The Practical Use of the Bemer Method for Exponentials

Update Version: September 5, 2006
Ron Doerfler (http://www.myreckonings.com)

In Chapter 4 of my book, ***Dead Reckoning: Calculating Without Instruments***, a method is given for mentally calculating exponentials, i.e., raising a number to a power where neither the number or power is typically an integer. An example of this is calculating $10^{2.85251} = 712.049\ldots$ This operation 10^n is often called an antilogarithm, which is the terminology used in the book. This paper provides some additional tips that in practice can considerably lessen the effort when using the Bemer method in the book to mentally calculate exponentials. The complete algorithm is presented here; there is no need to have read the book.

One application of calculating exponentials is in finding a higher-order root **r** of a number **N**. First, the common, or base 10, logarithm of **N** is found (through one of the other methods given in the book) and divided by the order of the root. The answer is then 10 raised to that result, or $10^{\log(N)/r}$. This could be done using any base, so for example we could also have calculated $e^{\ln(N)/r}$, where **e** is the famous mathematical constant and ln(N) is the natural logarithm of **N**, or the logarithm to base **e**. Separately, e^n is often a solution to mathematical or physical equations, so it is useful to be able to calculate this function directly.

However, e^n can be represented as $10^{(0.4343\ldots)\,n}$, where $0.4343\ldots = \log(e)$, so having a convenient method for mentally calculating 10^n provides, with one multiplication, a corresponding method for calculating e^n or any other number raised to a power. In practice, it is usually easier to raise 10 to a power, since we can extract all integer powers of 10 as a first step, so the book and this paper deal directly with this calculation.

In 1958, R.W. Bemer published a method of calculating logarithms for use in early decimal computers.[1] While other algorithms covered in the book are more suited to mental calculations of logarithms themselves, the inverse of Bemer's method lends itself very well to mental calculations of exponentials, generally a more difficult thing to do. Let's discuss the straightforward use of Bemer's method for exponentials before delving into strategies to optimize its use for mental calculation.

The Straightforward Use of the Bemer Method

The first step in finding 10^n is to eliminate digits to the left of the decimal point. For example, if $n = 2.85251$, we convert $10^{2.85251}$ to $10^2 \times 10^{.85251}$ instead, and we now need only to find the second term and multiply it by 100.

We can use the following series to find that second term:

$$e^x = 1 + x + x^2/2 + \ldots$$

Since $e^x = 10^{x/\log(e)}$ and $1/\log(e) = 2.3026$, we can convert this to:

$$10^x = 1 + 2.303x + (2.303x)^2/2 \quad \text{approximately}$$

We have eliminated higher-order terms in the series, so obviously this approximation is only accurate if **x** is small. The general approach is to subtract convenient powers of 10 from **n** in order to leave a small enough **x** to get an accurate approximation from this formula. For example, if x = 0.1, the error in this approximation is 0.002. We are striving here for five digit accuracy in finding 10^n, and for that we need to have x < 0.019. The squared term in this formula can be calculated to a low number of digits (two digits for x = 0.019), and we have options for simplifying or eliminating this term that will be presented later, so this is not as impossible as it might seem at this time.

The powers of 10 that we will subtract are those that produce simple fractions to use as multipliers: 11/2, 11/3, 11/4, … 11/10. Bemer chose these particular fractions to multiply logarithms to get them near 1 in order to use a formula for log(1+x), as described in the book or reference [1], but they turn out to be convenient for our inverse operation 10^x as well.

In our example, we need to find $10^{.85251}$ after the digits to the right of the decimal point are extracted, but x = .85251 is not nearly small enough for our approximation. But 11/2 = $10^{.74036}$, so we can let $10^{.85251}$ = 11/2 x $10^{.11215}$, where .11215 = .85251 - .74036. This decreases x significantly, although it is still far outside the range for five digit accuracy. We can repeat this process for other fractions, based on Table 1.

Multiplier M	Power of 10 = log(M)
11/2	.74036
11/3	.56427
11/4	.43933
11/5	.34242
11/6	.26324
11/7	.19629
11/8	.13830
11/9	.08715
11/10	.04139

Table 1. Multipliers and Log Values in the Bemer Algorithm.

