The Regina Rexx Interpreter – Math Functions

Patrick TJ McPhee (ptjm@interlog.com)
DataMirror Corporation

version 1.0.0, 27 October 2001

Contents

1 Introduction

This paper describes two function libraries which provide common trigonometric functions to rexx programs. Amath was written during the fall of 1998 in response to the newsgroup posting contained in the file amigamath.txt. It is my hope that this library is compatible with the amiga rexx math library. Rxmath was written during the the fall of 2001 to provide a library compatible with IBM's rxmath library, which had appeared earlier in the year.

The libraries are little more than wrappers around the C math library—in particular they do not allow arbitrary-precision operations. If arbitrary-precision calculations are required, I recommend John Brock's rxxmath library, which is written in rexx, and which does provide arbitrary-precision support, at the expense of speed.

The two libraries are functionally equivalent, with these exceptions.

- RxMath supports arguments in radians, degrees, or gradians, while amath accepts radians only;
- RxMath has a function which returns the value of π (but $\pi = 3.141592653897832\ldots$);
- amath supports inverse hyperbolic functions on some (read: Unix) platforms;
- amath provides secant and cosecant functions (but these are easily derived from sine and cosine);
- amath provides ceil and floor functions (but these can be achieved using format( )).

The functions are provided in hopes that they will be useful, but there is no warranty.

2 Housekeeping Functions

2.1 MathLoadFuncs/MathDropFuncs

In both libraries, the only exported function is called MathLoadFuncs. It must be loaded using RxFuncAdd, then called to make the other functions available. It was my intention for MathLoadFuncs to query the rexx interpreter for the setting of numeric digits, ignoring subsequent changes to the setting, however there is no way to do this with the current Regina implementation, so it always ignores the numeric digits setting.

To make the RxMath library functions available:

```
call rxfuncadd 'mathloadfuncs', 'rxmath', 'mathloadfuncs'
call mathLoadFuncs
```

while to make the amath library functions available:

```
call rxfuncadd 'mathloadfuncs', 'rexxmath', 'mathloadfuncs'
call mathLoadFuncs
```

It is possible to load both libraries in the same program, however the statements above reuse the function name 'mathloadfuncs'. Since you only ever need to call the function once for each library, this doesn't matter, but if it bothers you, the third argument to rxFuncAdd is the name of the function within the rexx interpreter, and you can change it to anything you like.

If you have an error loading these libraries (or any library), you can sometimes get useful information by calling the Regina-specific function rxfuncerrmsg:

```
if rxfuncadd('mathloadfuncs', 'rexxmath', 'mathloadfuncs') then
  say rxfuncerrmsg()
else
  call mathLoadFuncs
```

I know of a few problems which will prevent these libraries from loading: If you use rexx.exe from the Regina distribution, you cannot load external function libraries. You must use regina.exe instead. The difference between the two programs is that rexx.exe has the Rexx interpreter linked into the executable, while regina.exe loads it from a shared library. Programs run faster in the statically linked interpreter, but it can't load function libraries (it's not technically possible on Windows, although it is on most Unix systems, Regina still doesn't allow it).

If you are using the initial release of Windows 95, it does not include the file msvcrt.dll. You must get this file from Microsoft support. Many applications install it, but I find that unbelievable, since it's supposed to be part of the operating system.

Most systems expect the first argument to rxfuncadd to exactly match the case of the function name in the shared library (it's all lower-case). Unix systems expect the second argument to match the case of the file name containing the shared object (which is generally all lower-case). More recent versions of Regina have a work-around to allow the case in the rxfuncadd call to differ from the actual case in the library.

```
call mathDropFuncs
```

After calling mathLoadFuncs, you can call mathDropFuncs to unregister all the functions. This is a useful operation with IBM interpreters, since they make function libraries global to all rexx programs, and there's no way to replace a library without unloading it first. There's no real value when using Regina.

3 RxMath Library

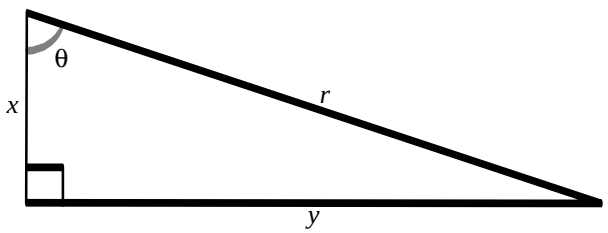
The RxMath Library is a straight copy of the interface of IBM's RxMath library, which appeared in March 2001. The implementation is based on IBM's documentation, while this manual is based on my reading of the C code about a month after implementation. It's shorter than IBM's manual, but I feel it's more informative, and it's got some slick diagrams.

