

# Boldly Going ... Identifying Constants

© Valentín Albilló (PPC #4747)

Welcome to another installment of the new **Boldly Going** series. This time it features a relatively simple program which builds upon the extremely small general purpose routine **DEC2FRC** (featured/discussed elsewhere in this issue), to provide basic functionality for an advanced, very useful and *most impressive* feature which is nevertheless absent in our beloved machines' built-in instruction sets, namely **identifying numeric constants**, i.e., the capability of, *given some real, numeric value, to try and identify its exact symbolic form* if possible, and that failing, to provide an approximate symbolic expression of user-specified relative accuracy.

This will allow us to perform some pretty nifty feats, such as:

- Give *exact, symbolic* results for definite *integrals* (even if they can't be expressed in terms of elementary functions), finite or infinite *summations*, and specific values of transcendental functions. For instance, we'll find that

$$\int_0^{\frac{\pi}{2}} x^2 \operatorname{Ln}^2(2 \operatorname{Cos}(x)) \cdot dx \quad \text{equals} \quad \frac{11}{1440} \pi^5$$

- *Simplifying* certain complicated expressions by identifying the computed result as a much simpler symbolic expression. For instance,

$$\frac{\operatorname{Sinh} \frac{\pi}{4}}{\operatorname{Cosh} \frac{\pi}{4} - \operatorname{Sinh} \frac{\pi}{4}} + \frac{\operatorname{Cosh} \frac{\pi}{4}}{\operatorname{Cosh} \frac{\pi}{4} + \operatorname{Sinh} \frac{\pi}{4}} \quad \text{equals} \quad \operatorname{Cosh} \frac{\pi}{2}$$

- Perform *exact arithmetic* with expressions involving *fractions*. For instance:

$$\frac{1}{7} + \frac{2}{13} - \frac{3}{19} + \frac{1}{23} \quad \text{equals} \quad \frac{7249}{39767}$$

- Find out *simpler* or *alternate* symbolic forms. For instance:

$$\frac{\sqrt{3}-1}{2\sqrt{2}} \quad \text{equals} \quad \operatorname{Sin}(15^\circ)$$

- Identify the exact *symbolic* value of some given *limit*. For instance:

$$\operatorname{Lim}_{x \rightarrow 0} (1 + \operatorname{Sin}(x))^{\operatorname{Cot}(2x)} \quad \text{equals} \quad \sqrt{e}$$

## Program listing for the HP-71B

This 31-line (1,562-byte) **BASIC** program listing consists of the base subprogram **DEC2FRC** and the main subprogram **IDENTIFP**, but for convenience's sake there's also a 1-line subprogram **IDENTIFY** to allow for simpler calls using *defaults*, as well as a small front-end program to provide interactive input and labeled output.

### *The front-end, "driver" main program*

```
10 DESTROY ALL @DIM S$[80] @STD @T$="identified as " @INPUT "Value=";X
12 INPUT "#Cn,Pw,Rt,Fn,Err=","3,3,3,4,1E-9";C,P,R,F,K
14 CALL IDENTIFP(X,S$,C,P,R,F,K,V) @ IF V<95 THEN T$="might be "
16 DISP X;T$;S$;" (" ;STR$(V) ;"%")
```

