Computing the Arctangent

(in a reasonable amount of time)

by

Mark C. Hendricks, Ph.D.

(Revised, May 19, 2011)

Abstract:

The arctangent function, although easy to calculate, has historically been very slow to converge, especially near its limits of ±1. This has led to many complex methods to calculate $\pi = 4 \arctan(1)$, due to the slowness of the arctangent function. This paper will demonstrate a new technique for calculating the arctangent function for long-precision numbers with greatly improved speed – around 1 second for 10000 digits over the entire range of the function.

Computing the arctangent (in a reasonable amount of time)

The arctangent can be calculated using the formula:

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots \quad \text{for } |x| \le 1 \qquad (1)$$

This series, although accurate, converges very slowly for all but extremely small numbers. For most of the area of interest, the computation time is ridiculously long, becoming quite large as *x* approaches 1 (see table 1, below).

Table 1. Time to calculate arctangent of a 10000-digit number to 10000 digits.

x	time	*x*	time	*x*	time
.00122222…	13.3 sec.	.01222222…	20.3 sec.	.11111111…	**40.6 sec.**
.00322222…	15.5 sec.	.03222222…	26.0 sec.	.31111111…	**76.3 sec.**
.00522222…	17.0 sec.	.05222222…	30.2 sec.	.51111111…	**132.8 sec.**
.00722222…	18.1 sec.	.07222222…	33.9 sec.	.71111111…	**261.3 sec.**
.00922222…	19.0 sec.	.09222222…	37.4 sec.	.91111111…	**958.1 sec.**

These times are shown in Figure 1, curve (1).

The computation time increases with the number of digits in *x*, and with the precision to which we compute the arctangent. In this paper, we will be dealing with numbers with a length of 10000 digits and computing arctangents to 10000 digits.

Our goal is to get the computation time down to approximately one second for all values of *x*, with a floating point length of 10000.

Increasing Computation Speed.

The power series converges more quickly for short numbers (the smallest group we can handle is 8 digits), and for small numbers (numbers with many zeros following the decimal point). In the ideal case, we could split *x* into two parts, one short and one small, e.g., .1234567890123456789012… = .12345678 + .000000009012345678890012…

Unfortunately, the formula for computing the sum of two arctangents does not directly give the arctangent of the sum. The formula

$$\arctan u + \arctan v = \arctan\left(\frac{u+v}{1-uv}\right) \qquad (2)$$

does not give the results we want. However if we let $x = \left(\frac{u+v}{1-uv}\right)$ in equation (2), and solve for *v*, we get

$$v = \left(\frac{x-u}{1+xu}\right). \qquad (3)$$

Re-writing equation 2, we obtain

$$\arctan u + \arctan\left(\frac{x-u}{1+xu}\right) = \arctan x\,. \qquad (4)$$

This gives us the result we are looking for, which is a way to compute the arctangent of a part of the argument [u] and another [much smaller] part [v, from equation (3)], and sum them to get the arctangent of x. Note that the long division involved in calculating v must be carried out to at least the desired floating-point length.

Example 1.

To illustrate the value of this technique, let u = the first 8 digits of x, and evaluate equation (4):

x = .1234567890123456789012…
u = .12345678

$$\arctan x = \arctan(.12345678) + \arctan\left(\frac{.00000000901234567890 12...}{1+.12345678901234567890 12...\times.12345678}\right).$$

While this appears more complicated and does involve one long division, the result can be computed in much less time, because:

- u is short (only 8 digits), so arctan u will compute faster,
- v is *much* smaller and will converge much faster.

Assuming x has 10000 significant digits, arctan x will compute in 5.28 seconds, as compared to 42.6 seconds doing it the old way, and this includes the time for the long divide:

Time to compute arctan u	.373 sec
Time to compute v (long divide)	.076 sec
Time to compute arctan v	4.810 sec
Total	5.283 sec

Curve (2) of Figure 1 shows the time improvement using this method.

The accuracy is the same (10000 digits) but the time has been shortened by a factor of 8. The time, over 5 seconds, is still too long.

