Intel Math Libraries Testing and Validation Methodologies

Evgeny Gvozdev, Intel® Corporation,
evgeny.gvozdev@intel.com

International Conference Dedicated to the 100th Anniversary of the Birthday of Sergei L. Sobolev
Novosibirsk
October 8, 2008

Content

- A little bit history
- Requirements
- Design Decisions
- Results
- Summary

A Little Bit History

- Story started about 10 years ago
- Intel® C Compiler
- LIBM: **fast** and **accurate** elementary math functions
- Test suites:
  - Internally used
  - Elefunt* (Elementary Function Tests)[3]
  - UCB* (Univ. California Berkeley Tests)
    http://www.netlib.org/fp/ucbtest.tgz

A Little Bit History

- Subset of functions, precisions
- Limited functionality
- Hard to extend
- Mostly random data rather than carefully selected
- Very limited special argument tests
- Black-box. Not-implementation/algorithm specific

There was no comprehensive suite for testing and verification of Elementary Math functions

Requirements

- Tests
  - Accuracy
  - Performance
  - Standards compliance
  - Function properties
  - Identities
- Real and complex
  - Float
  - Double
  - Long double (80 bits)
  - Quad (128 bits)

Requirements

- Portable (OS, compiler, processor, endian)
- Usable for testing and verification
- Expandable for testing new functions
- Operates in different FP environments
- Tunable interface, workload, output verboseness
- Flexible, extendable input data

Need a powerful, portable, extendable, tunable Tool

Design Decisions.
White/black box, data driven, extendable

- Investigated each function to test
- Possible implementation algorithms and their hard paths
- Possible different algorithm paths for performance testing
- Binary arithmetic specific cases
- ~140 MB of data
- Thousands fixedvector arguments per function
- Tens/hundreds of thousand run-time generated
- Ability to set up custom data

Design Decisions.
Data driven, pre-calculated

- Initial set
  - Random, Min/Max, Denorms, Integers, …
- Hard to round cases (Table Maker's Dilemma)[5]
- Other hard to process data
  - close to n*Pi
  - overflow/underflow thresholds
  - common Table implementation methods boundaries
- Standards conformance data
- All worst data from many various LIBMs testing collected

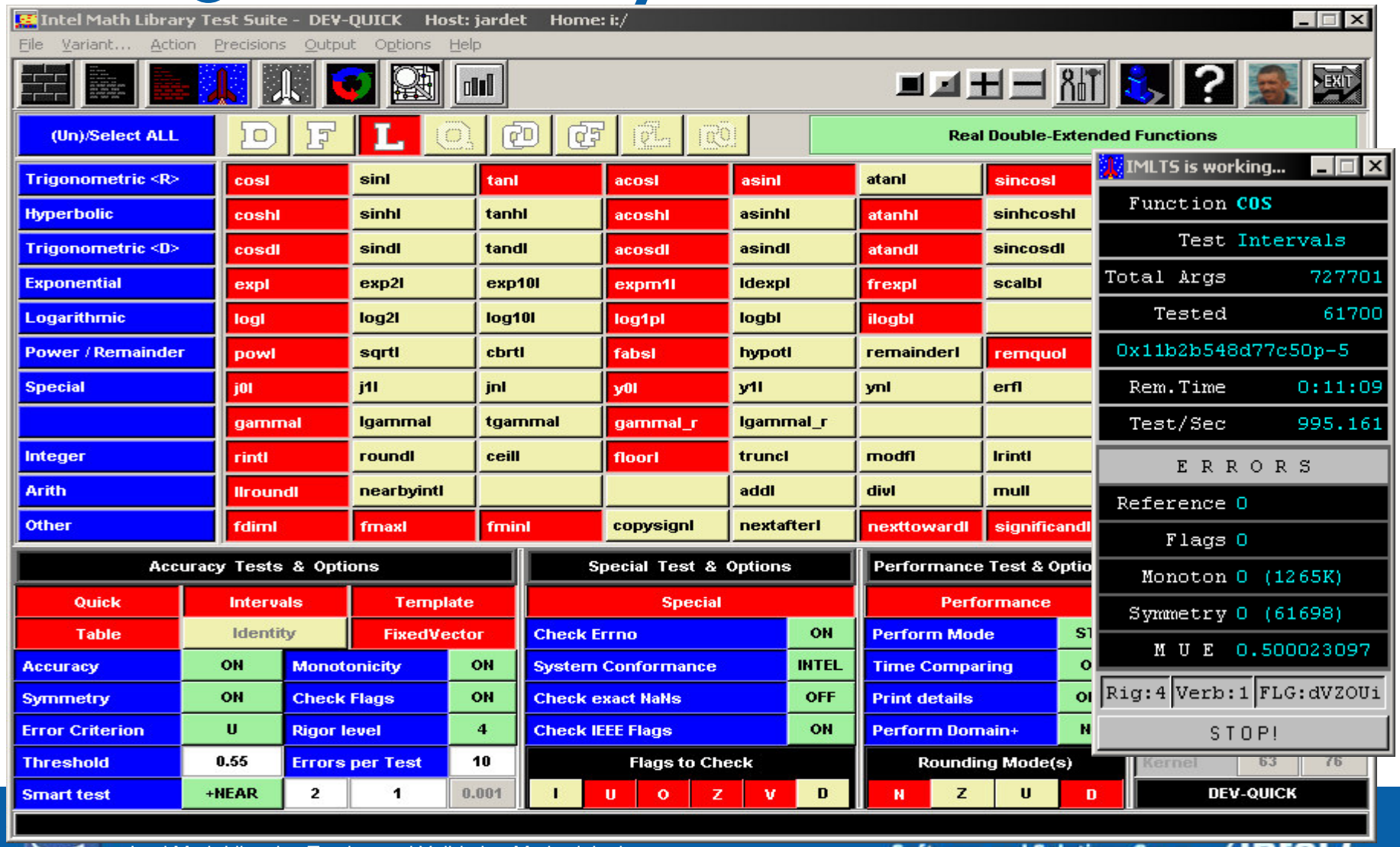
Design Decisions.
Data driven, run-time generated, tunable

- Random Intervals
- Neighborhood of specific points
- Test specified interval exhaustively
- Table-lookup (exhaustive n most significant bits of mantissa)
- Direct manipulations of bits in specific mantissa range
- N perturbations, K bits changed in mantissa bits range B1:B0
- N = 3, K = 2, B1 = 5, B0 = 2
- S EXP …MMM**0101**MM
- S EXP …MMM**1010**MM
- S EXP …MMM**0110**MM

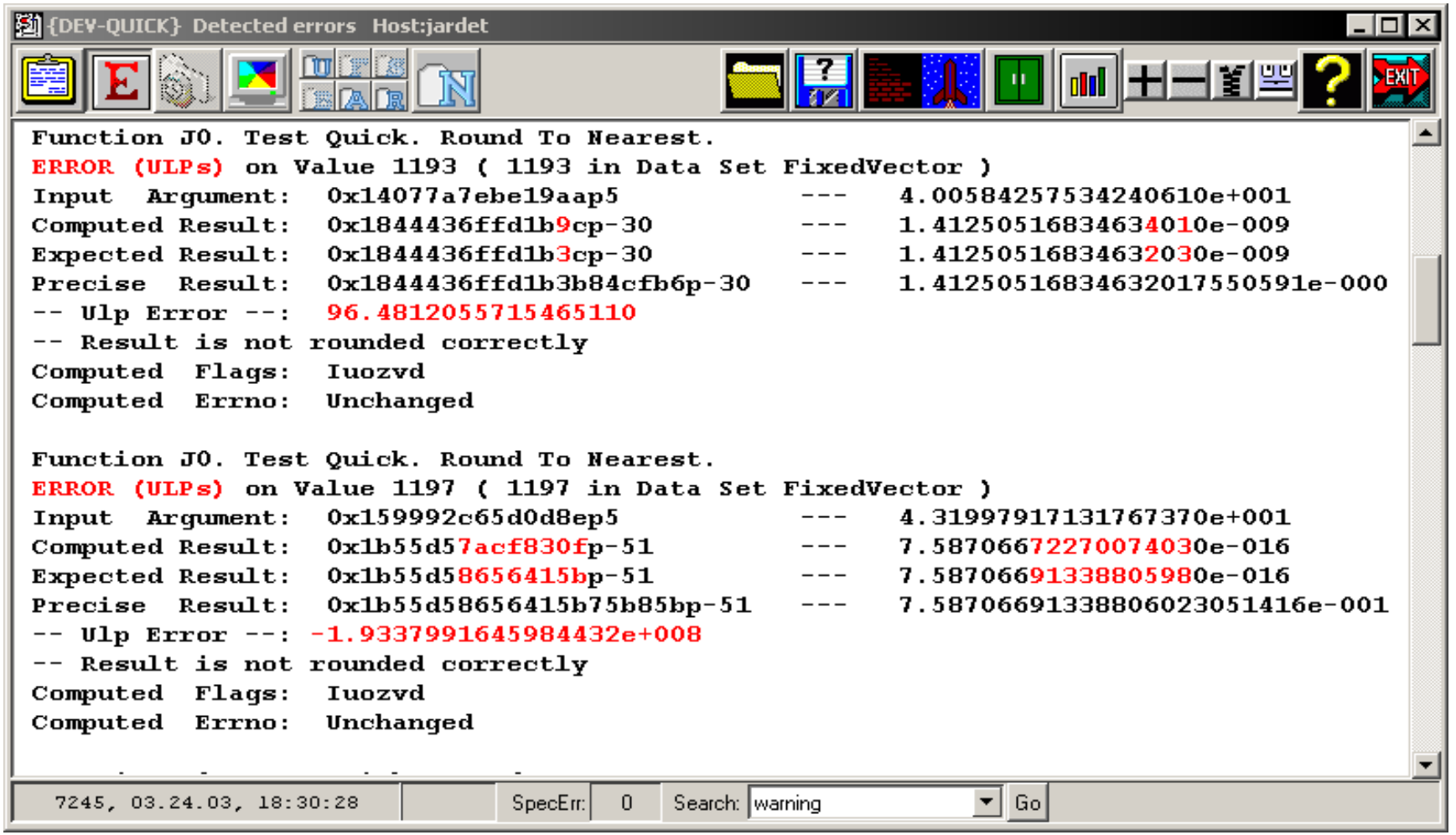
Design Decisions.
convenient, developers/managers usable design

