COMPUTATION ERRORS AND THE CALCULUS STUDENT

by Peter Shenkin
John Jay College of Criminal Justice

Calculus is most often taught in what we shall call the 'classical mathematical manner.' By this we mean that a theory is constructed in which the existence of solutions (or lack of solutions) to certain general types of problems is proved or demonstrated. Examples generally illustrate the given result almost exactly with 'closed form' solutions the rule. It seems that for the most part both the topics taught and the methods of teaching have not changed much for the past 20 (50) years. The advent of handheld calculators and personal computers present an opportunity to modernize the ways in which the concepts of calculus are taught and possibly to modify the core concepts. We feel that these opportunities have not yet been adequately exploited. It is true that many calculus texts have added problems which can be solved by calculator or computer, but for the most part these problems are not integrated into the course itself. The prospect for change is hopeful. Several authors have already written software packages for calculus courses and there are several independently written commercial packages such as those written by John Kemeny at True Basic, Inc. This paper discusses results from several programs and software packages developed at the John Jay College and currently being used in calculus and precalculus sections. In particular, we shall give special emphasis to errors which may occur when computers are used to 'prove' and evaluate results.

An obvious application where computers may be used is in the estimation of limits. It is quite easy, for example, to write a program in BASIC, PASCAL or some higher level language which will permit the student to guess (prove) what the limit of a certain function at a certain point is by evaluating the function at several arguments getting nearer and nearer to the desired point (or just getting larger and larger if the limit is at infinity.) The student may see that the value seems to be approaching some limiting value. Even though this does not actually show that a limit exists we feel that a computer demonstration of how a sequence seems to approach some value is a very valuable way of demonstrating just what a limit is.

There are problems with this numerical approach, however. The particular type of problem we would like to talk about in this paper involves various types of errors which are generated when using computers for computations in calculus course. If these errors are not considered then computer results can be misleading and even entirely incorrect.

A typical example is the evaluation of the limit of (1+1/x)^x as x goes to infinity. From classical calculus we know that the limit is e or about 2.718281828459045. The results from the programs ESINGS.BAS, and ESINGERR.BAS displayed in the appendix were calculated using Microsoft's QUIKBASIC 4.0 and show an interesting result. If we keep x below about 2500 then it seems that as x gets large our function is approaching 2.71.... So the student in effect 'proves' what the limit is by looking at a finite number of terms. This is certainly not correct in the classical mathematical sense but to most students this seems like going to the limit. In this particular case a seemingly (to the student) strange thing happens x is increased further and further. At first it seems that there is no limit. Then we reach a new limit. This new limit seems to be 1. How did this occur?

In actuality, BASIC was only holding 7 significant digits of every number. 1+1/x eventually becomes 1 (in the computer) as x increases and 1 to any power is 1. Thus just about all significance was lost. Pity the poor student who 'proved' that the limit is 1 via this example. Notice that in ESINGERR.BAS we managed to display more than 7 digits even though BASIC only supports 7 in its 'single precision' mode. Even the values of 1 + 1/x are incorrect in ESINGERR.BAS as the 1 + 1/x ERROR column indicates. Note, also from ESINGERR.BAS that even though single precision gives 7 digits of accuracy our estimate seems to be most accurate when x has only 4 digits (x=2560).

EDOUB.BAS is the essentially the same program as ESINGERR.BAS with the exception that all computations are done in 'double precision' mode. This means that there are 15 or 16 digits of accuracy in values and computations. Notice how much closer our results come to the 'true' limit in this case. However, we do give a printout which shows that even with double precision, if we let x increase enough we get a 'limit' of 1.

The graphs of (1 + 1/x)^x shown were generated by a graphics program described in "A Computer Companion for Undergraduate Mathematics" by Wieschenberg and Shenkin and graphically display the results shown in EDOUB.BAS. The graphics program was written in True BASIC. These graphs seem to indicate that (1 + 1/x)^x approaches zero very rapidly (in the computer) for a value of x in the neighborhood of 9E15.