Repeating the process

In the above example, the time to compute arctan v is the largest portion of the total time. It turns out that that time is nearly constant over the entire range of the function [the time to compute arctan u will be addressed later].

If we were able to shorten the time to compute the entire arctangent by splitting the number into two parts, could we apply the same process to compute arctan v? The answer is yes. The process may be repeated several times, with varying degrees of improvement.

If we define $x_1 = v$, from equation (3), choose a new u_1 and define v_1 according to equation (3), then substituting into equation (4) we get

$$\arctan v = \arctan x_1 = \arctan u_1 + \arctan v_1, \quad \text{or}$$
$$\arctan u_1 + \arctan\left(\frac{x_1 - u_1}{1 + x_1 u_1}\right) = \arctan x_1 = \arctan v. \qquad (5)$$

Now we know that v is <.00000001 (has at least 8 leading zeros). If we choose u_1 to be the first 32 digits of v, u_1 will also have 8 leading zeros and be 32 digits long. This will converge faster than arctan v, and the new v_1 will have at least 32 leading zeros.

Example 1 (continued)

Continuing with the example above, let u_1 = the first 32 digits of v, and calculate the arctan of u_1 and v_1, adding the results together and with the arctan u. The time now looks like :

Time to compute arctan u	.377 sec.
Time to compute v (long divide)	.076 sec.
Time to compute arctan u_1	.091 sec
Time to compute v_1 (long divide)	.069 sec
Time to compute arctan v_1	1.191 sec
Total	1.806 sec

The time to compute v, v_1, arctan u_1 and arctan v_1 are nearly constant at about 1430 msec throughout the range of numbers.

Curve (3) of Figure 1 shows the time improvement using 2 steps of this process (3 partial arctangents).

How Far Can We Go?

The process can be repeated as many times as you like, but after about 4 iterations, or 5 partial arctangents, the time increase due to the extra v_n calculation exceeds the time saved by breaking up the arctangent calculations.

The break points can be set anywhere as long as they are multiples of 8, the digits in one "block". Breaking the number again after n_2 digits gives a u_2 with 32 leading zeros and a length of n_2, and so on. The optimum break points were determined by experiment to be 8, 24, 96, and 400. (see Table 2.)

Table 2: Break points for arctangent calculation.

term number	break point	length of v	leading zeros in v	significant digits in v
1	8	8	0	8
2	24	24	8	16
3	96	96	24	72
4	400	400	96	304
5	(max)	10000	400	9600

Example 1 (conclusion)

Using the breakpoints given above, the results are:

Time to compute arctan u	.379 sec.
Time to compute v (long divide)	.076 sec.
Time to compute arctan u_1	.069 sec
Time to compute v_1 (long divide)	.069 sec
Time to compute arctan u_2	.075 sec.
Time to compute v_2 (long divide)	.069 sec.
Time to compute arctan u_3	.067 sec
Time to compute v_3 (long divide)	.070 sec
Time to compute arctan v_3	.099 sec
Total	.974 sec

Curves (4) and (5) of Figure 1 show the improved times for 4 and 5 partial arctangents.

The 5-step process yields results under one second for $x < 0.2$. The largest variable remaining is the time to compute arctan u. As u approaches 1, this time can still become large -- as much as 11 seconds for $u = 0.9$.

What can we do about the first term?

Once again, we bump up against the problem that the arctangent converges very slowly as x approaches 1. No matter what we do to the rest of the number, we can't split u_1 up that way, and even if we could, there would be no improvement since the first part of the Term would still be very close to 1.

However, there is another way to force the value of x to be in a reasonable range. Looking again at equation (4), if u is 0.1, then

$$\arctan x = \arctan 0.1 + \arctan\left(\frac{x-0.1}{1+0.1x}\right) \qquad (6)$$

If we calculate $v = \left(\frac{x - 0.1}{1 + 0.1x} \right)$,

then v is closer to zero (and farther from 1) and will converge faster.