- Command line interface with GUI on top
- Text input, text and .csv output, easy to pre/post process
  - 1.0, 3ff00000, 0x1p0
- Tunable configuration
  - Suite common config file - *function specific script* - **run option**
- Test specific argument and interval from command line
- Exclude specific intervals from testing
- Testing arbitrary (up to ~300 bits) precision "kernels"
- On the fly performance comparison of 2 libraries

Design Decisions.
Intel® Math Library Test Suite GUI



Design Decisions.
GUI Errors file



Design Decisions.
Text Summary file

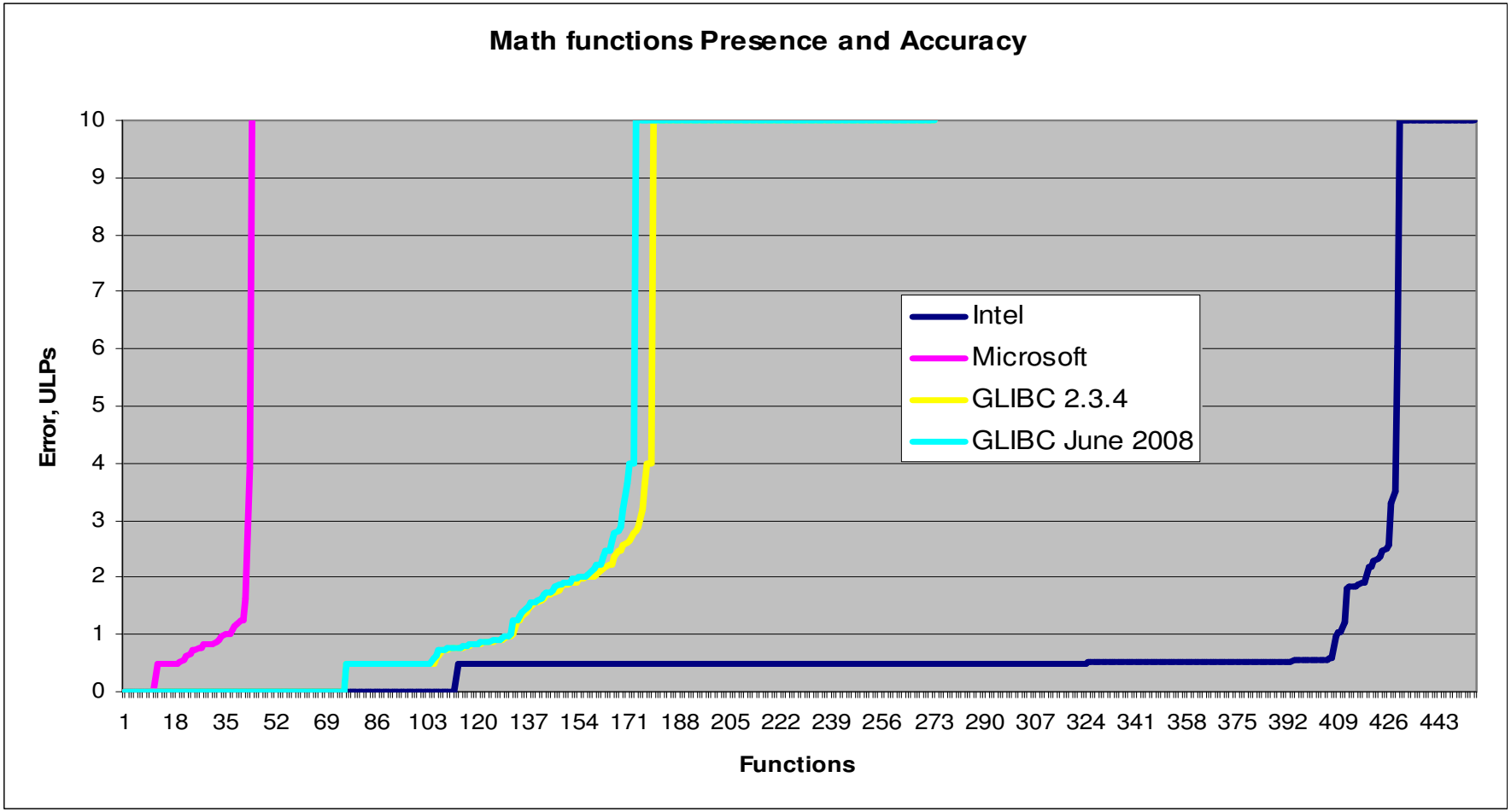
Function	PASS/ FAIL	Max Ulp error	ULPs	Error Counts Mo	Sy	Fl	IUOZVD	E	B	Roun ding	Arguments Spec/Base
SIN	PASS	0.515082	0	P	P	P	******	*	*	Near	42/31102
COS	PASS	0.518672	0	P	P	P	******	*	*	Near	40/30380
TAN	PASS	0.541852	0	P	P	P	******	*	*	Near	42/25031
ASIN	FAIL	0.535745	0	P	P	99	N*****	*	*	Near	42/27973
ACOS	FAIL	0.531348	0	P	-	76	B****N	*	*	Near	41/29519
ACOS	FAIL	0.531348	0	P	-	40	B****N	*	*	Near	41/1300
ACOS	FAIL	0.999864	0	P	-	41	S****N	*	B	+Inf	41/1300
ACOS	FAIL	1.000545	0	P	-	40	N****N	*	*	-Inf	41/1300
ACOS	FAIL	1.002306	0	P	-	40	N****N	*	*	Zero	41/1300

White/black box, data driven, convenient, developers/managers usable design

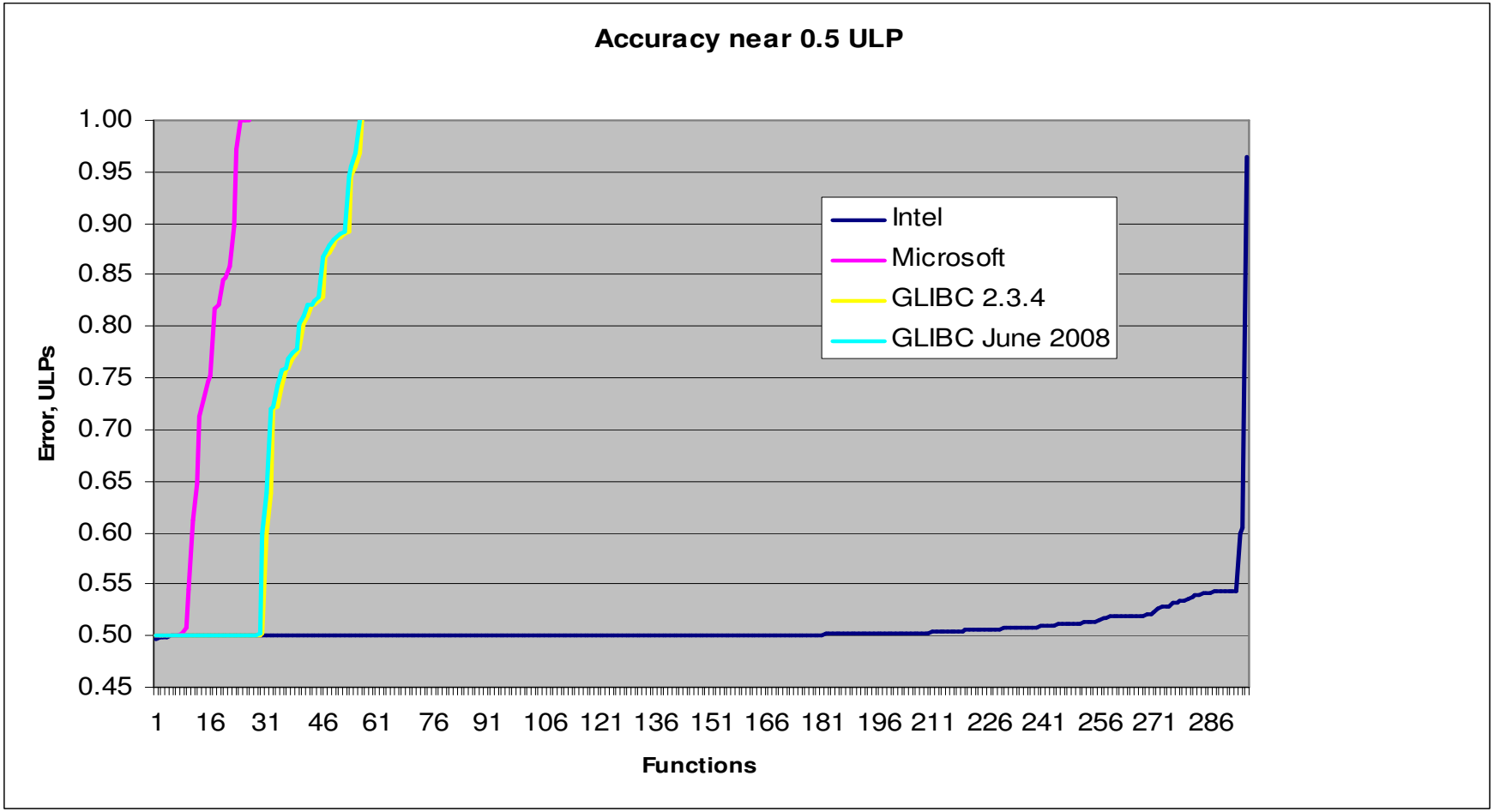
Results.

