Infinite Sums

Sums and series show up in a multitude of mathematical applications. The sum key on screen 6 can do most finite sums with no more than 100,000 terms fairly quickly, if the terms are not too complicated.

The sum key can also approximate finite sums with more terms, as well as series with infinitely many terms. We will look at several different types of sums.

The sum function has 4 inputs: a, b, h, n, sum

This says to sum function n from a to b with steps of h: $f_n(a) + f_n(a+h) + f_n(a+2h) + \cdots + f_n(b)$

In case these steps don't hit b exactly, the sum stops short of b.

1, enter, 100, enter, 2, enter, 3, sum   computes $f_3(1) + f_3(3) + f_3(5) + \cdots + f_3(99)$

Example 1.   Here is a finite sum:

$$f(n) = \sum_{k=1}^{n} k^2 = 1 + 4 + 9 + \cdots + n^2$$

To evaluate f(1,000) we first define function f1(k) = k^2.

f1: 1, func, x^2

Then sum f1(k) for k from 1 to 1,000 by steps of 1.

−2, fix, 7, func, 1, enter, 1000, enter, 1, enter, 1, sum   333,833,500.00

Since all the terms are small integers in this case, there are no rounding errors and the result is exact. Like the integration key, the sum key also returns an estimated relative error in the y-register. Pressing x↔y to see the error shows 1.0e−55, meaning the result should be accurate to about full precision.

Example 2.   Next do an infinite series. Find S = zeta(2) from the Riemann zeta function:

$$S = \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \frac{1}{5^2} + \cdots$$

This time we can't just add all the terms. Some series converge fast enough that we could keep adding terms until the partial sums stop changing. That won't work here either, since it would take over 10^{50} terms before the partial sums were accurate to 50 digits.

To handle cases like this, the sum key uses some advanced techniques to more quickly approximate the sum. However, they can't be applied to every infinite series. Here are the conditions that must be satisfied before the sum key will try to approximate the sum:

1. The number of terms is greater than 100,000.
2. The signs of the terms are either all the same or strictly alternating.
3. If the signs of the terms are all the same, the step size h must be 1 or -1.

When the signs of the terms are all the same, the function needs to be smooth enough that we can replace the sum to infinity of $f(k)$ by an integral to infinity of $f(x)$ and numerically evaluate the integral.

Like the integrate key, if we want the upper limit to be infinity we enter a number so large that the terms that far out will be completely negligible with respect to the final sum. A number like 10^{9999} usually works.

For entering powers of ten, 10^{9999} will be written "e9999" and entered into the calculator by pressing the keys "eex, 9999". To get a negative value instead, "−e9999" means pressing the keys "1, chs, eex, 9999", and "e−9999" would be "eex, chs, 9999".

Since the zeta sum satisfies the three conditions, we can define $\mathrm{f2}(k) = 1/k^2$ and ask for the infinite sum:

f2: 1, func, 1/x, x^2

40, fix, 7, func, 1, enter, e9999, enter, 1, enter, 2, sum

1.6449340668482264364724151666460251892189

Pressing x↔y to look at the y-register gives the estimated relative error:

10, sci, x↔y 1.000000000e−55

This means the sum function estimates that over 50 significant digits should be correct.

For this example, the true sum is $\pi^2/6$, so we can check to see how many digits were actually correct.

1, func, x↔y, π, x^2, 6, /, − 1.000000000e−56

With this series, adding the first million terms would get only 6 correct significant digits, but the sum key has given over 50-digit accuracy.

