[6.60] Faster, more accurate exponential integrals

A post on the comp.sys.hp48 newsgroup announced that the HP-49G did not accurately find a numerical value for this integral

$$\int_{-3.724}^{\infty} \frac{e^{-x}}{x} dx$$

Except for a sign change, this is identical to the special function called the exponential integral:

$$Ei(x) = -\int_{-x}^{\infty} \frac{e^{-x}}{x} dx$$

The TI-89 / TI-92 Plus also cannot find accurate values for this integral, and it is no wonder, since at the integrand singularity at x = 0, the limit is +infinity from the right and -infinity from the left. One expedient solution is a routine specifically designed to find the exponential integral:

```
ei(x)
Func
©(x) exponential integral Ei
©13jul02/dburkett@infinet.com

local eps,euler,fpmin,maxit,k,fact,prev,sum1,term

6ᴇ⁻12→eps                            © Desired relative accuracy
.577215664901532→euler               © Euler's constant
100→maxit                            © Maximum iterations allowed for convergence
1ᴇ⁻900→fpmin                         © Number near floating-point minimum

if x≤0:return "ei arg error"         © Argument must be > 0
if x<fpmin then                      © Handle very small arguments
 return euler+ln(x)
elseif x≤⁻ln(eps) then               © Use power series for x < 25.840
 0→sum1
 1→fact
 for k,1,maxit
  fact*x/k→fact
  fact/k→term
  sum1+term→sum1
  if term<eps*sum1
   return sum1+ln(x)+euler
 endfor
 return "ei failed series"           © Return error string if convergence fails
else                                 © Use asymptotic expansion for large arguments
0→sum1
 1→term
 for k,1,maxit
  term→prev
  term*k/x→term
  if term<eps:goto l1
  if term<prev then
   sum1+term→sum1
  else
   sum1-prev→sum1
   goto l1
  endif
 endfor
 lbl l1
 return e^(x)*(1+sum1)/x
endif

EndFunc
```

I ported this algorithm, from *Numerical Recipes in Fortran*, to TI Basic. The input argument must be greater than zero, and error message strings may be returned. You can check for this condition by using *getType()* on the result.

This table shows some results for the original example problem.

Method	Result	Correct significant digits	Execution time
-ei(3.724)	-16.2252 9269 7647	About 11	2 sec
`∫(e^(-x))/x,x,-3.724,∞)`	-16.2967 8867 2016	2	134 sec
`nInt(e^(-x))/x,x,-3.724,∞)`	-13.5159 0839 6945	1	53 sec
`nInt(e^(-x))/x,x,-3.724,Ø) + nInt(e^(-x))/x,x,Ø,∞)`	-16.2967 8867 2016	2	56 sec

It is interesting that the built-in `∫()` function is slightly more accurate that the purely numerical *nInt()*, perhaps because it symbolically finds the singularity at x = 0, and integrates over two ranges divided at x = 0. Circumstantial evidence for this supposition is provided by that fourth method, which manually uses *nInt()* over the two ranges, and returns the same result as `∫()`.

Bhuvanesh Bhatt has also written a function for Ei(x), and you can get it at

http://tiger.towson.edu/~bbhatt1/ti/

This is a C function called *ExpIntEi()*, found in his C Special Functions package. You will need the usual hacks to use this as a true function on HW2 calculators; see the site for details.

For more information on the exponential integral, try

Handbook of Mathematical Functions, Milton Abramowitz and Irene A. Stegun, Dover, 1965. Section 5 describes and defines Ei(x), as well as its related integrals and interrelations. I used the table of numerical values to test *ei(x)*.

Atlas for Computing Mathematical Functions, William J. Thompson, Wiley-Interscience, 1997. Thompson calls Ei(x) the 'exponential integral of the second kind', and coverage begins in section 5.1.2. You can often get this book inexpensively from the bookseller Edward R. Hamilton, at *http://www.hamiltonbook.com/*

As mentioned, the algorithm for *ei(x)* comes from

Numerical Recipes in Fortran, 2e, William H. Press et al, Cambridge University Press, 1992. Section 6.3 covers the exponential integrals. This book is available on-line at *http://www.nr.com*.

You can also use *ei(x)* to find values for the logarithmic integral li(x):

$$\mathrm{li}(x) = \oint_0^x \frac{dt}{\ln(t)} \qquad x > 1$$

since li(x) = Ei(ln(x))