- Linux*/Microsoft* Windows*/HP-UX*/Sun* Solaris* OS
- Intel® C Compiler GNU*/Microsoft*/HP-UX* compilers
- IA32/Intel® 64/Itanium®/Sun* SPARC* architectures
- Intel XScale® microarchitecture
- Intel, GLIBC, Microsoft*, HP-UX*, Correctly rounded … LIBMs
- VML Vector Math Library. Intel® MKL
- SVML Short Vector Math Library. Intel® C Compiler
- Intel® C Compiler validation
- Intel hardware validation

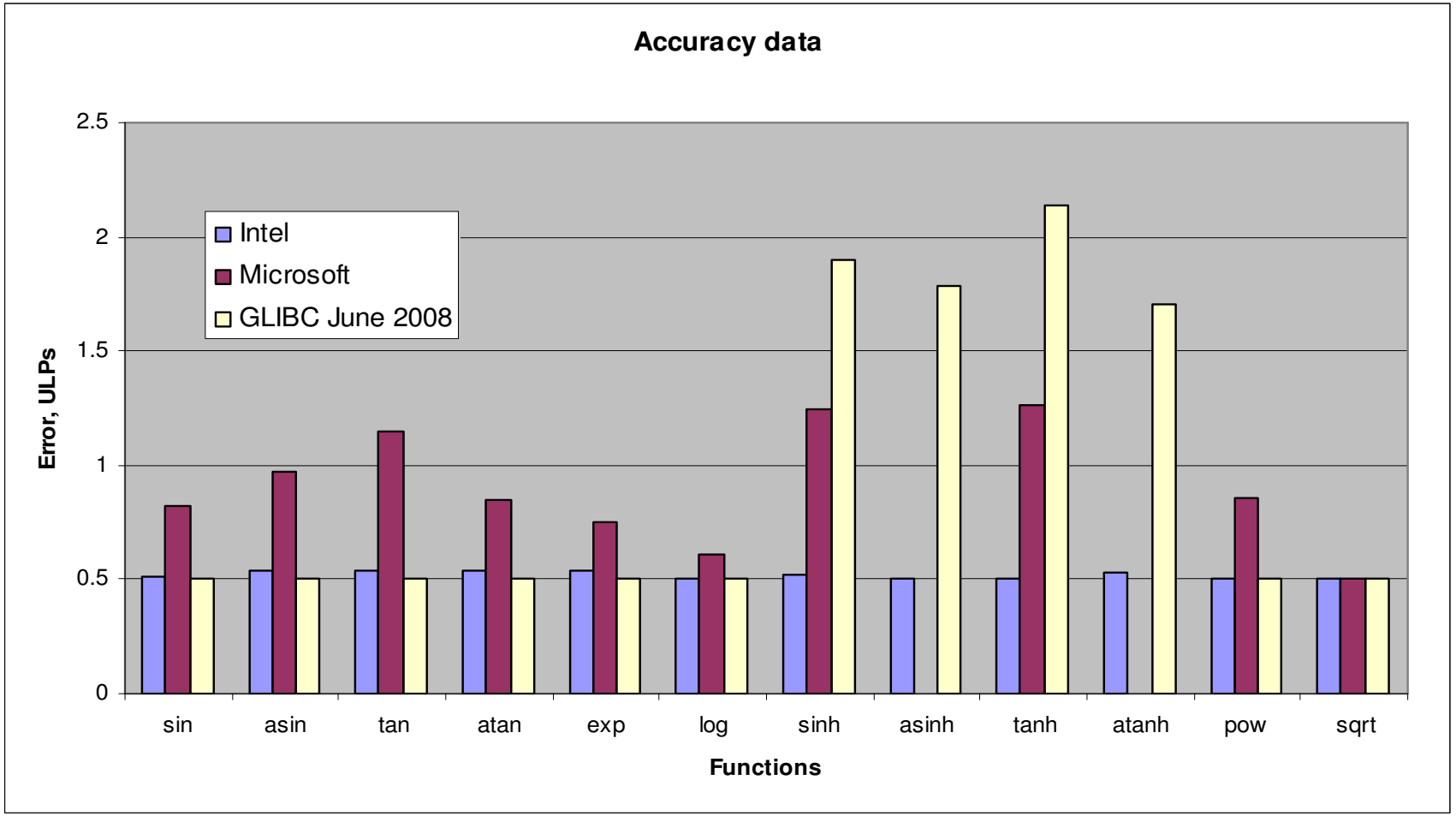
Results.
Intel, GLIBC, Microsoft* LIBMs accuracy