### *The simpler call with default parameters*

```
18 SUB IDENTIFY(X,S$) @ CALL IDENTIFP(X,S$,3,3,3,4,.00000001,0)
```

### *The full-fledged identifying subprogram*

```
20 SUB IDENTIFP(X,S$,B,L,A,F,K,V) @ OPTION BASE 1 @ DIM T$[80],G$[80]
22 DATA 6,PI,EXP(1),LN(2),.577215664902,(1+SQR(5))/2,PI*LN(2)
24 DATA "Pi","e","Ln(2)","EulerGamma","Phi","(Pi*Ln(2))"
26 DATA 26,(,),(),Sin(),Asin(),Cos(),Acos(),Tan(),Atan(),Exp(),Ln(),10^(),Lgt()
28 DATA 2^(),Log2(),Sinh(),Asinh(),Cosh(),Acosh(),Tanh(),Atanh()
30 DATA 1/Sin(),Asin(1/()),1/Cos(),Acos(1/()),1/Tan(),Atan(1/())
32 READ M @ DIM C(M),C$(M) @ READ C,C$,Z @ DIM F$(Z) @READ F$ @ W1=INF
34 U=ABS(X) @Z=MAX(1,MIN(Z,2*F+2)) @M=MIN(M,B) @A=MAX(1,A) @L=MAX(1,L)
36 ON ERROR GOTO 44 @ FOR J=2 TO Z @ G$=F$(J) @ P=POS(G$,"()")
38 Y=VAL(G$[1,P]&"U"&G$[P+1]) @ IF J=2 THEN H=40 ELSE H=1
40 FOR R=1 TO A @P=0@S=1 @W=4*H @GOSUB 46 @FOR I=1 TO M @FOR P=-L TO L
42 P=P+(P=0) @ S=C(I)^P @ W=H @ GOSUB 46 @ NEXT P @ NEXT I @ NEXT R
44 NEXT J @ OFF ERROR @ GOTO 52
46 CALL DEC2FRC(Y^R*S,N,D,K) @ W=(N*N+D*D)/W
48 IF W<W1 THEN V=W/W1 @ W1=W @ N1=N @ D1=D @ I1=I @ P1=P @R1=R @ J1=J
50 IF ABS(N)+ABS(D)>1100 THEN RETURN
52 S$="([2-(R1#1)] @ T$=STR$(N1) @ IF D1#1 THEN T$=T$&"/"&STR$(D1)
54 IF P1=0 THEN 58 ELSE T$=T$&"*"/"[1+(P1>0),1+(P1>0)]@IF T$="1*" THEN T$=""
56 T$=T$&C$(I1) @ IF ABS(P1)#1 THEN T$=T$&"^"&STR$(ABS(P1))
58 S$=S$&T$ @ Q=F$(J1)<>"()" @ IF R1#1 THEN S$=S$&")^(1/"&STR$(R1)&")"
60 IF Q THEN G$=F$(J1+2*MOD(J1,2)-1) @ Q=POS(G$,"()")
   @ S$=G$[1,Q]&S$&G$[Q+1]
62 S$="-"[SGN(X)+2]&S$ @ V=100-INT(100*V)
```

### *The base Decimal-to-Fraction subprogram*

```
64 SUB DEC2FRC(X,N,D,W) @V=1 @N=1 @D=0 @Y=INF @Z=ABS(X) @F=SGN(X) @X=Z
66 C=INT(X)@IF FP(X) THEN X=1/FP(X)@S=N ELSE N=(N*C+U)*F @D=D*C+V @END
68 T=D @ N=N*C+U @ U=S @ D=D*C+V @ V=T @ R=N/D
   @ IF ABS(R/Z-1)<=W THEN N=N*F @ END
70 IF R=Y OR MAX(N,D)>1E12 THEN N=U*F @ D=V ELSE Y=R @ GOTO 66
```

**Note:** No ROMs required to *enter/use* the program, but the Math ROM gets heavily used in the examples.

## Program description & syntax

As stated above, the program listing includes four different sections of code, namely three *subprograms* and one main, *front-end* program to allow for a convenient, interactive experience. Let's discuss them in turn, in reverse order:

### DEC2FRC: The Convert-Real-To-Fraction subprogram

The basic routine (*lines 64-70*) upon which the identifying subprogram depends. It converts a given real value to a fraction with the *lowest* possible terms, within a user-specified maximum relative error. It is discussed elsewhere in this same issue of *Datafile* but, for the sake of completeness, its calling syntax is the following:

```
CALL DEC2FRC (X,N,D,W) , where:
```

```
X      input : real value to convert to fractional form
N      output: integer numerator of the simplest fraction
D      output: integer denominator of the simplest fraction
W      input : maximum relative error (0 means maximum accuracy)
```

**Example:** Convert  $\pi$  to a rational with max. error  $\leq 1\text{E-}7$ ; with minimum error.

```
>CALL DEC2FRC (PI,N,D,1E-7) @ N;D,N/D;PI
      355 113                3.14159292035  3.14159265359
>CALL DEC2FRC (PI,N,D,0) @ N;D,N/D;PI
      1146408 364913         3.14159265359  3.14159265359
```

### IDENTIFP: The Main Identification subprogram

This is the main subprogram (*lines 20-62*) which attempts to identify the user-given real value; it can be *fine-tuned* by specifying various parameters when issuing the call, according to the following syntax:

```
CALL IDENTIFP (X,S$,B,L,A,F,K,V) , where:
```

```
X      input : real value to identify
S$     output: symbolic expression which represents the value
B      input : max. number of predefined constants to try
L      input : max. positive/negative power to try
A      input : max. Nth-root to try
F      input : max. number of predefined functions to try
K      input : max. relative error for rationalization
V      output: identification's confidence indicator (0-100%)
```

**Example:** Identify the value  $-2.34305547341$

```
>CALL IDENTIFP (-2.34305547341,S$,3,3,3,12,1E-9,V) @ S$;V
      -1/Sin((11/13/Pi^2)^(1/3))    100
```

So  $-2.34305547341$  is identified as  $-Cosec \sqrt[3]{\frac{11}{13 \pi^2}}$  with 100% confidence

