Testing COSY's INSRF

George F. Corliss
Marquette University

October 28, 2003

Abstract

Execution-based tests of COSY interval enclosures for intrinsic functions for various values of INSRF reveal no violations of containment, even for INSRF 1;

1 Context

COSY uses a procedure INSRF to control the amount of outward rounding applied to the result of evaluation of native intrinsic functions. According to the COSY Infinity Version 1.8 Programming Manual, MSUHEP-20703, May 2001,

> Procedure INSRF (1 argument) sets the factor f. $f\epsilon$ is the outward rounding constant for interval intrinsic functions, where ϵ is the software determined machine error, and is the outward rounding constant for the other interval rounding including the binary operations. By default f is 10. The last specified f is kept independent of INSRND.

We report execution-based testing of COSY's intrinsic functions for various values of INSRF down to 1. Even at INSRF 1, we see no violations of containment.

2 Test methods

The tests follow those developed by Yu and Corliss and reported in George F. Corliss and Jun Yu, Testing COSY's Interval and Taylor Model Arithmetic to appear in Springer Lectures Notes on Computer Science (LNCS), *Numerical Software with Result Verification: Platforms, Algorithms, Applications in Engineering, Physics, and Economics,* R. Alt, A. Frommer, R. B. Kearfott, and W. Luther, eds.

The tests were run on a Toshiba Satellite 4090XDVD with an Intel Celeron at 400 Mhz, 128 MB RAM, running Windows 98.

I reproduced Jun's tests, minus the tests of +, -, *, and /. That left 635 tests. The code is attached as 3arith_01.fox. The test cases are specified in the attached COSY_inp.txt. I ran similar tests for INSRF 1, 2, ..., 10, 20, and 100.

2.1 Validating the action of INSRF

How do I know that the INSRF call has the intended effect?

In each case, I called INSRF n; during the initialization (see the codes in 3arith_01.fox and rand_chal.fox). Here are (inexactly rounded) results

```
 For INSRF 10;
   1060020      1060  (= SIN)       2
 [   3.141592653589792   3.141592653589794 ]
 [ -0.2542074623718143E-14  0.3231085104332683E-14 ]

For INSRF 1;
   1060020      1060         2
 [   3.141592653589792   3.141592653589794 ]
 [ -0.2542074623718138E-14  0.3231085104332677E-14 ]
```

We can see that the results for INSRF 1; are slightly tighter. Hence, I am reasonably certain that had the intended effect.

2.2 Testing methodology

The process was the same as before. I ran COSY and wrote in a binary form the endpoints passed to COSY's interval constructor and the interval

returned by COSY's evaluation. That binary file was read by Maple. Maple generated 10 equally spaced points in the argument interval and evaluated the same symbolic expression in high precision arithmetic. This time I used 200 digits.

Let X1 be a challenge interval written in ASCII test in the file COSY_inp.txt.

The endpoints of X1 are read by the standard input routines of COSY (probably rounded to nearest). Let X2 be the internal binary representation of X1. X2 is exact with respect to itself, but it is rounded from X1.

Let X3 := INTV(X2); (figuratively) in COSY. INTV rounds X2 outward.

Let F3 := intrinsic_function(X3); as evaluated by COSY.

Write the exact binary values of X2 (not X3) and F3 to a file.

Maple reads the binary values for X2 and F3.

Maple computes 10 equally-spaced points x in X2, and evaluates $f :=$ intrinsic_function(x), using 200 decimal digit arithmetic.

In Maple, any f not contained in F3 is a violation of containment.

In NO case was Maple's evaluation outside the COSY enclosure. We detected no containment failures, even for INSRF 1;

As we agreed during the 2002 tests, the interval COSY intrinsics see (output from the interval constructor) is slightly wider than the interval Maple sees (arguments passed to INTV).

3 Running the tests

I assume COSY is properly installed and in the PATH.

Navigate to the directory COSYoct.

The COSY program for Jun's tests is 3arith_01.fox. The input file containing the test cases is in 1Arith_inp/2arith_inpA.txt. Input files controlling the value for INSRF are in 1Insrf_vals/insrf10.txt (for INSRF 10;) and similarly named files.

In a DOS window, type `runtest A 10`. See the script runtest.bat to see how the various input and output files are copied.

That starts COSY. In the COSY window, type `3arith_01`.

After the COSY program finishes, in the DOS window, press [Enter] to complete the batch run. A binary version of the results is copied to 2Bin_results/4arith_binA_10.txt and to the subdirectory Maple. An ASCII text version (output is approximately rounded) is copied to 2Text_results/4arith_resA_10.txt.

Navigate to the Maple directory.

Open the Maple worksheet test_cosy.mws.

If necessary, edit the file name `readDataFile` to match the test case you are running. If the COSY run was done on a Unix machine, in the Maple code, you'll have to replace `readCOSYdouble.mpl` by `readFORTdouble.mpl`. These two files read big endian and little endian binary files.

Execute each Maple code block. You should see a final report on the number of test cases and the number of errors found.

4 Random tests

I re-ran the randomly generated tests, again removing the tests of +, -, *, and /. The settings I used yielded 367,730 tests. On my machine, that took COSY about 6-8 minutes, and about 6-7 hours to check with Maple. The code is attached as rand_chal.fox. I ran the tests for INSRF 1.