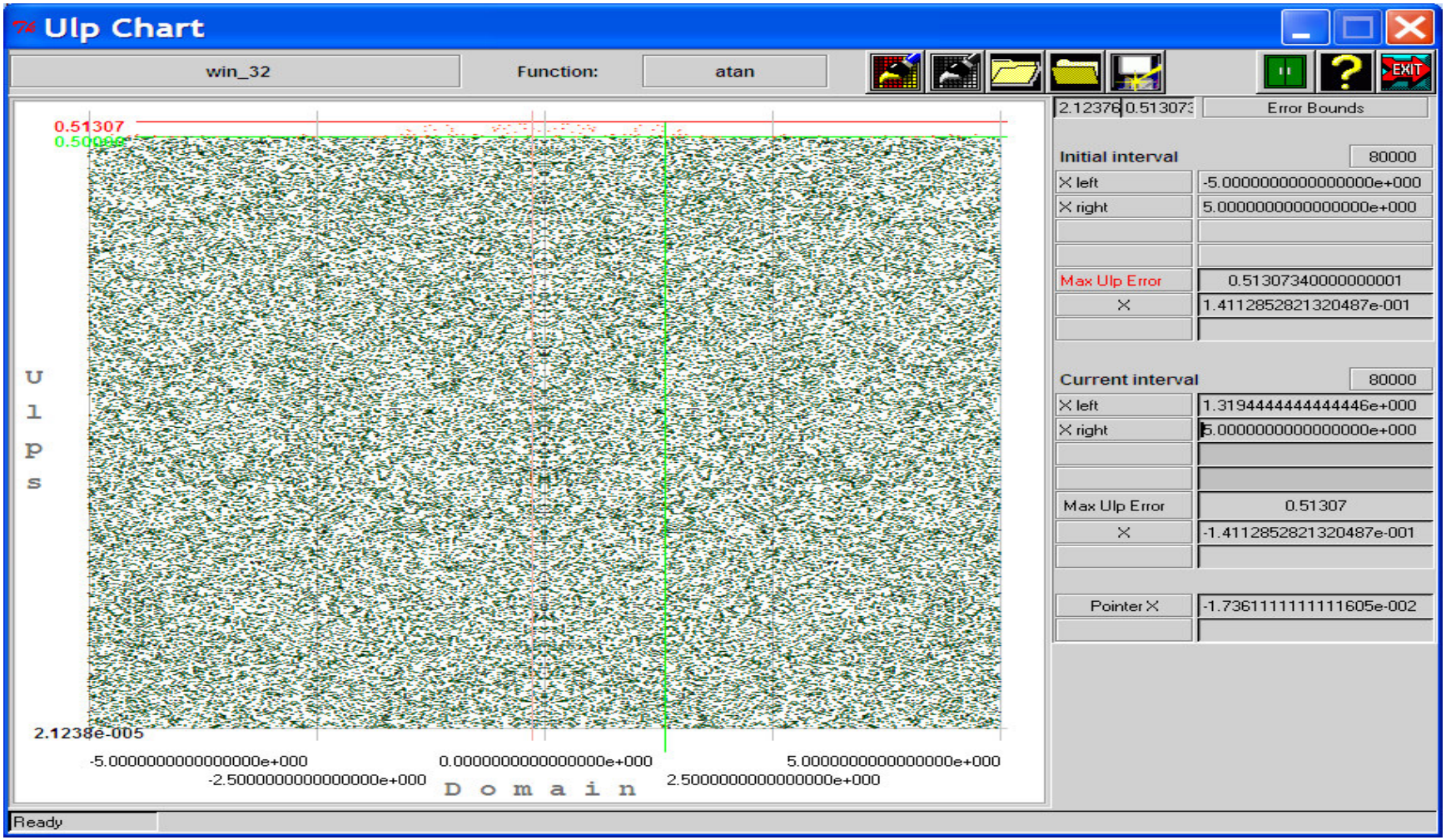
Results.
Close look-up at 0.5 ULP area



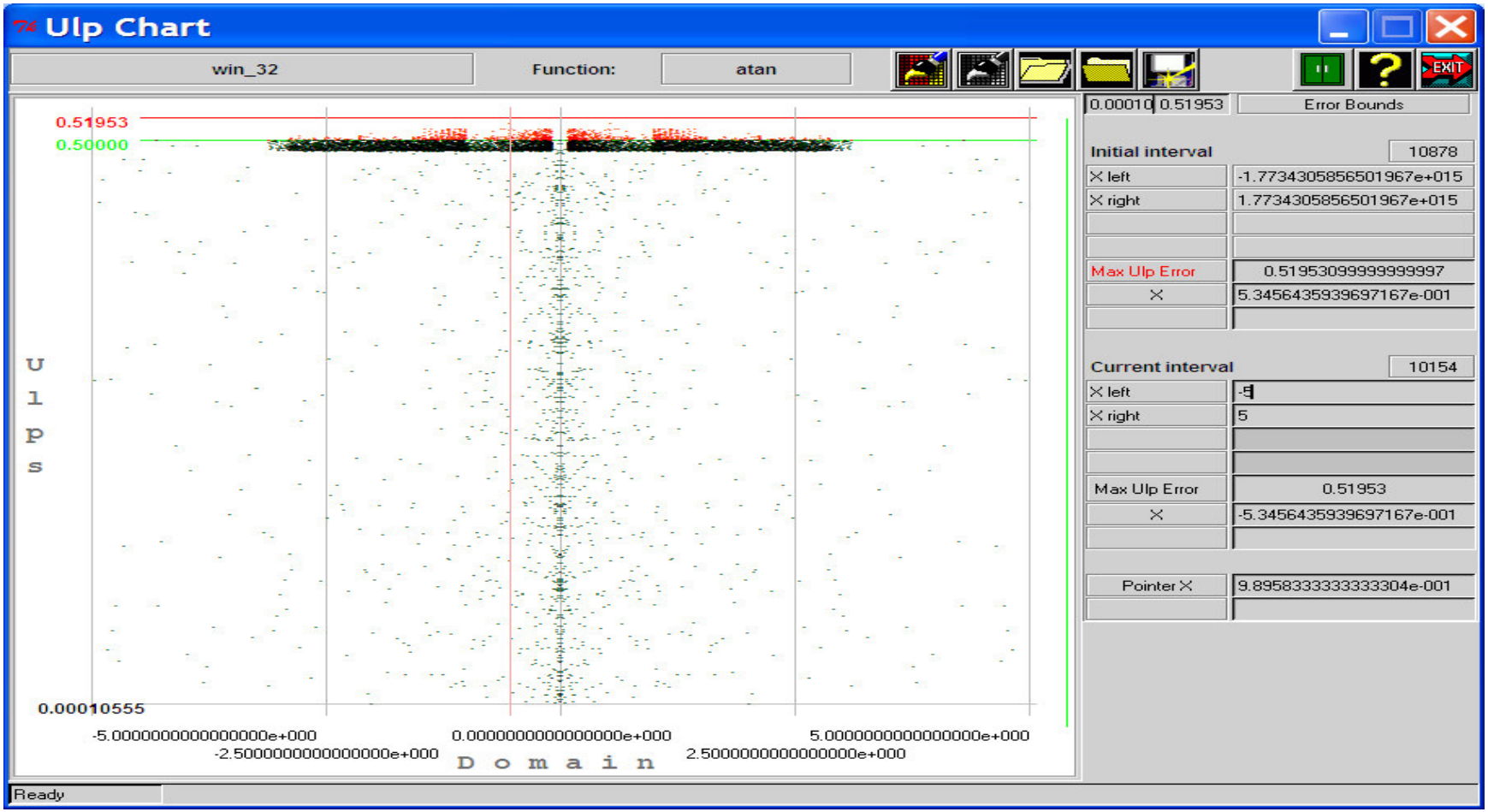
Results.
Most common functions comparison



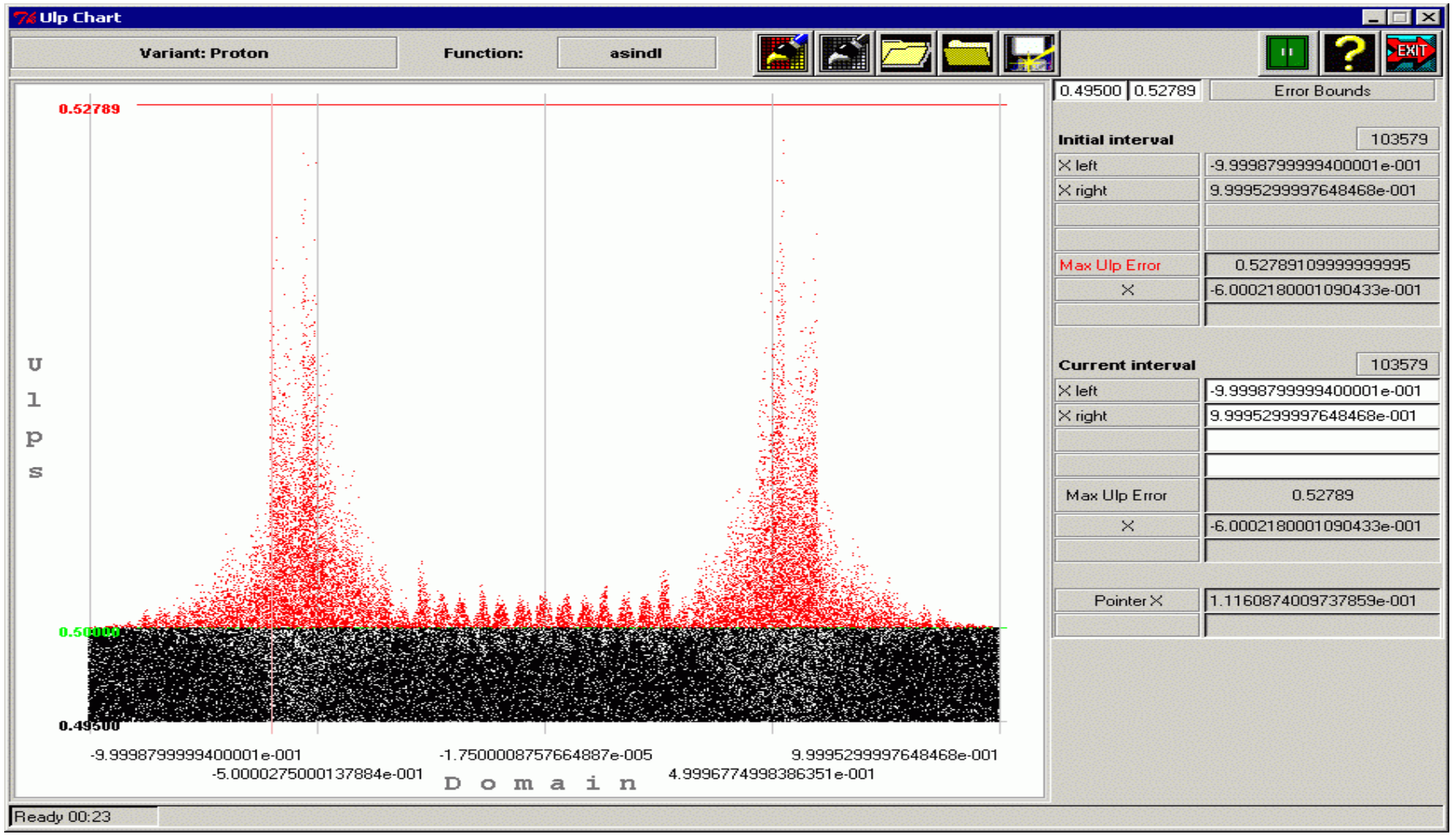
Results. GUI ULP chart
Intel atan(). Random data.



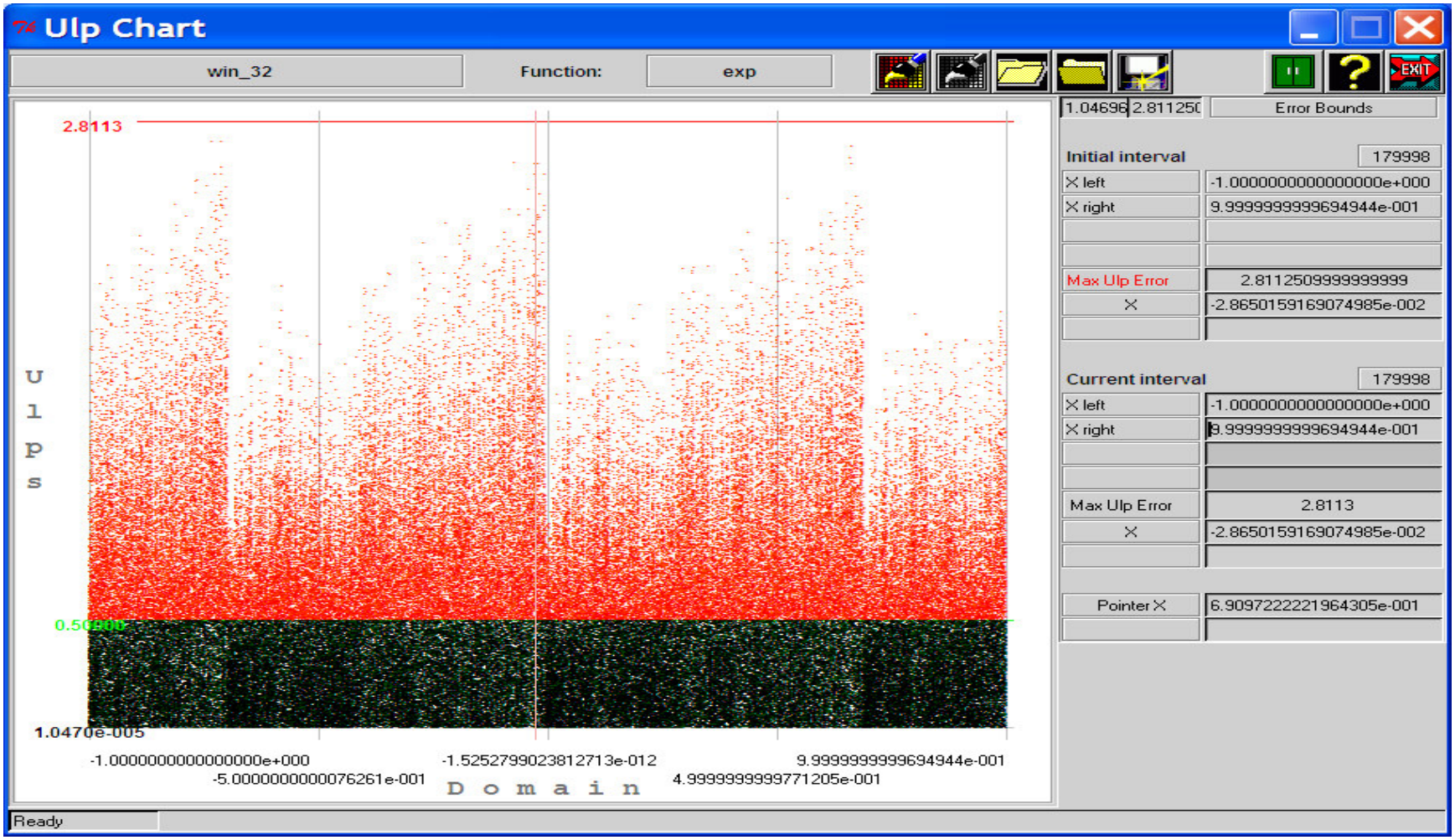
Results. GUI ULP chart
Intel atan(). Suite data.