I expect that this library is fully compatible with IBM's library, with two exceptions: First, I do no error checking, so non-numeric arguments are silently converted to 0, while invalid arguments will generally result in a return code of NaN, rather than ERROR. Second, I never check the numeric digits setting, meaning a default precision of 16 is in effect always. Most functions take precision as an argument, so lower precisions can be achieved either by adding 0 to the result, or by passing the desired precision to the function. I believe that the differences will result in a measurable improvement in performance.

The function naming convention is 'RxCalc' followed by a common abbreviation of the trigonometric function. Thus 'sine' becomes 'RxCalcSin'.

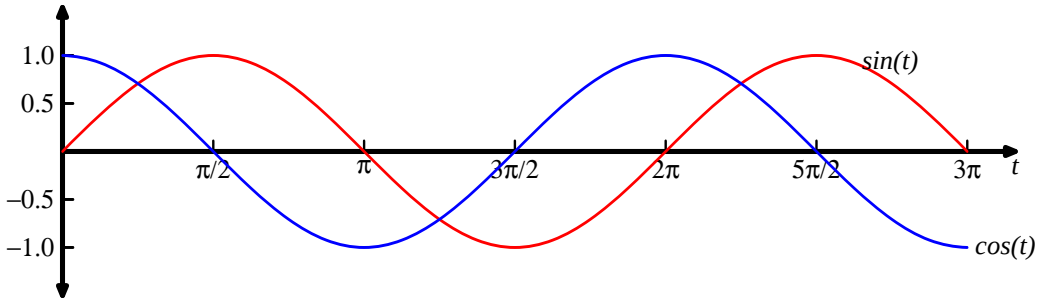
3.1 Trigonometric Functions

Trigonometric functions are functions on an angle, θ, which we typically define in terms of the cartesian co-ordinates, (x,y), of some point (r,θ). *x*, *y*, and *r* can be taken as the sides of a right-angled triangle, as shown in the figure.



sine θ = *y* / *r*
cosine θ = *x* / *r*
tangent θ = *y* / *x*
cosecant θ = *r* / *y*
secant θ = *r* / *x*
cotangent θ = *x* / *y*

Because of the triangle thing, trigonometric functions are often used in geometrical applications. Because they have a boring periodic nature, they are also used in used in signal processing applications. For instance, sine waves are used at the start of 'Lemmings' in the classic album *Pawn Hearts*.



In mathematics, angles are typically measured in radians. The size of the angle in radians is the ratio between the arc of a circle which subtends the angle and the radius of the circle. The traditional, non-mathematical measure of angles is in degrees, minutes, and seconds, which are a base-60 system, and so presumably go back to the Phonecians. There are 360 degrees in a circle, 60 minutes in a degree, and 60 seconds in a minute, althought you don't need to know that, since the library takes degrees as decimal numbers. Finally, engineering calculations often use gradians, which divide the circle into 400. The rxmath library accepts all three measures and can return all three types.

Since calculations involving radians typically involve π, there's a function which returns it (see below for a description of the *precision* argument:

RxCalcPi(*precision*)

The calling sequences of each trigonometric function is the same, so I'll describe one, then give a list of all of them.

RxCalcSin(*value*[, *type*][, *precision*]) → *result*

RxCalcSin takes an angle (*value*) and returns its sine. If *type* is not specified or begins with 'd', *value* is taken to be in degrees; if *type* begins with 'r', *value* is taken to be in radians; and if *type* begins with 'g', *value* is taken to be in gradians. For the inverse functions, *type* determines the type of the return value.

precision over-rides the current setting of numeric digits. It must be a whole number between 1 and 16, inclusive. Note that in this implementation, numeric digits is intended to be checked once at the time MathLoadFuncs is called (but is in fact never checked, since the SAA API doesn't provide a way to do it).