When the computer is used to numerically solve calculus type problems several types of errors may occur:

Roundoff error - Due to the 'discreteness' of the computer's number system. e.g. using QUIKBASIC single precision only 7 digits matter.

Truncation error - Occurs when a process requiring an infinite number of steps is terminated after a finite number of steps, e.g. stop the above process for x=2560 or truncate all Taylor series terms after the fifth term.

Propagated error - Error caused by error in some initial input, e.g. approximate PI.

Significance error - number of meaningful digits (significant digits) in an answer is less than expected.

Overflow and Underflow Error - Error caused when calculations, including intermediate calculations, get larger than the computer's infinity or closer to 0 than the smallest computer non-zero value.

We will look at the definite integral of $\exp(-x^2)$ to examine some of these types of error.

The program ETOXDBL.BAS (see Appendix) computes the definite integral of $\exp(-x^2)$ between 0 and some value input during the run of the program. The computations are done in double precision. A companion program,ETOX.BAS computes the same results using single precision arithmetic This program is not shown in the Appendix These programs use Taylor series approximations with a number of terms also chosen during the run of the program. The series for $\exp(-x^2)$ and the resulting integral approximation are convergent alternating series so the last term gives a good idea of the size of the error at least if enough terms are taken. If we look at some runs of the programs we see several things regarding error.

1. Truncation error can be a major factor. In fact look at the display showing an integral with limits from 0 to 3 with 13 terms has a value of over 70 for the last retained term. However, if the upper limit equals 1 the last term is on the order of 10^{-11}.

2. The classical truncation error bound (1st neglected term with alternating series of decreasing terms) is meaningless in single precision when these bounds yield values on the order of 10^{-9} while we are working with integral of order 1 with six or seven significant digits.

3. By comparing single and double precision printouts we see that roundoff error isn't severe but single precision is not necessarily accurate to 7 significant digits. Of course in the limiting process for e mentioned previously we saw that roundoff and significance error might be severe.

The results from the programs RTSSNG, RTSDBL compare the rectangular, trapezoidal and Simpsons rule for various interval widths. Note some of the following results:

1. The example printout from RTSSNG.BAS with $\exp(-t^2)$ as the integrand shows that, at least in the single precision case, increasing the number of subintervals does not necessarily improve the approximation.

2. This is again shown in the example where x^3 is integrated. Theoretically Simpson's rule should be exact here. However, look at the results as the number of subintervals increases.

3. We also show some results using the rectangular rule, the trapezoidal rule and Simpson's rule where we approximated e by 2.7. The results showed small changes in value all in the direction expected. This is an example of propagation error and does not seem to be serious in the integration problems we are tackling.

In conclusion, we feel that the student should be exposed to examples such as those described to aid them in results gotten by using a simple numerical rule as implemented on a computer may be more in error than theory as usually given in beginning calculus courses would indicate. The hands on, especially the interactive approach would probably be most useful to the greatest number of students.

APPENDIX

```
1 REM etoxdbl.bas
2 REM PROGRAM BY A. WIESCHENBERG
3 CLS
4 PRINT
5 PRINT "THIS PROGRAM IS TO FIND THE "
6 PRINT "DEFINITE INTEGRAL FROM 0 TO X OF E TO THE NEGATIVE T SQUARED"
7 PRINT
10 REM I = NUMBER OF TERMS DESIRED
20 REM X# = TO FIND THE INTEGRAL FROM 0 TO THIS VALUE X
30 INPUT "NUMBER OF TERMS DESIRED"; I
40 INPUT "FIND THE VALUE OF THE INTEGRAL FROM 0 TO:"; X#
50 T# = X#
60 S# = X#
70 K# = 0#
80 PRINT
90 PRINT "TERM          TERM VALUE                    SUM OF TERMS"
100 PRINT "-------------------------------------------------------------"
110 PRINT K#; TAB(13); T#; TAB(40); X#
120 FOR N = 1 TO I - 1
130     K# = K# + 1#
140     TN# = (T# * X# ^ 2# * (2# * K# - 1#)) / ((2# * K# + 1#) * K#) * (-1# ^ K#)
150     S# = S# + TN#
160     T# = TN#
170     PRINT K#; TAB(13); T#; TAB(40); S#
180 NEXT N
190 END
```

```
          THIS PROGRAM IS TO FIND THE
DEFINITE INTEGRAL FROM 0 TO X OF E TO THE NEGATIVE T SQUARED
NUMBER OF TERMS DESIRED  13
FIND THE VALUE OF THE INTEGRAL FROM 0 TO:  1
```