## IDENTIFY: The Convenience Simpler Call with Default Parameters

This one-line subprogram (*line 18*) directly calls **IDENTIFP** with *default* parameters which are appropriate for most cases. The simple calling syntax is as follows:

**CALL IDENTIFY(X,S\$)** , where:

**X**     *input* : real value to identify  
**S\$**    *output*: symbolic expression which represents the value

The following *default* parameters are assumed:

**3**     = max. number of predefined constants to try (*Pi, e, Ln 2*)  
**3**     = max. pos/neg. power to try (*up to cubes or 1/cubes*)  
**3**     = max. N<sup>th</sup>-root to try (*up to cubic roots*)  
**4**     = max. number of predef. functions to try (*Sin, Cos, Tan, Exp*)  
**1E-9** = max. relative error for rationalization

**Example:** Compute and identify all roots of  $x^4 - 6\sqrt{3}x^3 + 8x^2 + 2\sqrt{3}x - 1 = 0$

```
>DESTROY ALL @ OPTION BASE 1 @ DIM C(5) @ COMPLEX R(4) @ MAT INPUT C
C(1)? 1,-6*SQR(3),8,2*SQR(3),-1 [ENTER]
>MAT R=PROOT(C)
>FOR I=1 TO 4 @ S=REPT(R(I)) @ CALL IDENTIFY(S,S$) @ S;"=";S$ @NEXT I
.21255656167 = Tan(1/15*Pi)   -.445228685309 = -Tan(2/15*Pi)
1.11061251483 = Tan(4/15*Pi)   9.51436445421 = Tan(7/15*Pi)
```

## The Front-end, “driver” main program

Again for convenience, a simple front-end program (*lines 10-16*) is included, which when **RUN** simply prompts the user for the value to identify (*any numeric expression*) and any desired parameters, for which defaults are offered, namely:

```
#Cn = max. number of predefined constants to try (default = 3)
,Pw = max. pos/neg. power to try (default = 3 i.e: -3 to 3)
,Rt = max. Nth-root to try (default = 3 = up to cubic roots)
,Fn = max. number of predef. functions to try (default = 4)
,Err= max. relative error for rationalization (default = 1E-9)
```

It then calls **IDENTIFP** and outputs the resulting symbolic expression along with a *confidence indicator* (0-100%) which measures the identification’s reliability: values  $\geq 95\%$  are labeled as “*identified as*”, lesser values as “*might be*”.

**Example:** Compute and identify the value of  $I = \int_0^1 \frac{x^2 - 2x - 5}{x^3 + 6x^2 + 11x + 6} . dx$

```
>RUN
Value=INTEGRAL(0,1,1E-10,(IVAR^2-2*IVAR-5)/(IVAR^3+6*IVAR^2+11*IVAR+6))
#Cn,Pw,Rt,Fn,Err=3,4,3,4,1E-10 [ENTER]
-.471132142625 might be -Ln(6561/4096) (91%)
```

which, since  $6561=9^4$  and  $4096=8^4$ , readily simplifies to  $I = 4 \operatorname{Ln} \frac{8}{9}$

## Notes and Limitations

- The identification subprogram can *initially* recognize a symbolic expression of the generic form:

$$(+ \text{ or } -) F_i \left( \sqrt[R]{\frac{N}{D} C_j^P} \right)$$

where:

**F<sub>i</sub>**: predefined function or its inverse, where  $0 \leq i \leq \mathbf{F}$  (**#Fn**). The index  $i=0$  corresponds to no function applied.

- The functions to try are *predefined* in **DATA** statements at lines 26-30. The first value is the number of functions predefined (13+13 inverses), the remaining string values are the functions' names, which *must* be the *actual* name the **HP-71B** recognizes, with the argument represented by the *empty parentheses set*, ().
- Any function can be specified in the **DATA** statements but if the name's not recognized at run time or it causes any kind of run-time error (for certain arguments, f.i.), it will be skipped.
- By default, **F** is taken as **4**, i.e.: **SIN**, **COS**, **TAN**, **EXP**, and their inverses will be tried. Values of **F** in 6-9 include *hyperbolic functions* and require the Math ROM, else they will get skipped. Values of **F** in 10-12 define extra trigonometric functions: *Cosecant*, *Secant*, *Cotangent*, and their inverses.
- You can *extend* the identification capabilities by *adding your own* functions, including user-defined functions. See *Example 3* below for details. Running time is linear.

**R**:  $R^{\text{th}}$ -root to apply, where R goes from 1 to **A**. (**#Rt**) (1=no root)

**N**: Integer numerator or the simplest fraction within max.err. **K** (**Err**)

**D**: ditto, the denominator

**C<sub>j</sub>**: predefined constant, where  $0 \leq j \leq \mathbf{B}$  (**#Cn**). The index  $j=0$  corresponds to no predefined constant present.