The number 0.1 was chosen by balancing the additional overhead of recalculating v_n against the time saved by reducing the argument by the arctangent of that number. Also, arctan 0.1 is easy to calculate and converges quickly (about 200 msec) and we can use it multiple times, as will be shown later.

We can apply the function repetitively: $v_n = \left(\frac{v_{n-1} - 0.1}{1 + 0.1v_{n-1}} \right)$, and eventually end up with a v_n that is less than 0.1. This will require several long divides at about 72 msec each, but will leave us with a number less than 0.1 to calculate as shown in Example 1. The final result will need to have arctan 0.1 added in as many times as we subtracted 0.1 (and adjusted) to reduce v_n to less than .1. The add back takes, typically, about .05 msec. This process will have no effect on numbers < .1

In the example, if we first apply the procedure of adjusting the number by subtracting arctan .1, the total time becomes 1079 msec. However that includes the 200 msec to calculate arctan .1, which only needs to be done once and is saved for future arctangent calculations. Excluding that number, times for various numbers between .1 and 1 become:

0.123457…	879	msec
0.223457…	977	
0.323457…	1022	
0.423457…	1011	
0.523457…	1232	
0.623457…	1259	
0.723457…	1285	
0.823457…	1421	
0.923457…	1416	

These results are shown on Figure 1, curve (6).

These times are, at least, well-behaved in the vicinity of x=1, and are far closer to our goal of one second. Remember, all arctangents are carried out to 10000 digits, and all values of x here are 10000 significant digits long.

Reducing the argument in one step.

The problem with the above approach is that the value x has to be reduced up to 7 times, depending on how close it is to 1. Each reduction adds about 80 msec to the computation time. In this section, I will show how that can be done in one step.

We would like to obtain a set of numbers w_n between .1 and 1, whose arctangents are known without having to calculate those arctangents. Then equation (6) could be re-written as

$$\arctan x = \arctan w_n + \arctan\left(\frac{x - w_n}{1 + x w_n}\right).$$

If w_n is close to (but still less than) x, we can calculate

$$v_n = \arctan\left(\frac{x - w_n}{1 + x w_n}\right) \text{ in one step.}$$

The following shows how we determine the values for w_n.

Recalling equation (1) for the sum of two arctangents,

$$\arctan u + \arctan v = \arctan\left(\frac{u + v}{1 - uv}\right), \qquad (1)$$

let $w = \dfrac{u + v}{1 - uv}$, then

$$\arctan u + \arctan v = \arctan w \qquad (7)$$

Now if we assume that <u>u and v are rational numbers</u>, such that $u = \dfrac{a}{b}$ and $v = \dfrac{c}{d}$,

then $$w = \left(\frac{\frac{a}{b} + \frac{c}{d}}{1 - \frac{ac}{bd}}\right) = \left(\frac{ad + bc}{bd - ac}\right).$$

So w is also a rational number.

Equation (7) becomes

$$\arctan\left(\frac{a}{b}\right) + \arctan\left(\frac{c}{d}\right) = \arctan\left(\frac{ad + bc}{bd - ac}\right) \qquad (8)$$

. If we define

$e = ad + bc$ (8a)
$f = bd - ac$ (8b)

then e and f are both integers and $w=e/f$ is a rational number. Equation (8) can be written as

$$\arctan\left(\frac{a}{b}\right)+\arctan\left(\frac{c}{d}\right)=\arctan\left(\frac{e}{f}\right). \qquad (9)$$

Since we have already calculated arctan (.1), we would like to have arctan w_n be a multiple of arctan (.1). In other words, we want to find w_n such that

$$\arctan w_n = n\times\arctan(.1) \qquad (10)$$

It follows that

$$\arctan w_{n-1} = (n-1)\times\arctan(.1)$$
$$\arctan w_n - \arctan w_{n-1} = n\times\arctan(.1)-(n-1)\times\arctan(.1)$$
$$\arctan w_n - \arctan w_{n-1} = \arctan(.1)$$
$$\arctan w_n = \arctan w_{n-1} + \arctan(.1) \qquad (11)$$