TERM	TERM VALUE	SUM OF TERMS
0	1	1
1	-.3333333333333333	.6666666666666667
2	.1	.7666666666666667
3	-2.380952380952381D-02	.7428571428571429
4	4.629629629629629D-03	.7474867724867725
5	-7.575757575757576D-04	.7467291967291968
6	1.068376068376068D-04	.7468360343360344
7	-1.322751322751323D-05	.7468228068228069
8	1.458916900093371D-06	.7468242657397069
9	-1.450385222315047D-07	.7468241207011848
10	1.312253296380280D-08	.7468241338237177
11	-1.089222103714857D-09	.7468241327344955
12	8.350702795147238D-11	.7468241328180025

```
          THIS PROGRAM IS TO FIND THE
DEFINITE INTEGRAL FROM 0 TO X OF E TO THE NEGATIVE T SQUARED
NUMBER OF TERMS DESIRED  13
FIND THE VALUE OF THE INTEGRAL FROM 0 TO:  3
```

TERM	TERM VALUE	SUM OF TERMS
0	3	3
1	-9	-6
2	24.3	18.3
3	-52.07142857142857	-33.77142857142857
4	91.125	57.35357142857143
5	-134.2022727272727	-76.84870129870129
6	170.3336538461538	93.48495254745254
7	-189.8003571428571	-96.31540459540454
8	188.4047662815126	92.08936168610802
9	-168.5726856203007	-76.48332393541927
10	137.2663297193877	60.78300578519503
11	-102.5428312923489	-41.75982550715389
12	70.75455359172075	28.99472808456686

```
REM RTSDBL.BAS
REM RECTANGULAR, TRAPEZOIDAL, SIMPSONS'S RULE
REM DOUBLE PRECISION COMPUTATIONS
REM SHENKIN, WIESCHENBERG  01/06/88
DEFINT N
DEFDBL H, R-Z
DIM V(5000)
DEF FNF# (X) = EXP(-X ^ 2)
CLS
PRINT "INTEGRAL OF EXP(-X^2)"
PRINT "USING RECTANGULAR, TRAPEZOIDAL, & SIMPSON'S RULE "
PRINT
INPUT "DO YOU DESIRE PRINTED OUTPUT(Y/N) "; PRT$
INPUT "ENTER NUMBER OF SUBINTERVALS DESIRED (EVEN PLEASE) "; N
INPUT "THE INTEGRAL WILL BE FROM 0 TO ?                   "; X

H = X / N     'H = STEP SIZE

FOR I = 0 TO N
    V(I) = FNF#(I * X / N)
NEXT I

FOR I = 0 TO N
    IF I <> N THEN
        RN = RN + V(I)
    END IF

    IF I = 0 OR I = N THEN
       TN = TN + V(I) / 2
    ELSE
       TN = TN + V(I)
    END IF

    IF I = 0 OR I = N THEN
       SN = SN + V(I)
    ELSEIF I / 2 <> INT(I / 2) THEN
         SN = SN + 4 * V(I)
         ELSEIF I / 2 = INT(I / 2) THEN
         SN = SN + 2 * V(I)
    END IF
NEXT I
RN = RN * H
TN = TN * H
SN = SN * H / 3

PRINT
PRINT "INTEGRAL BY RECTANGULAR RULE = "; RN
PRINT "INTEGRAL BY TRAPEZOIDAL RULE = "; TN
PRINT "INTEGRAL BY SIMPSON'S RULE   = "; SN

IF UCASE$(PRT$) = "Y" THEN
    LPRINT "INTEGRAL OF EXP(-T^2) FROM 0 TO "; X
    LPRINT "NUMBER OF SUBINTERVALS =        "; N
    LPRINT "LENGTH OF ONE SUBINTERVAL       "; H

    LPRINT
    LPRINT "INTEGRAL BY RECTANGULAR RULE = "; RN
    LPRINT "INTEGRAL BY TRAPEZOIDAL RULE = "; TN
    LPRINT "INTEGRAL BY SIMPSON'S RULE   = "; SN
    LPRINT
    LPRINT
    LPRINT
END IF
```

```
RESULTS FROM RTSSNG.BAS
THESE ARE SINGLE PRECISION COMPUTATIONS

INTEGRAL OF EXP(-X^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         10
LENGTH OF ONE SUBINTERVAL        .1

INTEGRAL BY RECTANGULAR RULE =  .7778168
INTEGRAL BY TRAPEZOIDAL RULE =  .7462108
INTEGRAL BY SIMPSON'S RULE   =  .746825


INTEGRAL OF EXP(-X^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         100
LENGTH OF ONE SUBINTERVAL        .01

INTEGRAL BY RECTANGULAR RULE =  .7499784
INTEGRAL BY TRAPEZOIDAL RULE =  .7468178
INTEGRAL BY SIMPSON'S RULE   =  .7468241


INTEGRAL OF EXP(-X^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         1000
LENGTH OF ONE SUBINTERVAL        .001

INTEGRAL BY RECTANGULAR RULE =  .7471396
INTEGRAL BY TRAPEZOIDAL RULE =  .7468236
INTEGRAL BY SIMPSON'S RULE   =  .7468238


INTEGRAL OF EXP(-X^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         2000
LENGTH OF ONE SUBINTERVAL        .0005

INTEGRAL BY RECTANGULAR RULE =  .7469828
INTEGRAL BY TRAPEZOIDAL RULE =  .7468247
INTEGRAL BY SIMPSON'S RULE   =  .7468238


INTEGRAL OF EXP(-X^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         4000
LENGTH OF ONE SUBINTERVAL        .00025

INTEGRAL BY RECTANGULAR RULE =  .746903
INTEGRAL BY TRAPEZOIDAL RULE =  .746824
INTEGRAL BY SIMPSON'S RULE   =  .7468243
```