Example 3. Do an example with an alternating infinite series. G is Catalan's constant.

$$G = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)^2} = \frac{1}{1^2} - \frac{1}{3^2} + \frac{1}{5^2} - \frac{1}{7^2} + \frac{1}{9^2} + \cdots$$

f3: 1, func, 3, sto, 2, ∗, 1, +, x^2, 1/x, 3, rcl, −1, x↔y, y^x, ∗

40, fix, 7, func, 0, enter, e9999, enter, 1, enter, 3, sum

0.9159655941772190150546035149323841107741

The estimated error is 1.0e−55. Checking with the integrate key by subtracting an alternate form of Catalan's constant, $G = (1/2)\int_0^{\pi/2} x/\sin(x)\,dx$, gives about 6e−57, so we should have more than 50 digits correct.

Example 4. This is a very slowly convergent alternating series.

$$\sum_{k=2}^{\infty} \frac{(-1)^k}{\ln(k)} = \frac{1}{\ln(2)} - \frac{1}{\ln(3)} + \frac{1}{\ln(4)} - \frac{1}{\ln(5)} + \cdots$$

f4: 1, func, 4, sto, ln, 1/x, 4, rcl, −1, x↔y, $\mathrm{y^x}$, ∗

40, fix, 7, func, 2, enter, e9999, enter, 1, enter, 4, sum

0.9242781803271541148940810760031085753714

There is a trap here. $1/\ln(k)$ goes to zero so slowly that the sum to 10^{9999} terms is not a good approximation for the sum to infinity. Check f4(10^{9999}):

e9999, enter, 4, $\mathrm{f_n}$ 0.0000434337915694821309782107129629568039

Even using the biggest value in Calc-50's number system is not big enough. But values that would be bigger than the largest representable number are called "+overflow" in the calculator. The sum key will interpret +overflow as meaning infinity. One way to get overflow is "e99, $\mathrm{e^x}$".

1, func, e99, $\mathrm{e^x}$, 7, func, 2, x↔y, 1, enter, 4, sum

0.9242998972229388559595701813595900537733

That should be an accurate value for the infinite series, and shows that adding "only" 10^{9999} terms agrees with the infinite sum to just 4 significant digits.

Example 5. Many functions can be expressed as power series. Use e^x as an example.

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Define f5(x) to sum this series. It stores x in register 5, then sums f0(k), where f0 computes $x^k/k!$.

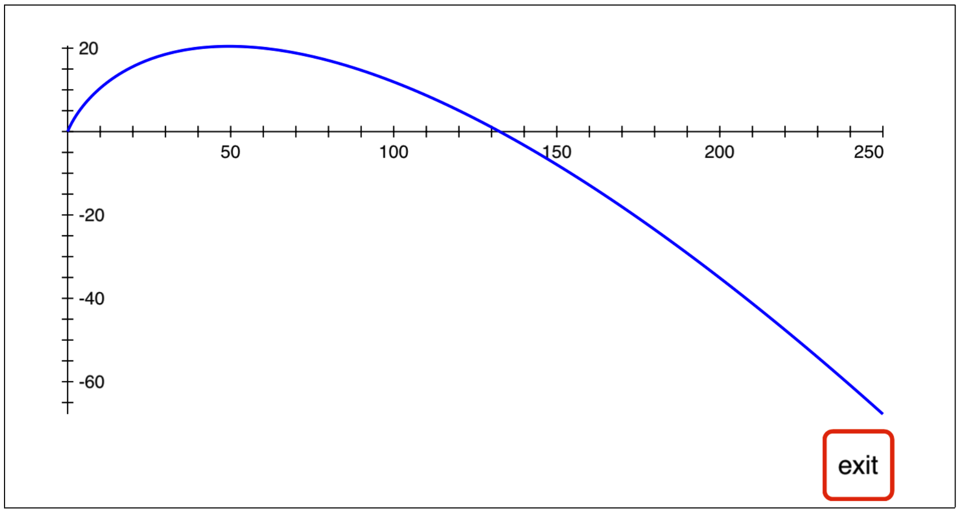
f5: 7, func, 5, sto, 0, enter, e5, enter, 1, enter, 0, sum

f0: 1, func, 0, sto, 5, rcl, x↔y, $\mathrm{y^x}$, 0, rcl, 3, func, x!, /

If we graph the base ten logarithm of f0(k), the y-values will show the power of ten for the k^{th} term in the series. Define f9 to use $x = 50$ and return log(f0(k)).

f9: 50, enter, 5, sto, x↔y, 0, $\mathrm{f_n}$, 5, func, log

7, func, 0, enter, 250, enter, 9, plot



Power series can have different convergence properties for different values of x. For this one, if $|x| > 1$ the terms increase in size for a while before they start to converge.