RxCalcSin	*value* is an angle, and the function returns its sine;
RxCalcArcSin	*value* is a number in the range $-1, 1$, and the function returns the angle in the range $-\frac{\pi}{2}, \frac{\pi}{2}$ whose sine is *value* (the inverse sine);
RxCalcCos	*value* is an angle, and the function returns its cosine;
RxCalcArcCos	*value* is a number in the range $-1, 1$ and the function returns its inverse cosine in the range $0, \pi$;
RxCalcTan	*value* is a number which is not a multiple of π, and the function returns its tangent;
RxCalcArcTan	*value* is a number, and the function returns its inverse tangent in the range $-\frac{\pi}{2}, \frac{\pi}{2}$;
RxCalcCotan	*value* is a number which is not a multiple of $\frac{\pi}{2}$, and the function returns its cotangent.

3.2 Exponential Functions

The natural logarithm is the function $\log x = \int_1^x \frac{1}{t} dt, x > 0$. It is the anti-derivative of $\frac{1}{x}$. The natural exponential function is its inverse function. It turns out that the natural exponential function is e^x, where $e = \lim_{h \to 0}(1+h)^{\frac{1}{h}}$. The exciting thing about the natural exponential function is that it is its own derivative.

Most of the exponential and logarithmic functions follow this calling sequence:

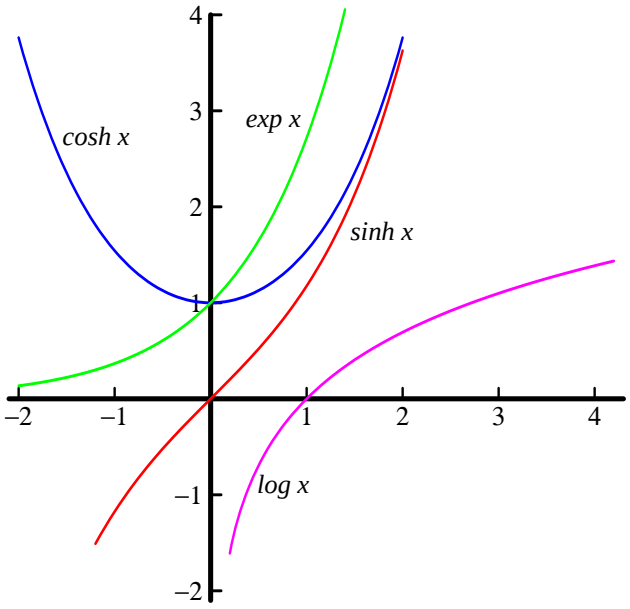
RxCalcExp(*value*[, *precision*]) → *result*

RxCalcExp takes *value*, a real number, and returns the natural exponent of the number. As with the trigonometric functions, *precision* can be used to override the numeric digits setting.

RxCalcExp	*value* is a real number, and the function returns its natural exponent;
RxCalcLog	*value* is a positive real number, and the function returns its natural logaritm;
RxCalcLog10	*value* is a positive real number, and the function returns its base-10 logaritm. Logarithms for other bases can be obtained by dividing by 10 and multiplying by the other number. *e.g.*, log2(*value*) is given by .2 * RxCalcLog10(*value*);
RxCalcSqrt	*value* is a positive real number, and the function returns its positive square root.

RxCalcPower(*base*, *exponent*[, *precision*]) → *result*

RxCalcPower returns *base* raised to the *exponent*th power. The standard Rexx exponential operator ** accepts only whole-number exponents, but RxCalcPower accepts any real number for both the base and the exponent.



3.3 Hyperbolic Functions

Hyperbolic sine and cosine are the functions $\sinh x = \frac{e^x - e^{-x}}{2}$ and $\cosh x = \frac{e^x + e^{-x}}{2}$. The reason for the names is that $(\cos t, \sin t), 0 \le t \le 2\pi$ defines the circle $x^2 + y^2 = 1$, while $\cosh t, \sinh t), t \in \mathbf{R}$ defines one branch of the hyperbola $x^2 - y^2 = 1$. Hyperbolic cosine is used in calculating things like sag and motion through resistive media. I don't know what the other hyperbolic functions are good for, but they're defined similarly to the trigonometric functions.

RxCalcSinH(*value*[, *precision*]) → *result*

RxCalcCosH(*value*[, *precision*]) → *result*

RxCalcTanH(*value*[, *precision*]) → *result*

RxCalcSinH returns the hyperbolic sine of *value* to *precision* digits. Similarly, RxCalcCosH returns the hyperbolic cosine, and RxCalcTanH returns the hyperbolic tangent ($\frac{\sinh x}{\cosh x}$).