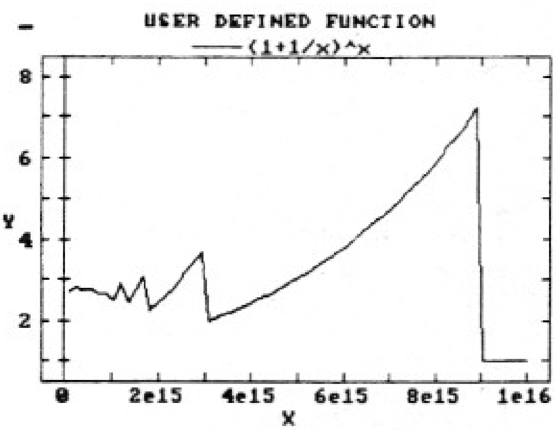
```
RESULTS ARE FROM RTSDBL.BAS
THESE ARE DOUBLE PRECISION COMPUTATIONS

INTEGRAL OF EXP(-T^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         10
LENGTH OF ONE SUBINTERVAL        .1

INTEGRAL BY RECTANGULAR RULE =  .7778168240731773
INTEGRAL BY TRAPEZOIDAL RULE =  .7462107961317495
INTEGRAL BY SIMPSON'S RULE   =  .7468249482544435


INTEGRAL OF EXP(-T^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         100
LENGTH OF ONE SUBINTERVAL        .01

INTEGRAL BY RECTANGULAR RULE =  .7499786042621125
INTEGRAL BY TRAPEZOIDAL RULE =  .7468180014679697
INTEGRAL BY SIMPSON'S RULE   =  .7468241328941758


INTEGRAL OF EXP(-T^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         1000
LENGTH OF ONE SUBINTERVAL        .001

INTEGRAL BY RECTANGULAR RULE =  .7471401317785986
INTEGRAL BY TRAPEZOIDAL RULE =  .7468240714991844
INTEGRAL BY SIMPSON'S RULE   =  .7468241328124359


INTEGRAL OF EXP(-T^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         2000
LENGTH OF ONE SUBINTERVAL        .0005

INTEGRAL BY RECTANGULAR RULE =  .7469821476238258
INTEGRAL BY TRAPEZOIDAL RULE =  .7468241174841186
INTEGRAL BY SIMPSON'S RULE   =  .7468241328124275


INTEGRAL OF EXP(-T^2) FROM 0 TO  1
NUMBER OF SUBINTERVALS =         4000
LENGTH OF ONE SUBINTERVAL        .00025

INTEGRAL BY RECTANGULAR RULE =  .7469031440502019
INTEGRAL BY TRAPEZOIDAL RULE =  .7468241289803483
INTEGRAL BY SIMPSON'S RULE   =  .7468241328124258
```