Substituting $w_n = e_n / f_n$

$$\arctan\left(\frac{e_n}{f_n}\right)=\arctan\left(\frac{e_{n-1}}{f_{n-1}}\right)+\arctan\left(\frac{1}{10}\right) \qquad (12)$$

If we let

$a = e_{n-1}$
$b = f_{n-1}$
$c = 1$
$d = 10$

then according to the definitions (8a and 8b) we can calculate e_n and f_n

$$e_n = e_{n-1}\times 10 + f_{n-1}\times 1$$
$$e_n = 10\times e_{n-1} + f_{n-1} \qquad (13)$$

$$f_n = f_{n-1}\times 10 - e_{n-1}\times 1$$
$$f_n = 10\times f_{n-1} - e_{n-1} \qquad (14)$$

Starting with $e_1 = 1$ and $f_1 = 10$ (that is, $w_1 = .1$), we now calculate e_n and f_n using equations (13) and (14) for the first 7 values, which are shown in table 3, below.

Table 3.

n	e_n	f_n	$w_n = e_n/f_n$ (approx)
1	1	10	.100000000
2	20	99	.202020202
3	299	970	.308247422
4	3960	9401	.421231783
5	49001	90050	.544153248
6	580060	851499	.681222174
7	6652099	7934930	.838331151

The values for w_n are intentionally kept as rationals (e_n/f_n) in order to avoid calculating them to high precision.

The values of e_n, f_n and w_n are pre-calculated and stored in a table in the program. w_n is only stored to 8 digits.

There is no need to go beyond $n = 7$, since the value of w_n would then be greater then 1.

From equation (4) we get

$$\arctan x = \arctan w_n + \arctan v, \quad \text{where}$$

$$v = \left(\frac{x - w_n}{1 + x w_n} \right) = \left(\frac{x - \frac{e_n}{f_n}}{1 + \frac{x e_n}{f_n}} \right)$$

$$v = \frac{x f_n - e_n}{f_n + x e_n} \qquad (15)$$

Substituting $\arctan w_n = n \times \arctan(.1)$, we get

$$\arctan x = \arctan v + n \times \arctan(.1) \qquad (16)$$

where v is calculated from equation (15).

The process then becomes:

1. Choose the largest value of w_n (from the last column in table 3) which is less than x,
2. Calculate arctan v = $\arctan\left(\frac{xf_n - e_n}{f_n + xe_n}\right)$, (17)
3. Add arctan (.1) n times.

It can be shown (see below) that v is < .1, whose arctangent will calculate quickly, somewhere around 900 milliseconds.

In step 1, it is only necessary to compare the first 8 digits. Any error would simply change the value of x where we switch to a different n. This would affect the point where the timing improves, but would not affect the accuracy of the calculation.

Note that e_n and f_n are integers less than 8 digits, which means that the numerator and denominator of equation (17) can be calculated with a quick short (8-digit) multiply and addition or subtraction. Only one long division is required.

Proving that v < .1

If n is chosen from table 3 such that $w_n < x < w_{n+1}$, then

$$x_{\max} = \frac{e_{n+1}}{f_{n+1}} = \frac{10e_n + f_n}{10f_n - e_n}$$

$$v_{\max} = \frac{x_{\max} f_n - e_n}{f_n + x_{\max} e_n}$$

$$= \frac{\frac{10e_n + f_n}{10f_n - e_n} f_n - e_n}{f_n + \frac{10e_n + f_n}{10f_n - e_n} e_n}$$

$$= \frac{f_n(10e_n + f_n) - e_n(10f_n - e_n)}{f_n(10f_n - e_n) + (10e_n + f_n)e_n}$$