I saw no violations of containment.

5 What did we see in Summer 2002?

At one point in the tests of the Summer 2002, Yu and Corliss claimed to see INSRF-related violations of containment. The test driver `3arith_02.fox` replicates those results.

The difference is in what challenge argument interval is passed to Maple.

In the outline of the test given above, we write to the binary result file the exact endpoints of X2, the values which are passed **into** the COSY interval constructor INTV. In `3arith_02.fox`, we instead pass the exact endpoints of X3, the **results** from INTV.

If we ask Maple to check against the **results** from INTV, even at INSRF 100; the test

$$\cos([-1.570796326794897, -1.570796326794897])$$

violates containment.

In my opinion, that shows that mathematically, the COSY cos intrinsic function violates containment for a particular argument whose exact value is close to [-1.570796326794897, -1.570796326794897].

Table 1: Relative Excess Width - Frequency

INSRF =	1	2	3	4	5	6	7	8	9	10	20	100
0	33	33	33	33	33	33	33	33	33	33	33	33
1	17	0	0	0	0	0	0	0	0	0	0	0
2	45	31	14	14	14	14	14	14	14	14	14	14
4	219	135	74	74	52	35	35	35	35	35	35	35
8	144	250	273	233	153	94	93	93	91	91	52	52
16	17	26	79	113	210	242	232	229	218	157	40	0
32	8	8	10	16	20	62	69	72	85	146	220	0
64	8	8	8	8	9	11	15	15	15	14	89	0
128	7	7	7	7	7	7	6	6	6	7	14	46
256	5	5	5	5	5	5	6	6	6	6	5	271
512	12	12	12	12	12	12	12	12	12	12	13	64
More	89	89	89	89	89	89	89	89	89	89	89	89
Total	604	604	604	604	604	604	604	604	604	604	604	604
Out of	635	635	635	635	635	635	635	635	635	635	635	635

At the same time, I accept the argument of the COSY authors that a package has a right to define its own interface. It corresponds to saying, "COSY, form this into an interval and take its cos." In that interpretation, I have been unable to construct an example that exposes the mathematical flaw I believe is there.

In fact, this example precisely shows the advantage of the COSY approach of rounding its intervals outward slightly in INTV.

6 Excess width

COSY uses a default INSRF 10; while even INSRF 1; showed no violations of containment in these tests.

That means that with INSRF > 1, COSY's enclosure interval is wider that it needs to be. The program 3arith_10.fox and the Maple worksheet wid_test_cosy.mws explore that. Table 1 summarizes those findings. Within Maple, given COSY's enclosing interval and the largest and smallest of the 10 evaluations Maple has done, we estimate the number of ULP's at each end the COSY enclosure is wider than the Maple samples.

Figure 1 suggests four excess ULP's on the left and three on the right. I

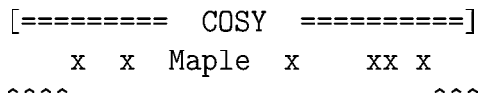

Figure 1: Excess width

compute ULP's in Maple by

```
ULPs := proc (a, b)
   local rewU:
   rewU := abs(b - a);
   if (a = 0) then
      return ( rewU * 2^1022 );
   else
      return ( (rewU / abs(a)) * 2^52 );
   end if:
end proc:
```

That is not a perfect measure, but it gives some indication.

In Table 1, the row labels mean 0: No excess width; 1: 1 ULP; 2: 2 ULP's; 4: 3 or 4 ULP's; 8: 5 - 8 ULP's; etc.

Table 1 shows the frequencies. For each INSRF value (columns), how many of the test cases exhibited that range of ULP's of excess width? You see that for higher values of INSRF, the median frequency moves toward greater overestimation.

Table 2 is cumulative: 0: No excess width; 1: 0 - 1 ULP; 2: 0 - 2 ULP's; 4: 0 - 4 ULP's; 8: 0 - 8 ULP's; etc.

7 Conclusions

The feature INSRF and its default value of 10 are intended to compensate for possible inaccuracies in native evaluation of intrinsic functions on which COSY depends. In general, there is no accurance of the quality of the underlying intrinsic function evaluations, so COSY must be conservative.

The tests reported here suggest that at least in the environment tested, it may be possible for COSY safely to use a smaller default for INSRF, yielding slightly tighter enclosures.

Table 2: Cummulative Relative Excess Width

INSRF =	1	2	3	4	5	6	7	8	9	10	20	100
0	33	33	33	33	33	33	33	33	33	33	33	33
1	50	33	33	33	33	33	33	33	33	33	33	33
2	95	64	47	47	47	47	47	47	47	47	47	47
4	314	199	121	121	99	82	82	82	82	82	82	82
8	458	449	394	354	252	176	175	175	173	173	134	134
16	475	475	473	467	462	418	407	404	391	330	174	134
32	483	483	483	483	482	480	476	476	476	476	394	134
64	491	491	491	491	491	491	491	491	491	490	483	134
128	498	498	498	498	498	498	497	497	497	497	497	180
256	503	503	503	503	503	503	503	503	503	503	502	451
512	515	515	515	515	515	515	515	515	515	515	515	515
More	604	604	604	604	604	604	604	604	604	604	604	604
Out of	635	635	635	635	635	635	635	635	635	635	635	635