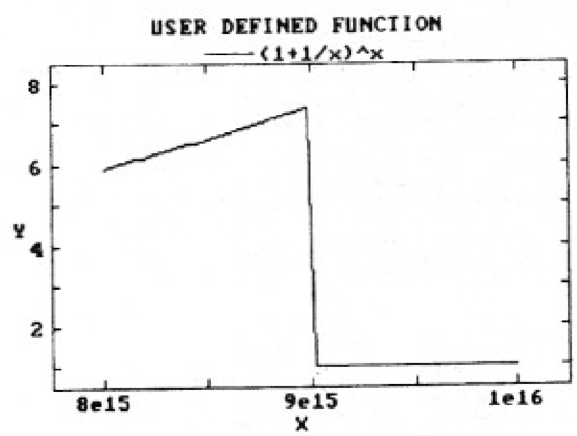
```
INTEGRAL OF X^3 FROM 0 TO  1
NUMBER OF SUBINTERVALS =         10
LENGTH OF ONE SUBINTERVAL        .1

INTEGRAL BY RECTANGULAR RULE =  .2025
INTEGRAL BY TRAPEZOIDAL RULE =  .2525
INTEGRAL BY SIMPSON'S RULE   =  .25


INTEGRAL OF X^3 FROM 0 TO  1
NUMBER OF SUBINTERVALS =         100
LENGTH OF ONE SUBINTERVAL        .01

INTEGRAL BY RECTANGULAR RULE =  .245025
INTEGRAL BY TRAPEZOIDAL RULE =  .250025
INTEGRAL BY SIMPSON'S RULE   =  .25


INTEGRAL OF X^3 FROM 0 TO  1
NUMBER OF SUBINTERVALS =         1000
LENGTH OF ONE SUBINTERVAL        .001

INTEGRAL BY RECTANGULAR RULE =  .2495003
INTEGRAL BY TRAPEZOIDAL RULE =  .2500003
INTEGRAL BY SIMPSON'S RULE   =  .2500001


INTEGRAL OF X^3 FROM 0 TO  1
NUMBER OF SUBINTERVALS =         2000
LENGTH OF ONE SUBINTERVAL        .0005

INTEGRAL BY RECTANGULAR RULE =  .2497501
INTEGRAL BY TRAPEZOIDAL RULE =  .2500001
INTEGRAL BY SIMPSON'S RULE   =  .25


INTEGRAL OF X^3 FROM 0 TO  1
NUMBER OF SUBINTERVALS =         4000
LENGTH OF ONE SUBINTERVAL        .00025

INTEGRAL BY RECTANGULAR RULE =  .2498751
INTEGRAL BY TRAPEZOIDAL RULE =  .2500001
INTEGRAL BY SIMPSON'S RULE   =  .2500002
```

```
REM ESINGS.BAS

LET h$ = "           x                    1 + 1/x              (1 + 1/x)^x"
LET f$ = " ###,###,###,### "
LET xmax = 1E+08

CLS
LPRINT "Problem: Evaluate e using (1+1/x)^x for various x"
LPRINT "         Calculate in single precision "
LPRINT

LPRINT h$
x = 10
DO WHILE x < xmax
   LET t1 = 1 + 1 / x
   LET t2 = t1 ^ x
   LPRINT USING f$; x; : LPRINT TAB(25); t1; TAB(47); t2
   x = 2 * x
LOOP
LPRINT CHR$(12)
```

