

## 1. Elementary Functions : Hardware vs Software Manufacturers

Computing the elementary functions is a difficult, highly technical task. Intel, AMD, Sun, and other hardware manufacturers have spent a lot of time and money on research to ensure that their floating point processors give the correct answers. Intel has gone so far as to hire PhDs in formal proof theory to develop proofs of the correctness of their implementations. As a result, it is reasonable to assume that the elementary functions in floating point processors are correct, and adhere to the IEEE Standard.

Such an assumption cannot be made of the software that uses these processors. You cannot throw any old argument  $x$  at these hardware functions : you must reduce  $x$  to the range specified by the manufacturer. This is where the software manufacturer can really mess things up. If they write software and ignore or are ignorant of argument reduction then their software will give the wrong answers.

For example,

- MATLAB 6.5 gives  $\sin(10^{22}) = -0.8522008497671888$ ,
- MAPLE 8 gives  $\sin(10^{22}) = -0.9873536182$ ,
- O-MATRIX 5.8 gives  $\sin(10^{22}) = -0.7340815352961015$ .
- O-MATRIX 6.0 gives  $\sin(10^{22}) = -0.8522008497671888$ .

Which of these, if any, is correct?

In fact MATLAB's answer is correct, but it was not always so. Although the earliest versions of MATLAB were written in Fortran, subsequent versions were written in C, and they used Microsoft's C compiler to generate machine code. When it was pointed out that some of their elementary functions gave wrong answers they investigated and found that Microsoft's compiler was not reducing the arguments in keeping with the hardware specifications. They then decided to write their own code.<sup>1</sup>

Numbers in the IEEE double-precision floating point number system have a range  $10^{-308} \leq |x| \leq 10^{308}$ . However, many of the elementary functions found in software systems such as MATLAB do not work properly over this range. For example MATLAB 6.5 gives  $\sin(1.0d22) = -0.852200849767189$ , which is correct value rounded to 15 digits, but the value of  $\sin(1.0d23) = -0.324053937643003$ , is not correct. The high-precision *PariGP* system gives the correct answer  $\sin(10^{23}) = 0.701140639861078\dots$  Not only are all MATLAB's digits wrong, the sign is wrong also. The same is true of MATLAB's  $\cos(x)$  and  $\tan(x)$  functions. Figures 1 and 2 show the relative errors in  $\sin(x)$  and  $\cos(x)$  for MATLAB 6.5 and O-MATRIX 5.8. MATLAB's functions give wrong results for  $x > 10^{22}$ , while O-MATRIX 5.8 is worse with wrong results for  $x > 10^{10}$ . O-MATRIX 6.0 is better, with wrong results for  $x > 10^{22}$ . It appears that O-MATRIX 6.0 has been upgraded to match MATLAB's accuracy.

Both of their  $\cos(x)$  functions exhibit a curious phenomenon : a huge error at  $x = 10^{29}$ .

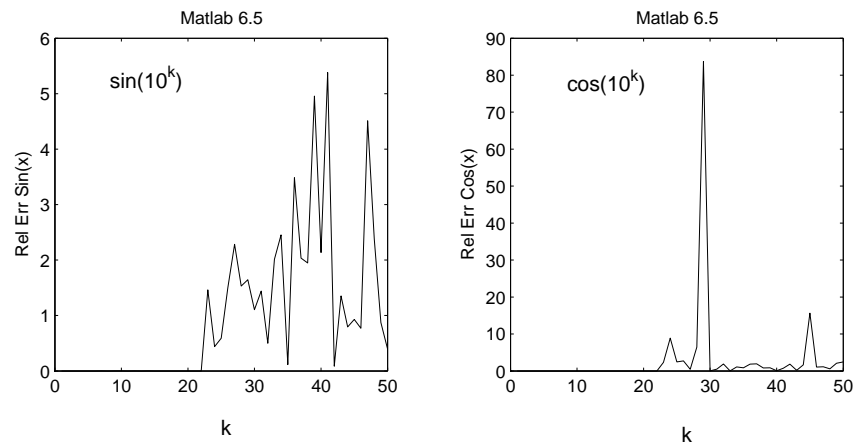
Why does MATLAB get the wrong values for  $x > 10^{22}$  ? The simple answer is that much higher precision is needed when computing  $\sin(x)$ , etc., for large arguments  $x$ . MATLAB would say that implementing  $\sin(x)$  to handle large arguments would make the function intolerably slow, and that most people use arguments that are much smaller than  $x = 10^{22}$ . This may be true, but the least they (and others) should do is check for arguments that are too big and then issue a warning to the user.

The main reason why hardware manufacturers get it right, and software manufacturers do not, seems to be this : *you cannot send out a corrective patch to 10 million Pentium 4 owners.*<sup>2</sup> Many software companies, without a hint of shame, charge for upgrades which are often just patches of errors in previous versions.