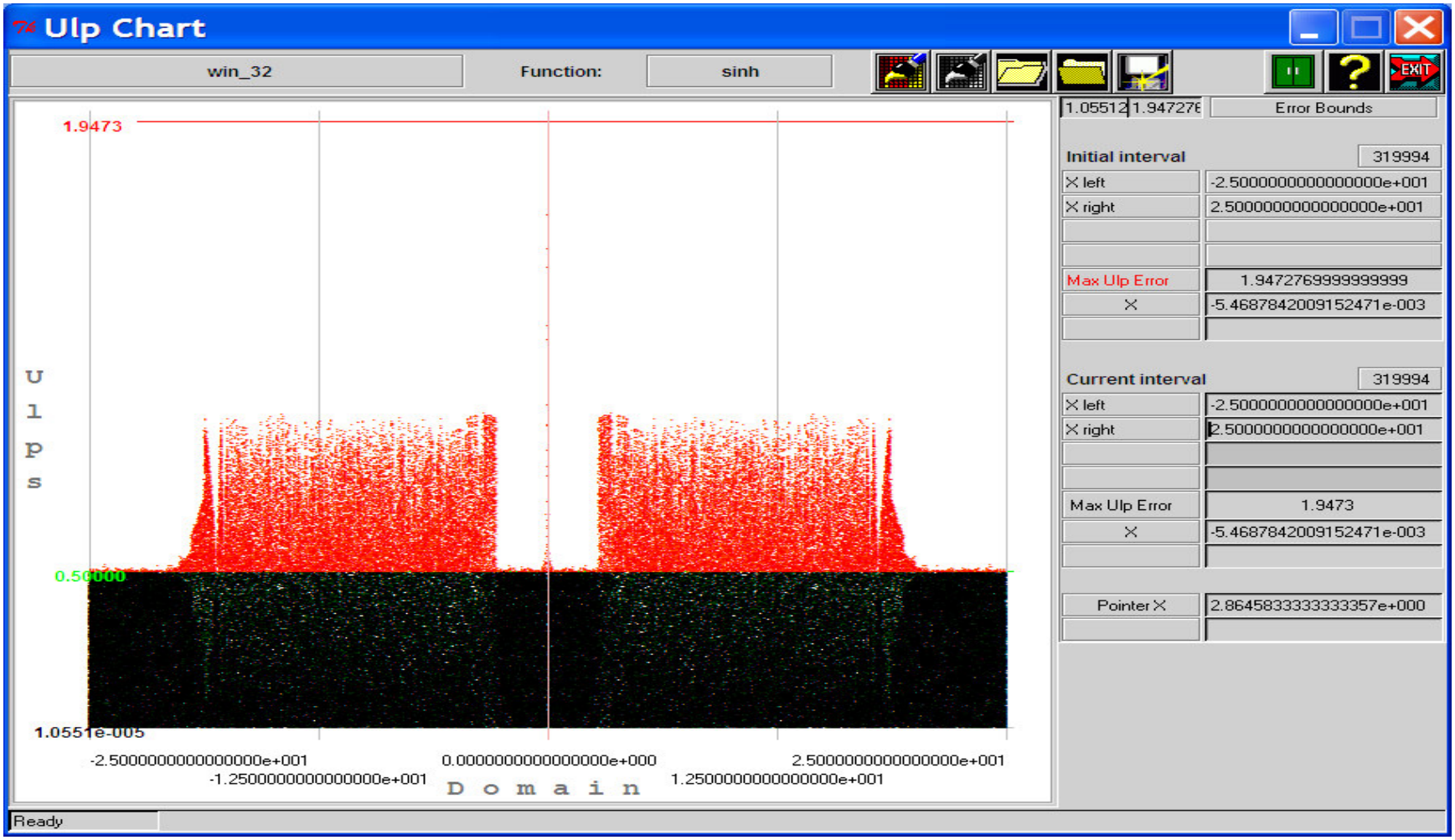
Results. GUI ULP chart
Intel asindl().



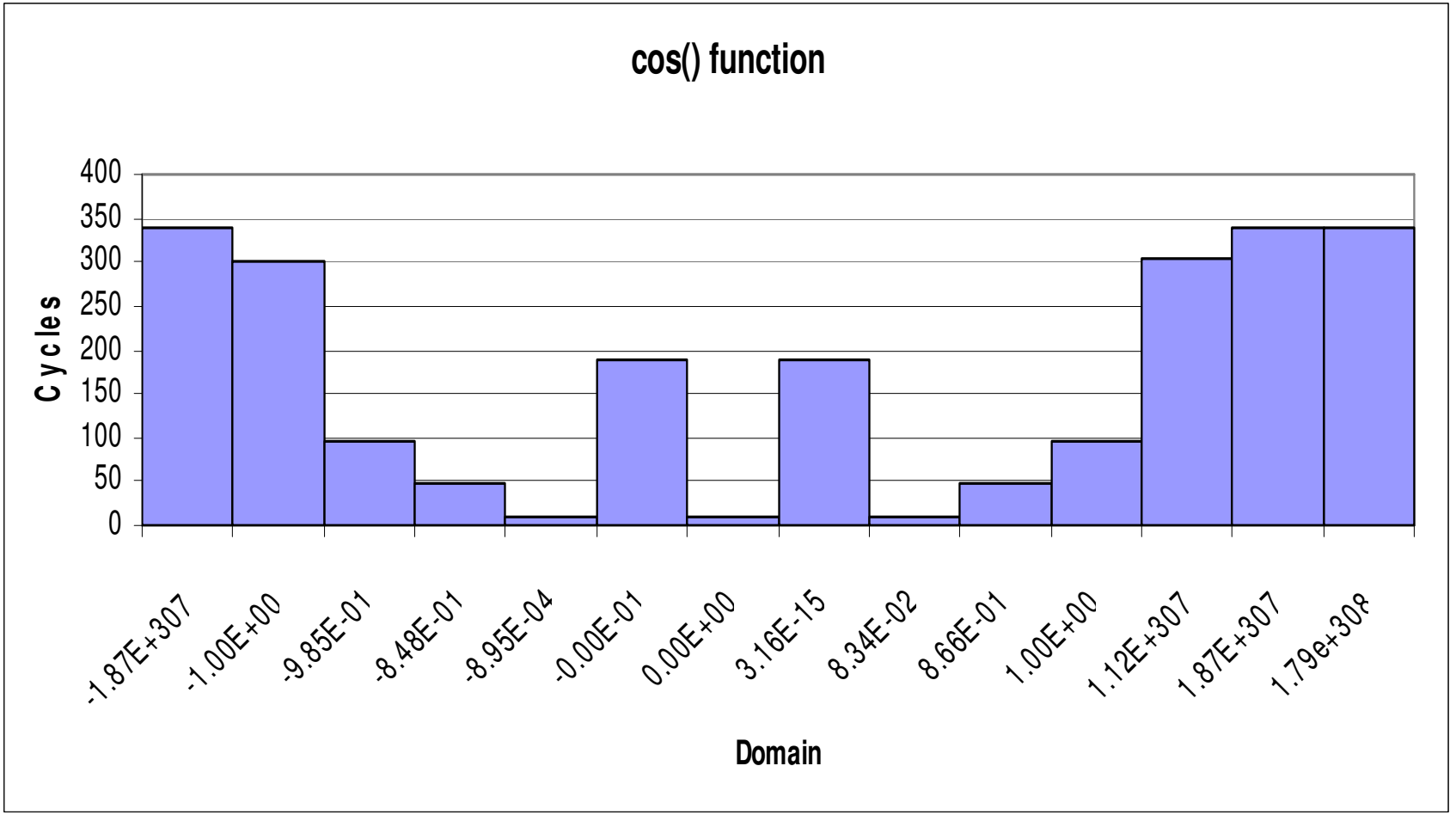
Results. GUI ULP chart Microsoft* exp().