- The constants to try are *predefined* in **DATA** statements at lines 22-24. The first integer value is the number of constants predefined (6), the remaining string values are first the constants' *values* (which can be arbitrary, evaluable numeric expressions), then the constants' user-specified *names*.
- You can give the constants arbitrary names ("EulerGamma", "Pi") but you *must* include the name in parentheses if the name's an *expression* ("(Ln(2)\*Pi)") for proper output syntax.

- By default **B** is taken as **3**, i.e.:  $\pi, e, Ln(2)$  will be tried, but it can go up to 6 for extra constants  $\gamma, \varphi, \pi Ln(2)$ , and further, you can *extend* the identification capabilities by *adding your own*, see **Example 4** below for details. Running time is linear.

**P**: P<sup>th</sup>-power to raise the constant to, where P goes from **-L** (i.e., 1/P<sup>th</sup>-power) to **+L (#Pw)**, including 1, i.e.: the constant *as is*.

- Symbolic expressions not of the generic form above won't be recognized, though their value *will be* if it has *another*, compatible form. In any case, the returned expression will evaluate to the given value within the max. relative error specified. For example, attempting to recognize  $\pi + e$  fails and gives:

```
>CALL IDENTIFP(PI+EXP(1),S$,5,3,3,8,1E-9,0) @ S$
Sinh((661/284*Phi^2)^(1/2))
```

i.e.: we get  $Sinh \sqrt{\frac{661}{284} \varphi}$ , which agrees with  $\pi + e$  to 9 decimal places.

- Identification may fail if the specified value isn't *accurate enough*. Further, specifying a smaller max. error and/or additional constants, powers, roots, or functions might help, at the expense of increased running time.
- The routine which assembles the symbolic expression for output (*lines 52-62*) is very simple and doesn't try to further simplify it if possible. For instance **17.3205080757** will be identified as  $\sqrt{300}$ , not the simpler  $10\sqrt{3}$ .
- The identification process includes a *quick-exit* mechanism which helps to *greatly reduce* the running time but may occasionally return a less simple expression than is possible. For instance, **.643501108793** will be identified as **Asin(3/5)** instead of the equivalent but slightly simpler **Atan(3/4)**.
- If the (absolute) value to identify exceeds about **15** and **Atanh** is one of the functions to try, it's possible that it gets incorrectly identified as **Atanh(1)**, because **Tanh** equals **1** to 12 digits for arguments above **14.6+**, so **1** is considered the exact value for **Atanh** in that case. You must avoid specifying **Atanh** as a function to try in such cases or else put it in the last place.
- Values of some trigonometric functions of moderately sized arguments may fail to be recognized because the inverse function will only return values in certain *limited ranges* due to the *periodicity*. Thus **SIN(1)** will be recognized because **ASIN(SIN(1))** is computed as **1**, but **SIN(2)** won't be because **ASIN(SIN(2))** isn't returned as **2** by the **HP-71B**.
- The identification process is *very* computation-intensive and subject to combinatorial explosion. Thus it runs best under **Emu71**, a fast emulator where the timing will be 15-30 seconds at most, instead of in a physical **HP-71B**, where running times can exceed 1-2 hours in complex cases.

## Examples galore

1. Use the *IDENTIFY* subprogram to help compute the exact symbolic value of:

$$a) S = \int_0^1 \frac{\text{Tan}^{-1}(\sqrt{x^2+2})}{(x^2+1)\sqrt{x^2+2}} \cdot dx \quad \left( = \frac{5}{96} \pi^2 \right)$$

First, we'll set up some modes and variables to be used in these examples:

```
>DESTROY ALL @ DIM S$(80) @ RADIANS @ STD @ K=.00000001
```

Now for the integral's numerical computation and subsequent identification:

```
>INTEGRAL(0,1,K,ATN(SQR(IVAR^2+2))/SQR(IVAR^2+2)/(IVAR^2+1))
.514041895882
>CALL IDENTIFY(RES,S$) @ S$ -> 5/96*Pi^2
```

$$b) S = \int_0^{\pi} \frac{x \text{Sin}(x)}{1+\text{Cos}^2(x)} \cdot dx \quad \left( = \frac{\pi^2}{4} \right)$$

```
>INTEGRAL(0,PI,K,IVAR*SIN(IVAR)/(1+COS(IVAR)^2))
2.46740110022
>CALL IDENTIFY(RES,S$) @ S$ -> 1/4*Pi^2
```

$$c) S = \sum_{k=0}^{\infty} \frac{1}{64^k} \left( \frac{16}{6k+1} + \frac{8}{6k+2} - \frac{2}{6k+4} - \frac{1}{6k+5} \right) \quad \left( = \frac{32\pi}{3\sqrt{3}} \right)$$