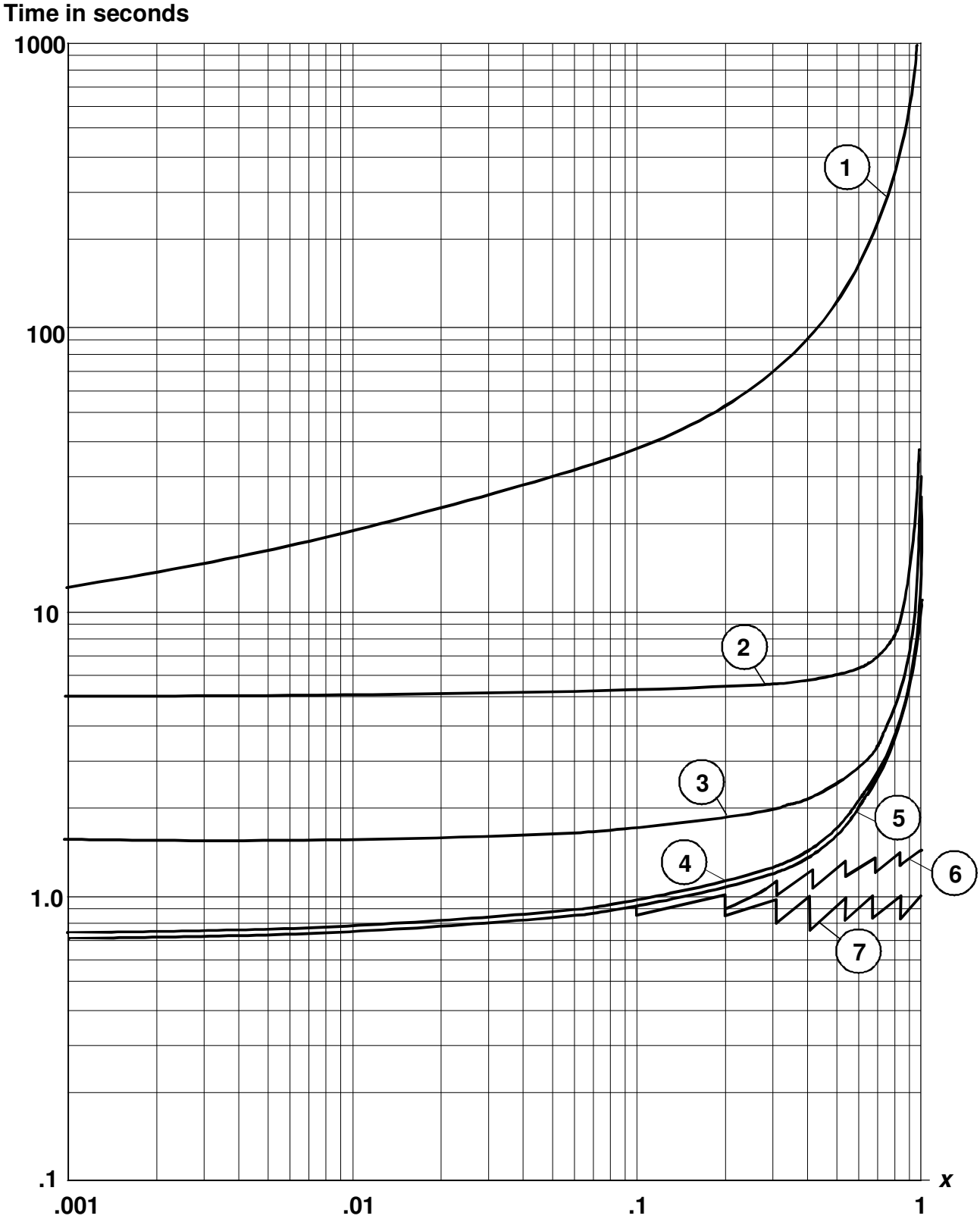
$$= \frac{10e_n f_n + f_n^2 - 10e_n f_n + e_n^2}{10f_n^2 - e_n f_n + 10e_n^2 + e_n f_n}$$

$$= \frac{\left(f_n^2 + e_n^2\right)}{10\left(f_n^2 + e_n^2\right)} = \frac{1}{10}$$

Example 1 (Extra Credit)

Applying the procedure described above to reduce the value of x to something less than 0.1, and using the 5-step arctangent calculation, the times are now kept near or below 1 second for all values of x. The final results are shown in Figure 1 curve (7).