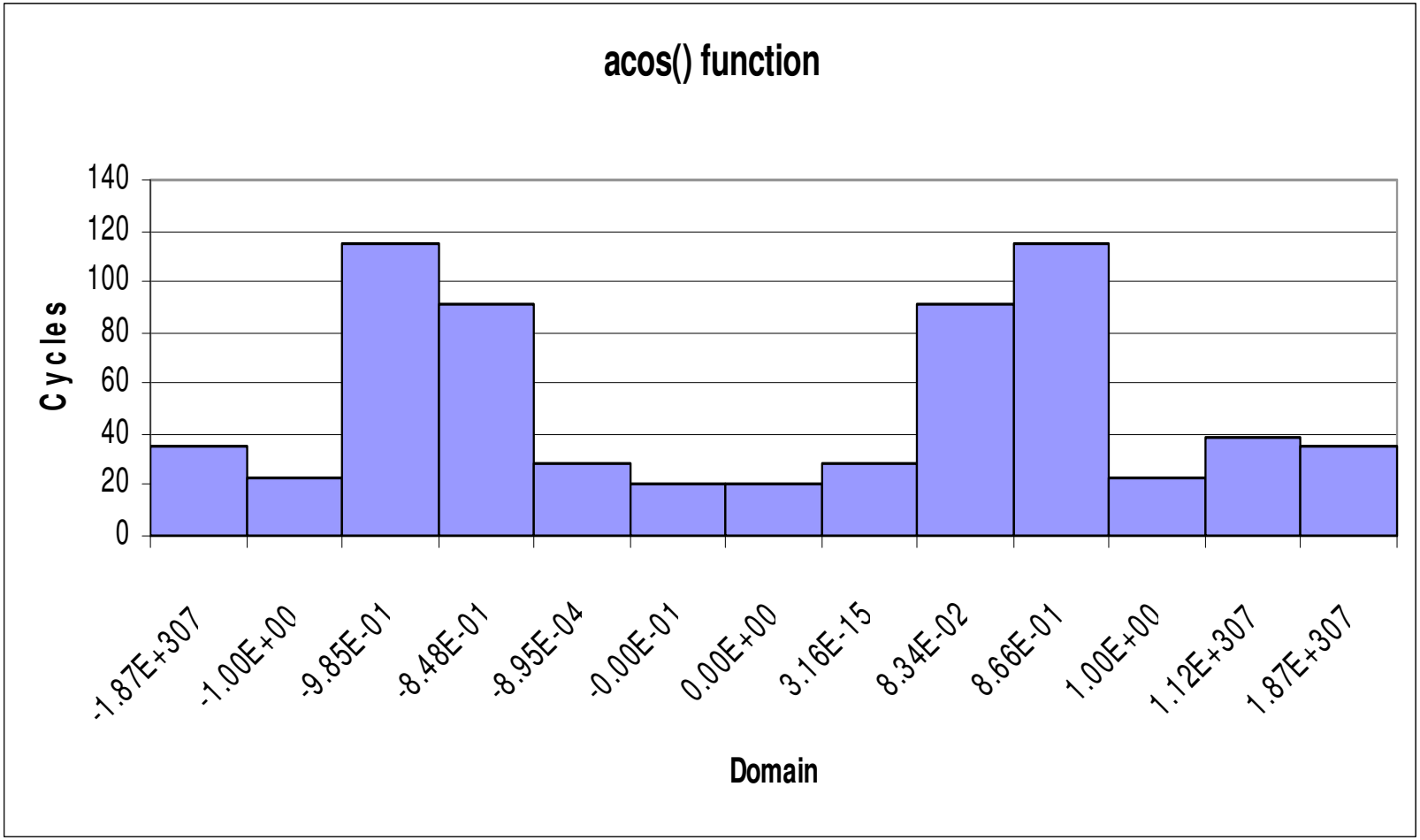
Results. GUI ULP chart Microsoft* sinh().



Results. Auto performance charts Intel cos().



Results. Auto performance charts Intel acos().



Results.
Correctly Rounded (CR) LIBMs

- **IBM* Accurate Portable MathLib*. 1999, 2002.**
  Round to nearest; No errno support; No exception flags settings
  sin cos tan asin acos atan exp log pow sqrt remainder atan2
  http://www.alphaworks.ibm.com/tech/mathlibrary4java
- A library of fast math subroutines for Java. Abraham Ziv, Moshe Olshansky
- **Sun* libmcr* 0.9 Beta; December 2004.**
  All round modes; No errno support; Set exception flags
  exp log pow sin cos tan atan
  http://www.sun.com/download/products.xml?id=41797765
- A reference correctly-rounded library of basic double-precision transcendental elementary functions.
- **ENS-Lyon crlibm* 0.8beta, 1.0beta1; January 2005, February 2007.**
  All round modes; No errno support; Tries to set Overflow/Underflow
  sin cos tan asin atan sinh cosh exp log log2 log10. LGPL.
  http://lipforge.ens-lyon.fr/projects/crlibm/
- A mathematical library (libm) with proven, IEEE-754 compliant, correct rounding in the four rounding modes, and performances comparable to standard libms.

Results.
CR LIBMs. Issues Found

- IBM* Accurate Portable MathLib*: pow, cot, sin, tan, remainder, atan2
- Sun* libmcr*: sin, tan, atan, exp, log, pow
- ENS-Lyon crlibm*: sin, cos, tan, atan, cot, sinh, log, log2, log10

- Architectures difference
- Coding errors
- C99 Special cases (Infs, NaN, Max/Min)
- Denormal operands
- Binade boundaries

Results.
Acknowledgements

- **IBM* Accurate Portable MathLib* readme.**
  "Our thanks are given to the IMLTS (Intel Math Library Test Suite) group, and particularly to Shane Story and Eugeny Gvozdev, for testing the library several times and discovering numerous bugs which otherwise could have remained unnoticed."
- **ENS-Lyon crlibm* .pdf doc.**
  "Many thanks to...The Intel Nizhniy-Novgorod Lab, especially Andrey Naraikin, Sergei Maidanov and Evgeny Gvozdev;"

Designers and Implementers

- Yuri Akutin
- Alexey Ershov
- Evgeny Gvozdev
- Svetlana Gvozdeva
- Vladimir Lunev
- Elena Luneva
- Victor Shumilin
- Shane Story

Most powerful tool in industry,
proven on many LIBMs
including designed as correctly rounded.

Summary

There was no comprehensive suite for testing and verification of Elementary Math functions

Need a powerful, portable, extendable, tunable Tool

White/black box, data driven, convenient, developers/managers usable design

Most powerful tool in industry, proven on many LIBMs including designed as correctly rounded.

References

- [1] ANSI/IEEE 754-1985, American National Standards for Floating-Point Arithmetic.
- [2] ISO/IEC 9899:1999(E) Programming languages — C
- [3] xWilliam J. Cody, Jr. and William Waite. Software Manual for the Elementary Functions, Argonne National Laboratory and Departament of Electrical Engineering University of Color, Englewood Cliffs, New Jersey 07632. Prentice-Hall Inc., 1980.
- [4] Jean-Michel Muller. Elementary Functions: Algorithms and Implementation. Birkhauser, Boston, Basel, Berlin, 1997.
- [5] Vicent Lefevre and Jean-Michel Muller. The Table Maker's Dilemma: our search for worst cases, Projet Arenaire, Laboratory LIP, Ecole Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, France. http://perso.ens-lyon.fr/jean-michel.muller/Intro-to-TMD.htm November, 1998.
- [6] Shmuel Gal and Boris Bachelis. An Accurate Elementary Mathematical Library for the IEEE Float Point Standard, IBM Israe Science and Technology and Scientific Center, Vol. 17, No. 1, Pages 26-45. ACM Transactions on Mathematical Software, June 1991.
- [7] IBM* Accurate Portable MathLib* http://www.alphaworks.ibm.com/tech/mathlibrary4java
- [8] Sun* libmcr*. http://www.sun.com/download/products.xml?id=41797765
- [9] ENS-Lyon crlibm* http://lipforge.ens-lyon.fr/projects/crlibm/


intel
Leap ahead™

Support Material

- Specific bugs found in correctly-rounded LIBMs

IBM* Accurate Portable MathLib*

```
Function COT. Test Quick. Round To Nearest.
Input  Argument:  7be81ae0dffa3b33       ---   7.34096043773508872e+288
Computed Result:  432c0a894108ce00       ---   3.94644198289587200e+15
Expected Result:  432c0a894108cd05       ---   3.94644198289574650e+15
Precise  Result:  432c0a894108cd049f2d86 ---   3.946441982895746310894e+0015

Function COT. Test Quick. Round To Nearest.
Input  Argument:  41339c6fd67805a7       ---   1.28523183776889159e+06
Computed Result:  c3340d0d167bc6c0       ---  -5.64384939715961600e+15
Expected Result:  c3340d0d167bccd6       ---  -5.64384939716117400e+15
Precise  Result:  c3340d0d167bccd6443e1a ---  -5.643849397161174266572e+0015
```

IBM* Accurate Portable MathLib*

```
Function SIN. Test Quick. Round To Nearest.
Input  Argument:  3fd02ae7238a0000       ---   2.52618584352603650e-001
Computed Result:  3fcffe0b1764ca4d       ---   2.49940287039481040e-001
Expected Result:  3fcffe0b1764ca4c       ---   2.49940287039481010e-001
Precise  Result:  3fcffe0b1764ca4c7d2329 ---   2.49940287039481026343739e-
  0001

Function REMAINDER. Test Quick. Round To Nearest.
Input  Argument:  bff0000000000000       ---  -1.00000000000000000e+000
                  bff00068db8bac71       ---  -1.00010000000000000e+000
Computed Result:  3f1a36e2eb1c0000       ---   9.99999999976694200e-005
Expected Result:  3f1a36e2eb1c4000       ---   9.99999999998898700e-005
Precise  Result:  3f1a36e2eb1c4000000000 ---   9.99999999998898658759570e-
  0005
```