The graph starts at zero, then rises to a maximum height of about 20 when k is around 50. After that it falls to about -68 when k is 250.

That means when $x = 50$, the terms in the power series get up to about 10^{20} before dropping to 10^{-68} at $k = 250$. This tells us that we can stop adding terms in the sum by the time we get to $k = 250$. At the calculator's 50-digit precision, adding a number smaller than 10^{-65} to a partial sum bigger than one doesn't change the partial sum.

We set the upper limit for the sum to be 10^5 in the f5 definition, because different x-values will take different number of terms. The sum function will detect when the sum has converged to the calculator's precision and stop adding more terms.

To compute e^{50} using its power series,

40, sci, 50, enter, 5, $\mathrm{f_n}$

5.1847055285870724640874533229334853848 27e+21

Check the answer: 10, sci, 1, func, 50, $\mathrm{e^x}$, $-$ 0.000000000e+0

Something different happens if we use $x = -50$ instead.

40, sci, -50, enter, 5, $\mathrm{f_n}$

1.9287498479639177830173428165270126000 00e$-$22

Check the answer: 10, sci, 1, func, -50, $\mathrm{e^x}$, $-$ 2.524716735e$-$57

We are used to thinking that errors under 10^{-50} mean that we have about full precision correct for the calculation. But this is *absolute error*, defined as the absolute value of the true answer minus the computed result.

The problem is that here the true answer is very small, so to have the computed result be accurate to over 50 significant digits we need to have the *relative error* less than 10^{-50}. Relative error is defined as

$$\left|\ (\text{true} - \text{computed})\ /\ \text{true}\ \right| \;=\; \left|\ \text{absolute error}\ /\ \text{true}\ \right|$$

Dividing 2.524716735e−57 by e^{-50} gives the relative error: 1.308991281e−35

So using the series to approximate e^x for $x = -50$ gives a result good to only about 35 significant digits. This loss of precision is caused by cancellation error, subtracting numbers that are nearly equal and getting a result that is much smaller than either.

The biggest individual terms in this sum are about $(-50)^{50}/50! = 2.9\text{e+}20$ in magnitude. But the final result of the sum was 1.9e−22, so many digits have cancelled out. Here is an example in 10-digit arithmetic, where it is easier to visualize the cancellation.

$$x = 123.4567899, \quad y = 123.4555555 \quad \Rightarrow \quad x - y = 0.0012344$$

Both x and y could be accurate to 10 significant digits compared to their respective true values, but since they are nearly equal their leading 5 digits cancelled out, leaving the result of their subtraction accurate to only 5 significant digits.

The sum key tries to estimate the relative error in the final result of the summation, and it returns the estimate in the y-register. After using the sum key, pressing x↔y shows that estimate.

For the e^{-50} sum above, the estimated relative error returned by the sum key is 1.344058571e−34, which agrees fairly well with the 1.3e−35 true relative error we found above.