Yes, we have to memorize this table in order to use the Bemer method. Actually, it is useful in finding logarithms to have memorized the logs of all integers less than 10, so if one of the entries in Table 1 is forgotten, the power of 10 associated with 11/m can be derived from log(11) – log(m), where log(11) = 1.04139. As a convenience, I've listed the logarithms of single digits in Table 2.

N	log(N)
1	0
2	.30103
3	.47712
4	.60206 = 2 x log(2)
5	.69897 = 1 – log(2)
6	.77815 = log(2) + log(3)
7	.84510
8	.90309 = 3 x log(2)
9	.95424 = 2 x log(3)

Table 2. Logarithms of Single-Digit Integers.

Subtracting these logarithms of single digits from log(11) provides values that have finer spacing near 0 rather than near 1, so we are able to subtract the differences from **x** to approach very close to 0. However, logarithms of single digits are much easier to remember due to the relationships between the logarithms for single digits that are given in the last table. With this advantage, and given that it is very useful to know these logs anyway when calculating other logarithms, it seems a shame that we do not use these directly. In fact, John McIntosh presents a scheme for exponentials that only uses log(2) and log(3) to provide a similar, very nice algorithm for exponentials.[2] I recommend reading his paper as well and comparing methods to find the one you prefer. John also provides a much more detailed error analysis of the approximation formula.

So we currently have $10^{.85251} = 11/2 \times 10^{.11215}$, and we can now subtract .08715 from the power by using the 11/9 multiplier, giving $10^{.85251} = 11/2 \times 11/9 \times 10^{.025}$. Using the approximation formula:

$2.303 \times .025 = .05758$
$.05758^2/2$ is approximately .00166
$1 + .05758 + .00166 = 1.05924$

So $10^{.85251} = 11/2 \times 11/9 \times 1.05924$

This is not a lot of fun to calculate, but one approach is to divide 1.05924 by 9 and 2 (or 18 directly), and then multiply the result by 11, then 11 again, remembering that $11y = 10y + y$. In the end, we get $10^{.85251} = 7.12045$, accurate to 5 digits to the actual value of 7.12049 even though the final value used in the approximation formula was a bit high to ensure this accuracy.

However, there are modifications to the Bemer method that we can use to simplify the algorithm in practice, and this is the purpose of this paper. We will consider these in the next section.

Practical Simplifications to the Bemer Method

The first strategy to simplify the Bemer method for mental calculation is to allow the use of negative numbers! We discussed earlier that digits to the left of the decimal point are removed by extracting 10 to that power. However, if the digit after the decimal point in the original exponent is greater than .5, we can extract one more power of 10 to leave a value between between -.5 and +.5. This halves the initial range to approximate, and it means that the log of the multipliers 11/2 and 11/3 are no longer needed, leaving only 7 values to memorize.

Now in intermediate steps where multipliers are chosen, we can also allow the log values to be either added or subtracted, and the resulting exponents to be either positive or negative. This is a very powerful technique for two reasons. First, we can halve the maximum final remainder used in the approximation formula without having to memorize another multiplier value. The lowest log value given in Table 2 is .04139, which in the straightforward algorithm is the maximum final remainder. Now we can subtract this value if needed to get a final remainder between -.02070 and +.02070, which brings this maximum very close indeed to that needed for a result accurate to five digits.

The other advantage of being able to add or subtract the log values is that adding flips the 11/n multiplier fraction--and this means that the 11's cancel out between successive additions and subtractions! This is where the unequal spacings of the log values in the Bemer method are a real advantage, because we can "spiral" around zero as we approach it by adding and subtracting log values, first greater than zero, then less than zero but closer, etc. This effectively doubles the resolution of the spacings between log values as well. It is mentioned in the book that the log values can be either added or subtracted, but these advantages are not explicitly described, mostly because I had not fully appreciated them at the time.

Since a given exponential can result in an odd number of multipliers rather than an even number of multipliers, we end up being able to utilize the convenient fractions 11/4, 11/5, etc., plus their inverses, plus all the ratios of combinations of 4, 5, 6, etc. including their inverses, by just knowing 7 log values.