Compute and identify the sum by running this code in some temporary file:

```
10 DESTROY ALL @ S=0 @ FOR I=0 TO 10
20 S=S+(16/(6*I+1)+8/(6*I+2)-2/(6*I+4)-1/(6*I+5))/64^I
30 NEXT I @ CALL IDENTIFY(S,S$) @ S$
(1024/27*Pi^2)^(1/2) , which simplifies to 32*Pi/(3*SQR(3))
```

$$d) \sum_{k=0}^{\infty} \frac{1}{64^k} \left( \frac{64}{(6k+1)^2} - \frac{160}{(6k+2)^2} - \frac{56}{(6k+3)^2} - \frac{40}{(6k+4)^2} + \frac{4}{(6k+5)^2} - \frac{1}{(6k+6)^2} \right) = 32\text{Ln}^2 2$$

Compute and identify the sum by running this code in some temporary file:

```
10 DESTROY ALL @ S=0 @ FOR I=0 TO 10
20 T=64/(6*I+1)^2-160/(6*I+2)^2-56/(6*I+3)^2
30 T=T-40/(6*I+4)^2+4/(6*I+5)^2-1/(6*I+6)^2 @ S=S+T/64^I
40 NEXT I @ CALL IDENTIFY(S,S$) @ S$
32*Ln(2)^2
```

2. Illustrate the difference between using the simpler call to **IDENTIFY** vs the full-fledged call to **IDENTIFP** while trying to identify these expressions:

$$a) S = \frac{1+\sqrt{5}}{4} \quad (= \operatorname{Sin} \frac{3\pi}{10} = \operatorname{Cos} \frac{\pi}{5} = \frac{\varphi}{2} \text{ (half the golden ratio) } )$$

```
>CALL IDENTIFY((1+SQR(5))/4,S$) @ S$
```

```
Sin(3/10*Pi)
```

```
>CALL IDENTIFP((1+SQR(5))/4,S$,5,3,3,4,1E-9,V) @ S$
```

```
1/2*Phi
```

The first call finds out the *sine* expression (instead of the slightly simpler *cosine* one because of the early termination feature), while the full-fledged call takes longer but does find the much simpler *golden ratio* relationship.

$$b) S = \sum_{k=1}^{\infty} \frac{1}{k^4} \quad (= \frac{\pi^4}{90} )$$

```
>S=0 @ FOR I=1000 TO 1 STEP -1 @ S=S+I^(-4) @ NEXT I
```

```
>CALL IDENTIFY(S,S$) @ S$
```

```
2143/1980
```

```
>CALL IDENTIFP(S,S$,3,4,3,4,1E-9,V) @ S$
```

```
1/90*Pi^4
```

This time the simpler call *fails* to correctly identify the sum, while the call to **IDENTIFP** *succeeds* when asked to search up to 4<sup>th</sup> powers.

$$c) x = \text{the root of } \sum_{k=1}^{\infty} \frac{k^k}{k!} x^k = \frac{1}{2}$$

Compute the root by running this code in some temporary file:

```
10 DESTROY ALL @ S=FNROOT(0,1/3,FNF(FVAR)-1/2) @ DISP S
20 DEF FNF(X) @ Y=0 @ K=1
30 T=(K*X)^K/FACT(K) @ IF Y+T#Y THEN Y=Y+T @ K=K+1 @ GOTO 30
40 FNF=Y
```

```
>RUN
```

```
.238843770192
```

```
>CALL IDENTIFY(S,S$) @ S$
```

```
(1/27/e)^(1/3) , which simplifies to  $x = \frac{1}{3 \sqrt[3]{e}}$ 
```

There's no need to issue the more complex call since the call to **IDENTIFY** *succeeded* in retrieving the correct symbolic expression for the root.



### 3. Show how to extend the functionality by adding new functions in order to recognize symbolic expressions of the form $N + \pi$ and $N - \pi$

We just need to enter a new **DATA** statement containing the proper definitions for *both* the new function *and its inverse*, which in this case will be:

```
31 DATA (Pi+()), (( )-Pi)
```

and we must also change line 26 **DATA 26, ( ) , ( ) , ...** to **DATA 28, ( ) , ( ) , ...** since we've added 2 new functions. Notice that the definitions are enclosed in parentheses (which are necessary for correct output syntax if the value is <0) and that their *argument* is represented by the *empty parentheses set*, ( ) .