4 AMath Library

As I mentioned in the introduction, the amiga math library is based on an almost anonymous newsgroup posting (reproduced in amigamath.txt), which describes a math library for Amiga Rexx. The function names were generally taken from the C math library, which probably got them from a ForTran math library. This is by way of saying that there isn't a naming convention.

I don't know if they are compatible with the Amiga library, but they've been on offer for a few years without any bug reports, so they can't be all bad [in testing the library for this release, I discovered that a few functions simply couldn't be called, so I guess it's not that great after all].

4.1 Trigonometric Functions

Please see section 3.1 for a general discussion of the trigonometric functions. The arguments to the amath trigonometric functions and the return codes of the inverse trigonometric functions are all in radians. Most of them follow this calling sequence:

Sin(*value*) → *result*

Sin	*value* is an angle, and the function returns its sine;
ASin	*value* is a number in the range $-1, 1$, and the function returns the angle in the range $-\frac{\pi}{2}, \frac{\pi}{2}$ whose sine is *value* (the inverse sine);
Cos	*value* is an angle, and the function returns its cosine;
ACos	*value* is a number in the range $-1, 1$ and the function returns its inverse cosine in the range $0, \pi$;
Tan	*value* is a number which is not a multiple of π, and the function returns its tangent;
CoT	*value* is a number which is not a multiple of $\frac{\pi}{2}$, and the function returns its cotangent.
CoTan	synonym for cot
CSc	*value* is an angle, and the function returns its cosecant;
Sec	*value* is an angle, and the function returns its secant;

The arctangent function has a slightly different syntax:

ATan(*value*[,*othervalue*) → *result*

If *othervalue* is not specified, atan returns the inverse tangent of *value* in the range $-\frac{\pi}{2}, \frac{\pi}{2}$. If *othervalue* is specified, atan returns the inverse tangent of $\frac{value}{othervalue}$ in the range indicated by the signs of the arguments. If *value* is positive and *othervalue* is negative, $\frac{\pi}{2} \leq result \leq \pi$.

4.2 Exponential Functions

Please see 3.2 for an interesting and informative disquisition on the origins of the exponential functions.[1]

The exponential functions have the same prototype as all the other functions:

Exp(*value*) → *result*

[1] But look no further than this section to find the word 'disquisition', possibly making its first appearance in any software manual. Professional software companies actually pay people to remove words like that from their manuals. I can't afford to do that.

Exp	*value* is a real number and the function returns its natural exponent;
Log	*value* is a positive real number, and the function returns its natural logaritm;
Log10	*value* is a positive real number, and the function returns its base-10 logaritm. Logarithms for other bases can be obtained by dividing by 10 and multiplying by the other number. *e.g.*, log2(*value*) is given by .2 * RxCalcLog10(*value*);
Sqrt	*value* is a positive real number, and the function returns its positive square root.

Pow(*base*, *exponent*) → *result*

Pow returns *base* raised to the *exponent*th power. The standard Rexx exponential operator ** accepts only whole-number exponents, but pow accepts any real number for both the base and the exponent. It has two synonyms: Power and XtoY.

4.3 Hyperbolic Functions

The AMath library has the same hyperbolic functions described in 3.3, but it adds inverse hyperbolics. The formulae for these are:

$$\text{asinh}x = \ln\left(x + \sqrt{x^2+1}\right)$$

$$\text{acosh}x = \ln\left(x + \sqrt{x^2-1}\right), x \geq 1$$

$$\text{atanh}x = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right), -1 < x < 1$$

SinH	*value* is a real number, and the function returns its hyperbolic sine;
ASinH	*value* is a real number, and the function returns its inverse hyberbolic sine;
CosH	*value* is a real number, and the function returns its hyperbolic cosine;
ACosH	*value* is a real number greater than or equal to 1, and the function returns its inverse hyperbolic cosine;
TanH	*value* is a real number and the function returns its hyperbolic tangent;
ATanH	*value* is a real number with absolute value less than 1 and the function returns its inverse hyperbolic tangent.

4.4 Numerical Functions

The numerical functions are just miscellaneous functions, but calling them numerical sounds more like they were thought out carefully. As with almost every other function in this library, the calling sequence is:

Ceil(*value*) → *result*

Ceil	*value* is a real number and the function returns the smallest integer greater than *value*;
Floor	*value* is a real number and the function returns the largest integer greater than *value*;
Int	Synonym for Floor;
NInt	*value* is a real number and the function returns nearest integer;
Fact	*value* is an integer and the function returns its factorial.

Index