Example 6. A doubly-infinite series.

$$\sum_{k=-\infty}^{\infty} \frac{k^2+2}{3k^4+k+1} \;=\; \cdots + \frac{11}{241} + \frac{6}{47} + 1 + 2 + \frac{3}{5} + \frac{2}{17} + \frac{11}{247} + \cdots$$

The easiest way to evaluate this is probably to split the sum into two parts and use the sum function twice. One sum will run from $k = 0$ to ∞, and the other from $k = -1$ to $-\infty$ by steps of -1. That way both will have terms that are positive and decreasing.

f6: 1, func, 6, sto, x^2, 2, +, 6, rcl, 4, y$^\text{x}$, 3, ∗, 6, rcl, +, 1, +, /

40, fix, 7, func, 0, enter, e9999, enter, 1, enter, 6, sum, 11, sto

2.8615414182808975417235941638253730693242

−1, enter, −e9999, enter, −1, enter, 6, sum, 12, sto

1.2730445194803128480890610662658550321696

Adding these two parts gives the result:

4.1345859377612103898126552300912281014938

Example 7. A series where the terms are neither all the same sign nor strictly alternating.

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} - \frac{4}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} - \frac{4}{10} + \cdots$$

The pattern that repeats in the signs is four positive terms followed by one negative term. The sum key will not be able to analyze this to get a value for the infinite series.

To make the pattern more regular we can group each set of five terms together, and write this series in a way the sum key can handle.

$$\sum_{k=0}^{\infty} \left(\frac{1}{5k+1} + \frac{1}{5k+2} + \frac{1}{5k+3} + \frac{1}{5k+4} - \frac{4}{5k+5} \right)$$

$$= \left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} - \frac{4}{5}\right) + \left(\frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} - \frac{4}{10}\right) + \cdots$$

When k gets very large there will be loss of accuracy due to cancellation. The sum of the four positive fractions in the k^{th} term will be very close to the size of the one negative fraction.

Getting a common denominator and simplifying the numerator eliminates the cancellation.

$$\sum_{k=0}^{\infty} \frac{1250k^3 + 2000k^2 + 1000k + 154}{(5k+1)(5k+2)(5k+3)(5k+4)(5k+5)}$$

Now calculating the k^{th} term will be stable since everything in this formula is positive. Define f7 to compute the k^{th} term.

f7: 7, sto, 1, func, 3, $\mathrm{y^x}$, 1250, *, 7, rcl, $\mathrm{x^2}$, 2000, *, +, 7, rcl, 1000, *, +, 154, +, 7, rcl, 5, *, 7, sto, 1, +, /, 7, rcl, 2, +, /, 7, rcl, 3, +, /, 7, rcl, 4, +, /, 7, rcl, 5, +, /

Sum the series:

40, fix, 7, func, 0, enter, e9999, enter, 1, enter, 7, sum

1.6094379124341003746007593332261876395256

The true sum for this example series is ln(5), and subtracting that from the sum key's result gives zero.

Example 8. A series with more complicated terms.

$$\sum_{k=0}^{\infty} \binom{2k}{k} \frac{1}{4^k(2k+1)^2}$$

Here we can't just use the sum key from 0 to e9999. The problem is that the binomial coefficients get big quickly, and will overflow before k gets to 10^{10}. The 4^k in the denominator is growing at about the same rate, so the entire k^{th} term is going to zero at the rate $1/k^{2.5}$, but to get the sum key to approximate the infinite series we can't compute $\binom{2k}{k}$ and 4^k separately for large k.

The easiest approach is to try extrapolation. Define f8 to compute the k^{th} term of the series. Use the extr key from screen 7 to extrapolate using partial sums from $k = 0$ to 100. That will avoid overflow problems by using only small values of k.

f8: 1, sto, 1, func, 2, *, 1, +, x^2, 1, rcl, 4, x↔y, y^x, *, 1/x, 3, func, 1, rcl, 2, *, 1, rcl, cmb, *

Extrapolate the sum.

40, fix, 7, func, 1, enter, 0, enter, 100, enter, 8, extr

1.0887930451518010652503444491188069736693

For this example problem the exact value of the sum is $\pi \ln(\sqrt{2})$, and subtracting this from the value above gives about −3e−45, so extrapolation using only the first 101 terms gave over 40 digits correct. See the "Extrapolation" page for more on using the extr key.