Let's check the extended recognition capabilities by evaluating and identifying the following definite integral, this time using the front-end:

$$S = \int_0^1 \frac{x^4(1-x)^4}{1+x^2} \cdot dx \quad \left( = \frac{22}{7} - \pi \right)$$

```
>RUN
```

```
Value=INTEGRAL(0,1,1E-12,(IVAR*(1-IVAR))^4/(1+IVAR^2))
#Cn,Pw,Rt,Fn,Err=3,3,3,14,1E-9
1.26448926735E-3 identified as ((22/7)-Pi) (100%)
```

and *now* we can also identify  $\pi + e$ , which earlier we couldn't ! :

```
>CALL IDENTIFP(5.85987448205,S$,3,3,3,14,1E-9,0) @ S$
(Pi+(e))
```

### 4. Show how to extend the functionality by adding new constants

Let's extend the functionality by predefining an additional constant, "**Gamma (1/4)**", approximately **3.62560990822**. We just need to add its *value* and *name* to the appropriate **DATA** statements. In this case, we'll enter:

```
23 DATA 3.62560990822
25 DATA "Gamma (1/4) "
```

and we must also change the statement 22 **DATA 6,PI,...** to 22 **DATA 7,PI,...** since we've added *one* new constant. Let's check it out by identifying:

$$S = \int_0^{\frac{\pi}{2}} \sqrt{2\pi \sin^3(x)} \cdot dx \quad \left( = \frac{1}{6} \Gamma^2\left(\frac{1}{4}\right) \right)$$

```
>RUN
```

```
Value=INTEGRAL(0,PI/2,1E-10,SQR(2*PI*SIN(IVAR)^3))
#Cn,Pw,Rt,Fn,Err=7,3,3,4,1E-9
2.19084120111 identified as 1/6*Gamma(1/4)^2 (100%)
```

5. Find exact symbolic values for the examples given in the introduction

a) Compute  $S = \int_0^{\frac{\pi}{2}} x^2 \ln^2(2 \cos(x)) \cdot dx$  ( $= \frac{11}{1440} \pi^5$ )

A tough integral because of the *singularity*, we'll use *two* subintervals:

```
>S=INTEGRAL(0,3*PI/8,1E-12,(IVAR*LN(2*COS(IVAR)))^2)
>S=S+INTEGRAL(3*PI/8,PI/2,1E-12,(IVAR*LN(2*COS(IVAR)))^2)
>RUN
Value=RES
#Cn,Pw,Rt,Fn,Err=3,5,3,4,1E-9
2.33765036938 identified as 11/1440*Pi^5 (100%)
```

b) Simplify  $\frac{\sinh \frac{\pi}{4}}{\cosh \frac{\pi}{4} - \sinh \frac{\pi}{4}} + \frac{\cosh \frac{\pi}{4}}{\cosh \frac{\pi}{4} + \sinh \frac{\pi}{4}}$  ( $= \cosh \frac{\pi}{2}$ )

```
>RUN
Value=SINH(PI/4)/(COSH(PI/4)-SINH(PI/4))+COSH(PI/4)/(COSH(PI/4)+SINH(PI/4))
#Cn,Pw,Rt,Fn,Err=3,3,3,9,1E-9
2.50917847867 identified as Cosh(1/2*Pi) (100%)
```

c) Compute as an exact fraction  $\frac{1}{7} + \frac{2}{13} - \frac{3}{19} + \frac{1}{23}$  ( $= \frac{7249}{39767}$ )

```
>RUN
Value=1/7+2/13-3/19+1/23
#Cn,Pw,Rt,Fn,Err=0,0,0,0,1E-9
.182286820731 identified as 7249/39767 (100%)
```

d) Find an alternate symbolic form of  $\frac{\sqrt{3}-1}{2\sqrt{2}}$  ( $= \sin(15^\circ)$ )

```
>DEGREES @ RUN
Value=(-1+SQR(3))/2/SQR(2)
#Cn,Pw,Rt,Fn,Err=3,3,3,4,1E-9
.258819045103 identified as Sin(15) (100%)
```

e) Identify the limit  $\lim_{x \rightarrow 0} (1 + \sin(x))^{\cot(2x)}$  ( $= \sqrt{e}$ )

```
>RADIANS @ RUN
Value=(1+SIN(1E-7))^(1/TAN(2E-7))
#Cn,Pw,Rt,Fn,Err=3,3,3,4,1E-7
1.64872122948 identified as (e)^(1/2) (100%)
```

