i)^2+\left.\frac{\partial^2 f}{\partial y^2}\right|_{x_i,y_i}\frac{1}{2!}(y_{i+1}-y_i)^2+$$

$$\frac{2}{2!}\left(\left.\frac{\partial^2 f}{\partial x\partial y}\right|_{x_i,y_i}\right)((x_{i+1}-x_i)(y_{i+1}-y_i))\cdots$$

The actual answer we are going for (using a calculator) is 68.43 so we need to add the second order terms next. The second order terms are:

$$\left.\frac{\partial^2 f}{\partial x^2}\right|_{x_i,y_i} = 4e^{2x}y^3\Big|_{0,2} = 4(2)^3 = 32$$

$$\left.\frac{\partial^2 f}{\partial y^2}\right|_{x_i,y_i} = 6e^{2x}y\Big|_{0,2} = 6(2) = 12$$

$$\left(\left.\frac{\partial^2 f}{\partial x\partial y}\right|_{x_i,y_i}\right) = 6e^{2x}y^2\Big|_{0,2} = 6(2)^2 = 24$$

We plug all these into our Taylor series expansion to get:

$$f(x_{i+1},y_{i+1}) \approx 25.2+32\frac{1}{2!}(1-0)^2+12\frac{1}{2!}(2.1-2)^2+\frac{2}{2!}24((1-0)(2.1-2))$$

$$f(x_{i+1},y_{i+1}) \approx 25.2+16+6(0.01)+24(0.1)=27.66$$

The absolute error is 68.43-27.66 = 40.77.