Problem: Evaluate e using (1+1/x)^x for various x
Calculate in single precision

x	1 + 1/x	(1 + 1/x)^x
10	1.1	2.593743
20	1.05	2.653295
40	1.025	2.685061
80	1.0125	2.701495
160	1.00625	2.709846
320	1.003125	2.714005
640	1.001562	2.71612
1,280	1.000781	2.717386
2,560	1.000391	2.717917
5,120	1.000195	2.717353
10,240	1.000098	2.717486
20,480	1.000049	2.720871
40,960	1.000024	2.720904
81,920	1.000012	2.707668
163,840	1.000006	2.707676
327,680	1.000003	2.761084
655,360	1.000002	2.761086
1,310,720	1.000001	2.553589
2,621,440	1	2.553589
5,242,880	1	3.490342
10,485,760	1	3.490343
20,971,520	1	1
41,943,040	1	1
83,886,080	1	1






```
REM ESINGERR.BAS

WIDTH "lpt1:", 132
LET h0$ = "                                                                    1 + 1/x              (1 + 1/x)^x"
LET h$ = "           x              1 + 1/x            (1 + 1/x)^x           ERROR                ERROR       "
LET f$ = " ###,###,###,###     #.################     #.################   ##.################   ##.################"
LET xmax = 1E+08

CLS
LPRINT "Problem: Evaluate e using (1+1/x)^x for various x"
LPRINT "         Calculate in single precision but display excess characters"
LPRINT "         Machine double precision value of e equals ", EXP(1#)
LPRINT

LPRINT h0$
LPRINT h$
x = 10
x# = 10#
DO WHILE x < xmax
   LET t1 = 1 + 1 / x
   LET t2 = t1 ^ x
   LET t1# = 1# + 1# / x#
   LPRINT USING f$; x, t1, t2, t1 - t1#, t2 - EXP(1#)
   x = 2 * x
   x# = 2# * x#
LOOP
LPRINT CHR$(12)
```

Problem: Evaluate e using (1+1/x)^x for various x
Calculate in single precision but display excess characters
Machine double precision value of e equals 2.718281828459045

x	1 + 1/x	(1 + 1/x)^x	1 + 1/x ERROR	(1 + 1/x)^x ERROR
10	1.100000023841858	2.593743085861206	0.0000000238418579	-0.1245387425978390
20	1.049999952316284	2.653295278549194	-0.0000000476837158	-0.0649865499098508
40	1.024999976158142	2.685061454772949	-0.0000000238418579	-0.0332203736860959
80	1.012500047683716	2.701495170593262	0.0000000476837158	-0.0167866578657834
160	1.006250023841858	2.709845781326294	0.0000000238418579	-0.0084360471327511
320	1.003124952316284	2.714005470275879	-0.0000000476837158	-0.0042763581831662
640	1.001562476158142	2.716119766235352	-0.0000000238418579	-0.0021620622236935
1,280	1.000781297683716	2.717386484146118	0.0000000476837158	-0.0008953443129269
2,560	1.000390648841858	2.717916965484619	0.0000000238418579	-0.0003648629744260
5,120	1.000195264816284	2.717353105545044	-0.0000000476837158	-0.0009287229140011
10,240	1.000097632408142	2.717485666275024	-0.0000000238418579	-0.0007961621840207
20,480	1.000048875808716	2.720871210098267	0.0000000476837158	0.0025893816392215
40,960	1.000024437904358	2.720904350280762	0.0000000238418579	0.0026225218217166
81,920	1.000012159347534	2.707667827606201	-0.0000000476837158	-0.0106140008528439
163,840	1.000006079673767	2.707676172256470	-0.0000000238418579	-0.0106056562025754
327,680	1.000003099441528	2.761084079742432	0.0000000476837158	0.0428022512833866
655,360	1.000001549720764	2.761086463928223	0.0000000238418579	0.0428046354691776
1,310,720	1.000000715255737	2.553588628768921	-0.0000000476837158	-0.1646931996901242
2,621,440	1.000000357627869	2.553589105606079	-0.0000000238418579	-0.1646927228529660
5,242,880	1.000000238418579	3.490342378616333	0:0000000476837158	0.7720605501572879
10,485,760	1.000000119209290	3.490342617034912	0.0000000238418579	0.7720607885758670
20,971,520	1.000000000000000	1.000000000000000	-0.0000000476837158	-1.7182818284590450
41,943,040	1.000000000000000	1.000000000000000	-0.0000000238418579	-1.7182818284590450
83,886,080	1.000000000000000	1.000000000000000	-0.0000000119209291	-1.7182818284590450

```
REM EDOUB.BAS

WIDTH "lpt1:", 132
LET h0$ = "
                                                                        "
LET h$ = "           x                     1 + 1/x              (1 + 1/x)^x              ERROR      "
LET f$ = " ##,###,###,###,###,###     #.################     #.################   ##.################"
LET xmax = 9E+16

CLS
LPRINT "Problem: Evaluate e using (1+1/x)^x for various x"
LPRINT "         Calculate in double precision"
LPRINT "         Machine double precision value of e equals "; EXP(1#)
LPRINT

LPRINT h$
x# = 10#
DO WHILE x# < xmax
   LET t1# = 1# + 1# / x#
   LET t2# = t1# ^ x#
   LPRINT USING f$; x#, t1#, t2#, t2# - EXP(1#)
   x# = 4# * x#
LOOP
LPRINT CHR$(12)
```