IBM* Accurate Portable MathLib*

```
Function REMAINDER. Test Quick. Round To Nearest.
Input  Argument:  bfebbbbbbbbbbbbc       ---  -8.6666666666666700e-001
                  bfd556f8c384071c       ---  -3.3343333333333470e-001
Computed Result:  3fc11ae5a6293bb0       ---   1.3363333333333490e-001
Expected Result:  3fc11ae5a6293bb8       ---   1.3363333333333710e-001
Precise  Result:  3fc11ae5a6293bb8000000 ---   1.3363333333333714776359e-
  0001

Function REMAINDER. Test Intervals. Round To Nearest.
Input  Argument:  bf50000000000000       ---  -9.7656250000000000e-004
                  bf2999999999999f       ---  -1.9531250000000150e-004
Computed Result:  3c30000000000000       ---   8.6736173798840355e-019
Expected Result:  3c2b000000000000       ---   7.3183646642771549e-019
Precise  Result:  3c2b0000000000000000 ---   7.3183646642771549295503 1e-
  0019
```

IBM* Accurate Portable MathLib*

```
Function REMAINDER. Test Intervals. Round To Nearest.
Input  Argument:  ffefffffffffffff       ---  -1.7976931348623157e+308
                  7fefffffffffffff       ---   1.7976931348623157e+308
Computed Result:  0000000000000000       ---   Zero
Expected Result:  8000000000000000       ---  -Zero
Precise  Result:  80000000000000000000 ---  -Zero

Function TAN. Test Special. Round To Nearest.
Input  Argument:  7ff0000000000001       ---   SNaN
Computed Result:  0000000000000000       ---   Zero
Expected Result:  7ff8000000000001       ---   QNaN

Function TAN. Test Special. Round To Nearest.
Input  Argument:  7ff8000000000001       ---   QNaN
Computed Result:  0000000000000000       ---   Zero
Expected Result:  7ff8000000000001       ---   QNaN
```

IBM* Accurate Portable MathLib*

```
Function TAN. Test Special. Round To Nearest.
Input  Argument:  7ff0000000000000       ---   Infinity
Computed Result:  0000000000000000       ---   Zero
Expected Result:  fff8000000000000       ---   QNaN_Indefinite

Function ATAN2. Test Quick. Round To Nearest.
Input  Argument:  7fefffffffffffff       ---   1.7976931348623157e+308
                  7fefffffffffffff       ---   1.7976931348623157e+308
Computed Result:  fff8000000000000       ---   QNaN_Indefinite
Expected Result:  3fe921fb54442d18       ---   7.8539816339744280e-001
Precise  Result:  3fe921fb54442d18469898 ---   7.85398163397448309615e-0001
```

IBM* Accurate Portable MathLib*

```
Function POW. Test Quick. Round To Nearest.
Input  Argument:  bf50000000000000       ---  -9.76562500000000000e-004
                  0000000000000000       ---   Zero
Computed Result:  7ff8000000000000       ---   QNaN
Expected Result:  3ff0000000000000       ---   1.00000000000000000e+000
Precise  Result:  3ff00000000000000000 ---   1.00000000000000000000e+0000

Function POW. Test Quick. Round To Nearest.
Input  Argument:  7fe0000001ffffff       ---   8.98846574128086550e+307
                  bff0000000000000       ---  -1.00000000000000000e+000
Computed Result:  0007fffffff00000       ---   1.11253692096455460e-308
Expected Result:  0007fffffff00001       ---   1.11253692096455510e-308
Precise  Result:  0007fffffff0000009fffff ---   1.11253692096455494187805e-0308
```

IBM* Accurate Portable MathLib*

```
Function POW. Test Special. Round To Nearest.
Input Arguments:  8000000000000000       ---  -Zero
                  bfefffffffffffff       ---  -9.99999999999999890e-001
Computed Result:  7ff8000000000000       ---   QNaN
Expected Result:  7ff0000000000000       ---   Infinity

Function POW. Test Special. Round To Nearest.
Input Arguments:  bff0000000000000       ---  -1.00000000000000000e+000
                  7ff0000000000000       ---   Infinity
Computed Result:  7ff8000000000000       ---   QNaN
Expected Result:  3ff0000000000000       ---   1.00000000000000000e+000

Function POW. Test Special. Round To Nearest.
Input Arguments:  3ff0000000000000       ---   1.00000000000000000e+000
                  fff8000000000001       ---   QNaN
Computed Result:  7ff8000000000000       ---   QNaN
Expected Result:  3ff0000000000000       ---   1.00000000000000000e+000
```

Sun* libmcr*

```
Function SIN. Test Quick. Round To Nearest.
Input  Argument:  000fffffffffffff       ---   2.22507385850720089e-308
Computed Result:  000ffffffffffffe       ---   2.22507385850720039e-308
Expected Result:  000fffffffffffff       ---   2.22507385850720089e-308
Precise  Result:  000ffffffffffffeffffff ---   2.225073858507200889024e-0308

Function TAN. Test Quick. Round To Nearest.
Input  Argument:  000fffffffffffff       ---   2.22507385850720089e-308
Computed Result:  0010000000000000       ---   2.22507385850720138e-308
Expected Result:  000fffffffffffff       ---   2.22507385850720089e-308
Precise  Result:  000ffffffffffff000000 ---   2.225073858507200889024e-0308
```

Sun* libmcr*

```
Function ATAN. Test Quick. Round To Nearest.
Input  Argument:  000fffffffffffff       ---   2.22507385850720089e-308
Computed Result:  000ffffffffffffe       ---   2.22507385850720039e-308
Expected Result:  000fffffffffffff       ---   2.22507385850720089e-308
Precise  Result:  000ffffffffffffeffffff ---   2.22507385850720088902459e-
  0308

Function EXP. Test Quick. Round To Nearest.
Input  Argument:  bc90000000000001       ---  -5.55111512312578393e-17
Computed Result:  3ff0000000000000       ---   1.00000000000000000e+00
Expected Result:  3feffffffffffff        ---   9.99999999999999889e-01
Precise  Result:  3feffffffffffff7fffff ---   9.99999999999999944488849e-
  0001
```

Sun* libmcr*

```
Function LOG. Test Special. Round To Nearest.
Input  Argument:  fff0000000000000       ---  -Infinity
Computed Result:  fff0000000000000       ---  -Infinity
Expected Result:  fff8000000000000       ---   QNaN_Indefinite

Function POW. Test Quick. Round To Nearest.
Input  Argument:  bfe0000000022000       ---  -5.00000000015461410e-01
                  c08e280000000000       ---  -9.65000000000000000e+02
Computed Result:  bfeffffffeffac004      ---  -9.99999970159479279e-01
Expected Result:  fc3ffffffeffac004      ---  -3.11850039059032140e+290
Precise  Result:  fc3ffffffeffac00403b0c6 ---  -3.118500390590321409706 3e+0290
```

Sun* libmcr*

```
Function POW. Test Quick. Round To Nearest.
Input  Argument:  bfe0000044000000       ---  -5.000001266598701 48e-01
                  c086100000000000       ---  -7.060000000000000 0e+02
Computed Result:  3feffe88f898fad7       ---   9.998211722775874 89e-01
Expected Result:  6c0ffe88f898fad7       ---   3.365884955792332 37e+212
Precise  Result:  6c0ffe88f898fad6805ef5 ---   3.365884955792332 1850473e+0212

Function POW. Test Quick. Round To Nearest.
Input  Argument:  bfeffffffffffff8       ---  -9.999999999999988 89e-01
                  bff0000000000000       ---  -1.000000000000000 0e+00
Computed Result:  bff0000000000000       ---  -1.000000000000000 0e+00
Expected Result:  bff0000000000001       ---  -1.000000000000002 2e+00
Precise  Result:  bff0000000000000800000 ---  -1.000000000000000 1110223e+0000
```

Sun* libmcr*

```
Function POW. Test Cmdline. Round To Nearest.
Input  Argument:  4050000000000013       ---   6.4000000000002700e+01
                  3fe0000000000000       ---   5.0000000000000000e-01
Computed Result:  402000000000000a       ---   8.0000000000001776e+00
Expected Result:  4020000000000009       ---   8.0000000000001599e+00
Precise  Result:  40200000000000097fffff ---   8.0000000000000168753899e+0000

Function POW. Test Intervals. Round To Nearest.
Input  Argument:  3fe0000000000000       ---   5.0000000000000000e-01
                  3ffffffffffffffb       ---   1.9999999999999889e+00
Computed Result:  3ff0000000000000       ---   1.0000000000000000e+00
Expected Result:  3fd0000000000003       ---   2.5000000000000167e-01
Precise  Result:  3fd000000000003773a77 ---   2.5000000000000019238699e-0001
```

Sun* libmcr* Exception flags

Function	Flags IUOZVD	Rounding	Function	Flags IUOZVD	Rounding	Function	Flags IUOZVD	Rounding
SIN	FF****	Near	ATAN	FF****	Near	POW	*F*FF*	Near
SIN	FF****	+Inf	ATAN	FF****	+Inf	POW	*F*FFF	+Inf
SIN	FF****	-Inf	ATAN	FF****	-Inf	POW	*F*FFF	-Inf
SIN	F*****	Zero	ATAN	F*****	Zero	POW	*F*FFF	Zero
COS	******	Near	EXP	*F****	Near			
COS	******	+Inf	EXP	*F****	+Inf			
COS	******	-Inf	EXP	*F****	-Inf			
COS	******	Zero	EXP	*F****	Zero			
TAN	******	Near	LOG	****F*	Near			
TAN	******	+Inf	LOG	****F*	+Inf			
TAN	******	-Inf	LOG	****F*	-Inf			
TAN	******	Zero	LOG	****F*	Zero			