Figure 1 - Time to Compute Arctangent of 10000-Digit Number to 10000 digits



Speed for other lengths

The calculations in this paper have been done at lengths and precision of 10000 digits. Time for these calculations is on the order of $(\text{digits})^2$. So, based on that knowledge and some experimental results, you can expect the following:

digits	Arctan time (expected)	Arctan Time (measured)
100	100 μsec	150 μsec
500	2.5 msec	2.6 msec
1000	10 msec	10 msec
5000	250 msec	230 msec
10000	1 sec	1 sec
20000	4 sec	4 sec
50000	25 sec	35 sec
100000	100 sec	200 sec

These routines were optimized for 10000 digits. At around 100, the overhead due to copying, allocating, freeing, initializing, etc., becomes significant compared to the time dividing and evaluating power series. Still, the numbers are quite impressive.

At 100000, the improvement from breaking the number apart could benefit from another breakpoint and a sixth partial arctangent. Setting the 5th breakpoint to 4000 brings the total time for 100000 down to around 120 seconds, and for 50000 down to 30 seconds.

This suggests that a truly optimized routine might have to dynamically set the breakpoints based on the length of the number. There is always room for improvement.

What about arctangents of numbers > 1, or negative numbers?

There is nothing new about those processes.

Numbers greater then 1 will not converge using the power series, however, we can use the identity

$$\arctan x = \frac{\pi}{2} - \arctan \frac{1}{x}$$

to calculate the complement of the angle and then subtract it from π/2. This adds about 80 milliseconds for the long divide, and assumes you have already calculated π, which takes about 600 milliseconds for a length of 10000. Very large numbers will lose precision because that means subtracting a very small number from π/2, and it will not be possible to regenerate the original number to full precision using the tangent function.

The arctangent of a negative number is the negative of the arctangent of the absolute value of the number, so we calculate the arctangent of the positive value and change the sign at the end.

Notes on Programming Techniques

Long numbers are maintained in a format which places 8 decimal digits in binary in a 32-bit word, or "block". Some assembler-language routines are used to allow multiplication and division on these 32-bit words. Fast long multiply and divide routines are used which take advantage of this 32-bit arithmetic, and when possible, 32-bit integer multiplication or division is used.

Calculation is done at a precision of 8 longer than the floating-point length. Conversions required in order to reduce the argument to a number which converges more quickly could introduce round-off errors, so the extra precision takes care of that. If the angle is required in degrees, then it must be multiplied by $180/\pi$. This calculation is also done at the precision of 8 extra digits, before rounding to the desired length. Power series calculations are done with an additional 10 digits of precision (rounded up to the next "block" of 8).

Timing measurements are done using the Windows functions:

QueryPerformanceFrequency ( ) and QueryPerformanceCounter ( ),

which allow timing on the order of the system clock, although operating system overhead often makes it difficult to get consistent results. The timing measurements are mostly used to determine which parts of the process are consuming the most time, and to assess overall performance.

Power series calculations are performed using a fixed number of digits to the right of the decimal point. This can result in a loss of precision if the argument contains many leading zeros, so for such numbers the length of the calculation is extended by the number of leading zeros. Numbers containing more leading zeros than half the floating-point length will see no 2^{nd} term in the expansion, and so arctan x is just x.

Reference Computer

The computer used for these tests is a Sony VGC-RB40 with a 3.0 GHz Pentium D processor and 1 GB of memory, running Windows XP. No other programs were running during the timing tests. The program does not take advantage of dual-processor support. Other computers will, of course, get different results, but the speed will be proportional to your processor clock speed. A processor with 32-bit arithmetic is required.

Conclusion

This paper has demonstrated how the arctangent, normally a very slow-converging function, may be calculated in a much shorter time using the processes described, with no loss of accuracy. Calculation times of around 1 second for 10000 digits have been achieved over the entire range of the function.

Although the code may not port directly to other types of computers, the techniques described here could be adapted to speed up calculation on any platform.