Problem: Evaluate e using (1+1/x)^x for various x
Calculate in double precision
Machine double precision value of e equals 2.718281828459045

x	1 + 1/x	(1 + 1/x)^x	ERROR
10	1.100000000000000	2.593742460100002	-0.1245393683590428
40	1.025000000000000	2.685063838389963	-0.0332179900690819
160	1.006250000000000	2.709835576307815	-0.0084462521512298
640	1.001562500000000	2.716161207947854	-0.0021206205111914
2,560	1.000390625000000	2.717751104075259	-0.0005307243837862
10,240	1.000097656250000	2.718149111732190	-0.0001327167268550
40,960	1.000024414062500	2.718248647060292	-0.0000331813987531
163,840	1.000006103515625	2.718273532928095	-0.0000082955309497
655,360	1.000001525878906	2.718279754735719	-0.0000020737233264
2,621,440	1.000000381469726	2.718281309355215	-0.0000005191038306
10,485,760	1.000000095367432	2.718281701372875	-0.0000001270861700
41,943,040	1.000000023841858	2.718281785928213	-0.0000000425308322
167,772,160	1.000000005960465	2.718281860863490	0.0000000324044449
671,088,640	1.000000001490116	2.718281664411550	-0.0000001640474951
2,684,354,560	1.000000000372529	2.718282476041694	0.0000006475826488
10,737,418,240	1.000000000093132	2.718279235978137	-0.0000025924809077
42,949,672,960	1.000000000023283	2.718292197869436	0.0000103694103912
171,798,691,840	1.000000000005821	2.718240351078550	-0.0000414773804955
687,194,767,360	1.000000000001455	2.718447744276496	0.0001659158174512
2,748,779,069,440	1.000000000000364	2.717618266438544	-0.0006635620205011
10,995,116,277,760	1.000000000000091	2.720937697156844	0.0026558686977989
43,980,465,111,040	1.000000000000023	2.707684251933760	-0.0105975765252855
175,921,860,444,160	1.000000000000006	2.761088538530093	0.0428067100910483
703,687,441,776,640	1.000000000000001	2.553589438062925	-0.1646923703961196
2,814,749,767,106,560	1.000000000000000	3.490342957461841	0.7720611290027954
11,258,999,068,426,240	1.000000000000000	1.000000000000000	-1.7182818284590450
45,035,996,273,704,960	1.000000000000000	1.000000000000000	-1.7182818284590450