## 6. Test suite to demonstrate what's possible and help check new versions

| <i>Expression to symbolically evaluate</i>                            | <i>Computed value<br/>(Up. limit &amp; rel. error<br/>for INTEGRAL )</i> | <i>Identification<br/>parameters</i> | <i>Identified symbolic<br/>value</i> |
|---|--|--------------------------------------|--------------------------------------|
| $\int_0^1 x^4 (1-x)^4 . dx$   | 1.5873015873E-3<br>(K=1E-8)  | <i>default</i>                       | $\frac{1}{630}$                      |
| $\int_0^\infty e^{-x^2} . dx$   | .886226925453<br>(U=10, K=1E-10)   | <i>default</i>                       | $\frac{\sqrt{\pi}}{2}$               |
| $\int_0^\infty \frac{e^{-x} - e^{-\pi x}}{x} . dx$                    | 1.14472988575<br>(U=20, K=1E-10)   | <i>default</i>                       | $\text{Ln } \pi$                     |
| $\int_0^\infty \frac{1}{1+x^4} . dx$                                  | 1.11072073421<br>(U=1E3, K=1E-10)  | <i>default</i>                       | $\frac{\pi}{2\sqrt{2}}$              |
| $\int_0^1 \text{Ln } \Gamma(x) . dx$                                  | .918938533029<br>(K=1E-10, takes<br>very long)                           | <i>default</i>                       | $\text{Ln } \sqrt{2\pi}$             |
| $\int_0^{\frac{\pi}{2}} \text{Sin}(x) \text{Ln Sin}(x) . dx$          | -.306852819438<br>(K=1E-10)  | <i>default</i>                       | $\text{Ln } \frac{2}{e}$             |
| $\int_0^1 \text{Ln } \frac{1+x}{1-x} . dx$                            | 1.38629436094<br>(K=1E-10)   | <i>default</i>                       | $2 \text{Ln } 2$                     |
| $\int_0^1 \frac{1}{x} \text{Ln } \frac{1+x}{1-x} . dx$                | 2.4674011001<br>(K=1E-10, takes<br>very long)                            | <i>default</i>                       | $\frac{\pi^2}{4}$                    |
| $\int_0^{\frac{\pi}{2}} \frac{-\text{Ln Cos}(x)}{\text{Tan}(x)} . dx$ | .41123351671<br>(K=1E-10)  | <i>default</i>                       | $\frac{\pi^2}{24}$                   |

| <i>Expression to symbolically evaluate</i>                                    | <i>Computed value<br/>(Up. limit &amp; rel. error<br/>for INTEGRAL )</i> | <i>Identification<br/>parameters</i> | <i>Identified symbolic<br/>value</i>   |
|---|--|--------------------------------------|--|
| $\int_0^{\infty} \frac{x^4}{(x^4 + x^2 + 1)^3} \cdot dx$                      | 3.77874867484E-2<br>(U=30, K=1E-10)                                      | default                              | $\frac{\pi}{48\sqrt{3}}$               |
| $\int_0^{\infty} \frac{x^3}{(x^4 + 7x^2 + 1)^{5/2}} \cdot dx$                 | 8.23045267136E-3<br>(U=60, K=1E-10)                                      | default                              | $\frac{2}{243}$                        |
| $\int_0^{\infty} \frac{1}{(x^2 + 1)(x^{1.776} + 1)} \cdot dx$                 | .785398163226<br>(U=2000, K=1E-7)  | default                              | $\frac{\pi}{4}$                        |
| $\int_0^{\frac{\pi}{2}} \frac{1}{(1 + \tan(x)^{2.007})} \cdot dx$             | .785398163398<br>(K=1E-10)   | default                              | $\frac{\pi}{4}$                        |
| $\sum_{k=1}^{\infty} \frac{(-1)^{(k+1)}}{(2k-1)^5}$                           | .996157828071<br>(U=76)  | 3, 5, 3, 4,<br>1E-9                  | $\frac{5\pi^5}{1536}$                  |
| $\int_0^{\infty} \frac{x}{e^x - 1} \cdot dx$                                  | 1.64493406686<br>(U=30, K=1E-10)   | default                              | $\frac{\pi^2}{6}$                      |
| $\int_0^{\infty} \frac{\ln^2(x)}{e^x} - \frac{x}{e^x - 1} \cdot dx$           | .333177923808<br>(subintervals)  | 5, 3, 3, 4,<br>1E-9                  | $\gamma^2$                             |
| $\sum_{k=1}^{\infty} \frac{-1}{k 10^k}$                                       | -.105360515657<br>(U=10)   | 3, 3, 3, 4,<br>1E-12                 | $\ln \frac{9}{10}$                     |
| $\frac{1}{2} \sqrt{2 - \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2}}}}}}$ | 2.45412285246E-2<br>(in DEGREES)   | default                              | $\sin\left(\frac{45^\circ}{32}\right)$ |
| $\sum_{k=0}^{\infty} \frac{1}{(2k+1)4^k}$                                     | 1.09861228867<br>(U=20)  | default                              | $\ln 3$                                |