ENS-Lyon crlibm*

sin, cos, tan, atan, round to nearest

```
Input  Argument:  7fe0000000000000       ---   8.98846567431157954e+307
Computed Result:  0000000000000000       ---   Zero
Expected Result:  3fe205248cbdb760       ---   5.63127779850884025e-01
Precise  Result:  3fe205248cbdb75fe5a5cc ---   5.631277798508840134529 4e-0001

Input  Argument:  7ff0000000000001       ---   SNaN
Computed Result:  0000000000000000       ---   Zero
Expected Result:  7ff8000000000001       ---   QNaN

Input  Argument:  7fffffffffffffff       ---   QNaN
Computed Result:  0000000000000000       ---   Zero
Expected Result:  7fffffffffffffff       ---   QNaN
```

ENS-Lyon crlibm*

sin, cos, tan; round to nearest

```
Input  Argument:   7ff0000000000000       ---   Infinity
Computed Result:   0000000000000000       ---   Zero
Expected Result:   fff8000000000000       ---   QNaN_Indefinite

Input  Argument:   fff0000000000000       ---  -Infinity
Computed Result:   0000000000000000       ---   Zero
Expected Result:   fff8000000000000       ---   QNaN_Indefinite
```

ENS-Lyon crlibm*

```
Function ATAN. Test Template. Round To Nearest.
Input  Argument:  43b8000000000000       ---   1.72938225691027046e+18
Computed Result:  7ff8000000000000       ---   QNaN
Expected Result:  3ff921fb54442d18       ---   1.57079632679489656e+00
Precise  Result:  3ff921fb54442d1845edee ---   1.570796326794896618653e+0000

Function SINH. Test Quick. Round To Nearest.
Input  Argument:  8000000000000000       ---  -Zero
Computed Result:  0000000000000000       ---   Zero
Expected Result:  8000000000000000       ---  -Zero
Precise  Result:  80000000000000000000 ---  -Zero
```

ENS-Lyon crlibm*

log2, log10; round to nearest

```
Input  Argument:  0000000000000000       ---   Zero
Computed Result:  7ff0000000000000       ---   Infinity
Expected Result:  fff0000000000000       ---  -Infinity

Input  Argument:  8000000000000000       ---  -Zero
Computed Result:  7ff0000000000000       ---   Infinity
Expected Result:  fff0000000000000       ---  -Infinity

Function COT. Test Quick. Round To Nearest.
Input  Argument:  3ffe417c1bda4617       ---   1.89098749999937410e+00
Computed Result:  bfd47e031e5863fb       ---  -3.20191173204477486e-01
Expected Result:  bfd538f5dc20b48a       ---  -3.31601586311827234e-01
Precise  Result:  bfd538f5dc20b48a613329 ---  -3.316015863118272555091 7e-0001
```

ENS-Lyon crlibm*

```
Function SIN. Test Template. Round To Zero.
Input  Argument:  0010000000000001       ---   2.22507385850720188e-308
Computed Result:  0010000000000001       ---   2.22507385850720188e-308
Expected Result:  0010000000000000       ---   2.22507385850720138e-308
Precise  Result:  001000000000000fffff ---   2.225073858507201877155 8e-0308

Function SIN. Test Template. Round To -Infinity.
Input  Argument:  0010010000000000       ---   2.2256170894296884e-308
Computed Result:  0010010000000000       ---   2.2256170894296884e-308
Expected Result:  00100fffffffff       ---   2.2256170894296800e-308
Precise  Result:  00100fffffffffffffff ---   2.225617089429688492802 9e-0308
```

ENS-Lyon crlibm*

```
Function EXP. Test Quick. Round To Zero.
Input  Argument:  3fe0000000000000       ---   5.0000000000000000e-01
Computed Result:  0000000000000000       ---   Zero
Expected Result:  3ffa61298e1e069b       ---   1.648721270700127 97e+00
Precise  Result:  3ffa61298e1e069bc972df ---   1.64872127070012814 68486e+0000

Function EXP. Test Intervals. Round To -Infinity.
Input  Argument:  bfd4bfec17d14ef8       ---  -3.2421400381029696 9e-01
Computed Result:  7fdfffffffffffff       ---   8.9884656743115785 4e+307
Expected Result:  3fe7239922267910       ---   7.2309548063256737 e-01
Precise  Result:  3fe723992226791040c657 ---   7.2309548063256737 260601e-0001
```

ENS-Lyon crlibm*

```
Function SIN. Test Quick. Round To -Infinity.
Input  Argument:  3ff921fb54c42d19       ---   1.570796326754193e+00
Computed Result:  3fefffffffffffe        ---   9.9999999999999778e-01
Expected Result:  3fefffffffffff         ---   9.9999999999999889e-01
Precise  Result:  3fefffffffffffffbffff ---   9.99999999999999826527e-0001

Function COS. Test Intervals. Round To -Infinity.
Input  Argument:  401921fb54c42d18       ---   6.28318531463016683e+00
Computed Result:  3fefffffffffffe        ---   9.9999999999999778e-01
Expected Result:  3fefffffffffff         ---   9.9999999999999889e-01
Precise  Result:  3fefffffffffffc00000 ---   9.99999999999999724442e-0001

Function ATAN. Test Intervals. Round To -Infinity.
Input  Argument:  3e40000000000000       ---   7.45058059692382812e-09
Computed Result:  3e3fffffffffffe        ---   7.45058059692382647e-09
Expected Result:  3e3fffffffffff         ---   7.45058059692382730e-09
Precise  Result:  3e3fffffffffffd55555 ---   7.450580596923827987136 5e-0009
```

ENS-Lyon crlibm*

```
Function COSH. Test Intervals. Round To -Infinity.
Input  Argument:  40865294a5294a53       ---   7.14322580645161338e+02
Computed Result:  7ff0000000000000       ---   Infinity
Expected Result:  7fefffffffffffff       ---   1.79769313486231571e+308
Precise  Result:  7ff76b6b13dc6a52dca893 ---   8.4202517717590034186 37e+0309

Function LOG. Test Quick. Round To +Infinity.
Input  Argument:  3ff0000000000000       ---   1.0000000000000000e+00
Computed Result:  0000000000000001       ---   4.9406564584124654 4e-324
Expected Result:  0000000000000000       ---   Zero
Precise  Result:  00000000000000000000 ---   Zero
```

ENS-Lyon crlibm* Exception flags

Function	Flags IUOZVD	Roun ding	Function	Flags IUOZVD	Roun ding
SIN	FF**F*	Near	ASIN	FF****	Near
SIN	FF**F*	+Inf	ASIN	FF****	+Inf
SIN	FF**F*	-Inf	ASIN	FF****	-Inf
SIN	FF**F*	Zero	ASIN	FF****	Zero
COS	F***FF	Near	ATAN	FF****	Near
COS	F***FF	+Inf	ATAN	FF****	+Inf
COS	F***F*	-Inf	ATAN	FF****	-Inf
COS	F***F*	Zero	ATAN	FF****	Zero
TAN	FF**F*	Near	EXP	FFF**F	Near
TAN	FF**F*	+Inf	EXP	FFF**F	+Inf
TAN	FF**F*	-Inf	EXP	FFF**F	-Inf
TAN	FF**F*	Zero	EXP	FFF**F	Zero

ENS-Lyon crlibm* Exception flags

Function	Flags IUOZVD	Roun ding	Function	Flags IUOZVD	Roun ding
SINH	FFF*F*	Near	LOG2	F**F**	Near
SINH	FFF*F*	+Inf	LOG2	***F**	+Inf
SINH	FFF*F*	-Inf	LOG2	***F**	-Inf
SINH	FFF*F*	Zero	LOG2	***F**	Zero
COSH	F*F*FF	Near	LOG10	F**F**	Near
COSH	F*F*F*	+Inf	LOG10	FF*F**	+Inf
COSH	F*F*F*	-Inf	LOG10	FF*F**	-Inf
COSH	F*F*F*	Zero	LOG10	FF*F**	Zero
LOG	***F**	Near			
LOG	F**F**	+Inf			
LOG	F**F**	-Inf			
LOG	F**F**	Zero			

IBM* Accurate Portable MathLib*

- Architectures difference.
  First many functions failed because developer implied FP ops performed in double precision (uses appropriate FP constants). Intel processor default mode is 80 bit for Linux. Using double precision resolved problem.
- pow: for x^{-1} user returns 1/x; Compiler uses fdiv hardware instruction, double roundings occurs for Intel processor precision control set to 53 bits. It works fine for full 80 bit precision control set.
- Developer bugs.
  sin: error slightly more than 0.5 ULP
  cot, remainder: Big ULP error
  cot: "for some unknown reason I thought that an absolute error of 1/x equals to that of x..."
- C99 Special cases: atan2, remainder, pow, tan: processing Infs, NaNs, Max/Min FP numbers.

Sun* libmcr*

- Architecture difference again – 80 vs. 64 bits internal calculations.

  "Arguments that result in correct results on SPARC, but fail for i386 machines. These are the result of i386 machines using double-extended to do FP arithmetic. SPARC uses only 64-bit floating point registers. That being the case, several failures on i386 are the result of constants that are meant for 64-bit FP registers. The exp() and pow() failures are isolated to the i386 and I believe have to do with double rounding issues. I'm looking for solutions for these ... as I type."

Sun* libmcr*, ENS-Lyon crlibm*

- Mostly developer's bugs.
  - Binade boundaries – change ULP value.
  - C99 Special cases.
  - Work with denormal numbers.
- "Errors replicated on SPARC, our initial development machine. These were coding errors that left holes in processing."
- "We have a stupid bug there: for directed rounding modes I compute the ulp of a number and then add or subtract it, and of course for 1 (and other numbers that lie exactly at the boundary of a binade) I subtract a quantity that is twice the ulp of the previous binade. The solution is to add/subtract 1 to the binary integer representing the double, which always work except for zero and infty."
- "Most errors related to special cases were stupid ones (reasoning only on positive values, < instead of <= , this kind of mistakes)."