Extrapolation doesn't always work this well. It takes more effort from us, but another possibility is to get an asymptotic approximation for $\binom{2k}{k} / 4^k$. It will be accurate for large values of k and allow the sum key to compute the k^{th} term without overflow.

In order to know how many terms of this asymptotic approximation are required, we will need to know more about how the sum key approximates series like this where all the terms have the same sign.

For these series the sum key explicitly adds the first 10,000 terms, then uses an integral and several derivatives to approximate the rest of the infinite series. See "Euler-Maclaurin formula" in Wikipedia. We will use the original formula in f8 for the first 10,000 terms.

The integral will use the asymptotic approximation to integrate out to infinity. We want the sum key to use the original formula for k up to term 9999, and switch to the asymptotic approximation after that. This series starts at zero and steps by 1, so term 9999 is $k = 9998$.

The asymptotic approximation is numerically stable and avoids intermediate overflow for large k. It is also much faster than the original formula, since the derivative terms in the Euler-Maclaurin formula need higher precision. In the original formula, the cmb function for binomial coefficients uses many Bernoulli numbers at high precision, and they are slow to compute and waste a lot of memory.

This asymptotic approximation is messy. To get 50 digit accuracy at $k = 10^4$, we will need the terms out to $1/k^{12}$. Using Wolfram Alpha on the internet, we can ask for a series expanded about infinity to be carried out to 13 terms:

Series[Binomial[2k, k]/4ˆk, {k, Infinity, 13}]

After simplifying Alpha's output and putting the $(2k+1)^2$ back in the denominator, we get this approximation for large k, where $t = 1/(8k)$:

$$\binom{2k}{k}\frac{1}{4^k(2k+1)^2} \approx \frac{1}{\sqrt{\pi k}(2k+1)^2}\Bigg(1 - t + \frac{1}{2}\,t^2 + \frac{5}{2}\,t^3 - \frac{21}{8}\,t^4 - \frac{399}{8}\,t^5 + \frac{869}{16}\,t^6 + \frac{39325}{16}\,t^7 - \frac{334477}{128}\,t^8 - \frac{28717403}{128}\,t^9 + \frac{59697183}{256}\,t^{10} + \frac{8400372435}{256}\,t^{11} - \frac{34429291905}{1024}\,t^{12}\Bigg)$$

We will use the original formula from $k = 0$ up to 9998, since the asymptotic formula is not accurate enough for small k. Then use the asymptotic formula for large k.

Define f7(k) so that it will use the select function to call f8 and compute the original formula if $0 \le k \le 9998$, otherwise call f9 and use the asymptotic formula.

f7: 1, sto, 7, func, 0, enter, 9998, enter, 8, sel

f9: 1, func, 1, rcl, 8, $*$, 1/x, enter, enter, enter, enter, -34429291905, $*$, 1024, /, 8400372435, enter, 256, /, +, $*$, 59697183, enter, 256, /, +, $*$, 28717403, enter, 128, /, $-$, $*$, 334477, enter, 128, /, $-$, $*$, 39325, enter, 16, /, +, $*$, 869, enter, 16, /, +, $*$, 399, enter, 8, /, $-$, $*$, 21, enter, 8, /, $-$, $*$, 2.5, +, $*$, 0.5, +, $*$, 1, $-$, $*$, 1, +, 1, rcl, 2, $*$, 1, +, x^2, /, 1, rcl, π, $*$, $\sqrt{x}$, /

f9 uses Horner's rule to evaluate the polynomial in t. See the "Arctan sum" page for more on Horner's rule. Now use the sum key to evaluate the series.

7, func, 0, enter, e9999, enter, 1, enter, 7, sum

1.0887930451518010652503444491188069736693

Subtracting the exact value, $\pi \ln(\sqrt{2})$, gives $-1.0\text{e}{-56}$, so we have the sum of the series to over 50 digits.