| <i>Expression to symbolically evaluate</i>  | <i>Computed value<br/>(Up. limit &amp; rel. error<br/>for INTEGRAL )</i> | <i>Identification<br/>parameters</i> | <i>Identified symbolic<br/>value</i>          |
|---|--|--------------------------------------|---|
| $\int_0^{2\pi} \frac{\cos^2(3x)}{5-4\cos(2x)} \cdot dx$   | 1.1780972451<br>(K=1E-10)  | default                              | $\frac{3\pi}{8}$                              |
| $\int_0^{\infty} \frac{x}{\sinh(x)} \cdot dx$   | 2.46740110027<br>(U=30, K=1E-10)   | default                              | $\frac{\pi^2}{4}$                             |
| $\int_0^{\frac{\pi}{2}} \frac{1}{9+4\sqrt{5}\cos(x)} \cdot dx$  | .111341014342<br>(K=1E-10)   | default                              | $\sin^{-1}\left(\frac{1}{9}\right)$           |
| $\int_0^{\infty} \frac{e^{-x^2} - e^{-x}}{x} \cdot dx$  | .288607832453<br>(U=30, K=1E-10)   | 5, 3, 3, 4,<br>1E-9                  | $\frac{\gamma}{2}$                            |
| $\int_0^{\infty} \frac{\sin(\pi x)}{\sinh\left(\frac{\pi}{2}x\right)} \cdot dx$                                 | .996272076217<br>(U=30, K=1E-10)   | 3, 3, 3, 9,<br>1E-9                  | $\tanh \pi$                                   |
| $\sum_{k=1}^{\infty} \frac{(1+\sqrt{5})^{2k-1}}{(2k-1)!}$   | 12.6971007574<br>(U=11)  | 5, 3, 3, 9,<br>1E-9                  | $\sinh(2\varphi)$                             |
| $\sum_{k=1}^{\infty} \frac{(5\cos^2\left(\frac{\pi}{5}\right))^{2k-1}}{(2k-1)!}$                                | 13.1702053741<br>(U=13)  | 5, 3, 3, 9,<br>1E-9                  | $\sinh\left(\frac{5}{4}\varphi^2\right)$      |
| $\int_0^{\frac{\pi}{2}} \frac{1}{\sin^2(x)} \cdot dx$<br>$\int_0^{\infty} \frac{e^{-x^2} - e^{-x}}{x} \cdot dx$ | 3.36816833521<br>(U=30, K=1E-10<br>para ambas<br>integrales)             | 5, 3, 3, 12,<br>1E-9                 | $\frac{1}{\tan\left(\frac{\gamma}{2}\right)}$ |

**Note:** If you don't have a Math ROM, simply identify the given *numeric* values

## Exercise 4U

Extend the functionality by adding *a new function*,  $\Gamma^2(x)$ , and its inverse. Check your implementation by computing and identifying these expressions:

a)  $\int_0^{\frac{\pi}{2}} \sqrt{\frac{\pi}{2} \sin(x)} \cdot dx$       b)  $\sqrt[3]{2} \sqrt{\frac{\pi}{3} \Gamma\left(\frac{1}{6}\right)}$

**Solution:**

```

1) Include the function and its inverse in a DATA statement. The Math ROM
   does provide  $\Gamma(x)$  as GAMMA(x) but not its inverse, so we must define that
   ourselves and include our own user-defined function FNG(x) instead:
27 DATA Gamma (^2, FNG())
2) Include the proper user-defined function's definition to compute the required
   inverse function, FNG, within the IDENTIFF subprogram:
21 DEF FNG(X)=FNROOT(.001,1,GAMMA(FVAR)^2-X)
3) Update the count at line 26 DATA 26, . . . to 26 DATA 28, . . . , to include
   the two new functions available now.
4) Compute and identify both expressions:
>RUN
Value=2^(1/3)*SQR(PI/3)*GAMMA(1/6)
#Cn,Pw,Rt,Fn,Err=3,3,3,6,1E-9
7.17671167272 identified as Gamma(1/3)^2 (100%)
>RUN
Value=INTEGRAL(0,PI/2,1E-10,SQR(PI/2*SIN(IVAR)))
#Cn,Pw,Rt,Fn,Err=3,3,3,6,1E-9
1.50164609469 identified as Gamma(3/4)^2 (100%)

```

### “Further reading”

As is, these simple routines can certainly identify a useful variety of numerical results, providing the simplest approximate expression when exact identification is not possible and, when running in a fast platform, their capabilities can be greatly expanded by adding extra predefined constants and functions. However, there’s a three-pronged problem with this approach: (1) the *exponential explosion* of cases to try, (2) the increasing need for *more precision* to discriminate the correct result among spurious fits, and (3) the *limited variety* of recognizable expressions.

Problems (1) and (3) can be tamed with *integer relation algorithms*, such as **LLL** and **PSLQ**. However, any implementation which must deal with non-trivial cases absolutely requires multiprecision. For instance, recognizing  $\sqrt[5]{5} - \sqrt[4]{4}$  needs *from 50- to 100-digit precision*, depending on the algorithm, and lots of CPU. Tricky !