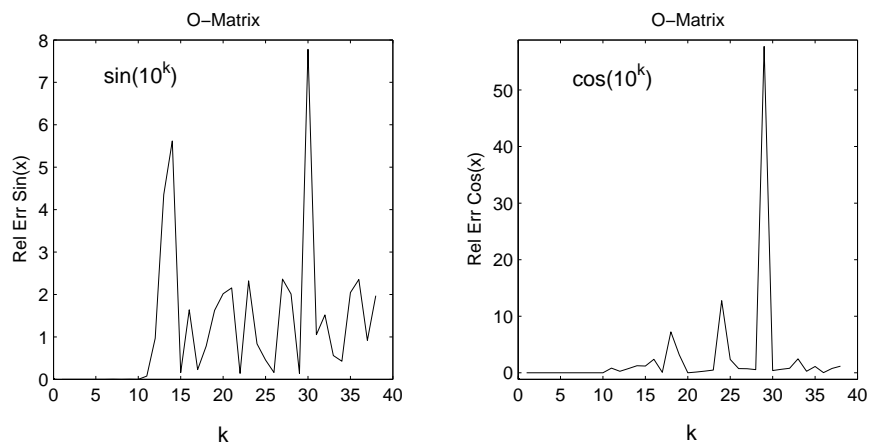
The correct values for Sin and Cos are shown in Table 1. These were obtained using *PariGP* but took 10 times longer to calculate than MATLAB. Although MATLAB and O-MATRIX give wrong answers for large arguments, we must be thankful for small mercies : at least MATLAB and O-MATRIX did not waste much time giving the wrong answers.

<sup>1</sup>See Cleve Moler's article on this subject in MATLAB's *Cleve's Corner*, Winter 2002.

<sup>2</sup>In 1995 Intel were forced to offer a replacement to 1 million people when the infamous FDIV bug was found in its Pentium I processor. Their accountants had to set aside \$450 million to cover this.



**Figure 1** : Relative Error in MATLAB'S SIN and COS functions



**Figure 2** : Relative Error in O-Matrix 5.8's SIN and COS functions

To see how difficult it is to calculate Sin for large arguments see Ng's excellent paper at <http://www.derekroconnor.net/Software/Ng--ArgReduction.pdf>

Table 1:  $\sin(x)$  and  $\cos(x)$  for Large Arguments  $x$ 

$k$	$\sin(x = 10^k)$	$\cos(x = 10^k)$
0	+0.8414709848078965066525023216	+0.5403023058681397174009366074
1	-0.5440211108893698134047476618	-0.8390715290764524522588639478
2	-0.5063656411097587936565576104	+0.8623188722876839341019385139
3	+0.8268795405320025602558874291	+0.5623790762907029910782492266
4	-0.3056143888882521413609100352	-0.9521553682590148512403867607
5	+0.0357487979720165093164705007	-0.9993608074382124518911354141
6	-0.3499935021712929521176524868	+0.9367521275331447869385325351
7	+0.4205477931907824912985065897	-0.9072703861817395611617127509
8	+0.9316390271097260080275166536	-0.3633850893556905538723753525
9	+0.5458434494486995642443872709	+0.8378871813639023343897756435
10	-0.4875060250875106915277942943	+0.8731196226768560011761913453
11	+0.9286936604965919528572271642	+0.3708477921647111590126646370
12	-0.6112387023768894981920204153	+0.7914463018528902700537662141
13	-0.2888852948175251222627810460	+0.9573637169008399352820267057
14	-0.2094083074964523026947459958	-0.9778282879685324783350041069
15	+0.8582727931702358355238863908	-0.5131937377869702522345361364
16	+0.7796880066069787502355279540	-0.6261681981330861717627850736
17	-0.4645301048353726961545241140	-0.8855573282976306850496332070
18	-0.9929693207404050762095530172	+0.1183719902187107326119543333
19	-0.9270631660486503852341222897	-0.3749051695507178301532205460
20	-0.6452512852657808442058117113	+0.7639704044417283004001468027
21	-0.6671201770718048469651957581	+0.7449501119831339058586523072
22	-0.8522008497671888017727058937	+0.5232147853951389454975944734
23	+0.7011406398610784694692418305	-0.7130230032300482952160866229
24	-0.9964722291025832583758359242	+0.0839231590642822663624958335
25	-0.7447898487448297999022969598	-0.6672990942648233120970585698
26	+0.8534551480056754104710936355	-0.5211662981646172019029260309
27	+0.7180634961391176660795156521	-0.6959775969903538128251834949
28	-0.9878291038355987467552561610	-0.1555431181870733532435700137
29	+0.9999592845984057657235009128	-0.0090238099184624657752512848
30	-0.0901169019121380580303864289	-0.9959311944053957023942485880

Calculated using PariGP 2.2.7 with real precision = 28 significant digits