Another area for improvement is the final approximation formula. This is a difficult step, but there are things we can do to ease it. First, we can use a technique described in the book. Here we eliminate the squared term completely and truncate the 2.303 multiplier to simply 2.3 instead. Then the formula becomes:

$10^x = 1 + 2.3x$ approximately

In order to use this formula for the larger **x** values in the final range -.02 to +.02 while retaining our desired accuracy, we need to add a correction at the end. Let's say that we are approximating $10^{.013}$ in the last step. We would begin with the formula above to get 1 + 2.3(.013) = 1.0299. For the correction, let the power of 10 be represented by ±(a+b), where **a** = .01 or .02 only. Then a=.01 and b=.003 here. We now add the term (cb+d), where **c** and **d** are given in Table 3.

Sign	a	c	d
+	.02	.1	.00113
+	.01	.1 / 2	.00029
-	.01	.1 / 2	.00024
-	.02	.1	.00099

Table 3. Correction Values for $10^{\pm(a+b)}$.

So we add (.1/2) x .003 + .00029 = .00044, giving 1.0299 + .00044 = 1.03034 as our approximation for $10^{.013}$, compared to the actual value of 1.03039.

A second option is to use the full formula, but with 2.3023 replacing the 2.303 multiplier:

$10^x = 1 + 2.3023x + (2.3023x)^2/2$ approximately

This is not as radical as it seems, since the original multiplier 2.303 is rounded up from the actual value of 2.30258... The advantage of this new multiplier is that x is multiplied by 2.3, and then that result is shifted right three decimal places and added to itself. For $10^{.013}$ we have:

2.3(.013) = .0299
.0299 + .0000299 = .0299299 or .02993
$.03^2/2 = .00045$
1 + .02993 + .00045 = 1.03038

This result is closer to the actual value of 1.03039, but it does involve a squaring operation. Also, using this smaller multiplier adds to the error from truncating the series. We should add .00001 when **x**>.011 to preserve 5-digit accuracy up to **x**=.017. Either alternative presented above is easier than the original approximation formula.

Finally, be creative and flexible as you solve these problems! Without creativity, mental calculation is drudgery; by applying creativity the calculations stretch our minds, serve as puzzles to challenge us, and offer personal satisfaction with the result. Look at those intermediate exponents as they appear. Is one near the log of a single digit number that you happen to know? If so, use it. Can we use a pair of log values that are not the obvious ones but whose difference is really close to the exponent? Do we happen to know $10^{.1}$ or $10^{.2}$? Can we use fewer digits in the approximation formula, or multiplication shortcuts for the particular numbers? Is it easier to multiply by 2.3 or by (2.2 + .1)? Can we replace a decimal remainder with a fraction that is easier to multiply? Do we really need the squared term in the approximation formula with a small remainder, and if so, can we just roughly estimate it? Do you prefer the algorithm in reference [2] of this paper? Most importantly, can you come up with a better algorithm for mentally calculating exponentials—I'd love to hear it.

Let's finish by repeating our calculation of $10^{2.85251}$ using these new strategies:

$10^{2.85251} = 10^3 \times 10^{-.14749}$
$10^{-.14749} = 7/11 \times 10^{.04880}$ since -.14749 + .19629 = .04880
$= 7/11 \times 11/10 \times 10^{.00741}$ since .04880 - .04139 = .00741

2.3(.00741) = .01704 Shift 3 places and add .000017 to .01704 to get .01706
$.017^2/2 = .00014$
1 + .01706 + .00014 = 1.01720
Adding .00001 since x>.011 gives 1.01721

$10^{2.85251} = 10^3 \times 7/10 \times 1.01721$
$= 10^2 \times 7 \times 1.01721$
= 712.047 compared to the actual value of 712.049…

Or, just maybe, we might have remembered that log(7) = .84510, so:

$10^{2.85251} = 10^2 \times 7 \times 10^{.00741}$

and we could have saved a step or two with this bit of experience.

References:

[1]R.W. Bemer, "A Subroutine Method for Calculating Logarithms," ***Communications of the Association for Computing Machinery, 1*** (1958) pp. 5-7. This paper can also be viewed at http://www.myreckonings.com/Dead_Reckoning/Chapter_4/Materials/p5-bemer.pdf

[2]John McIntosh, "Exponentials". This essay can be read at http://www.urticator.net/essay/6/641.html.