Double Sum

Approximate this double sum: $S = \sum_{k=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{j^3 + k^2}$

For double sums with a finite and moderately-sized number of terms, we can define one function that uses the sum key to do each j sum, then use a second sum function to add that first function with respect to k.

This double sum has slow convergence in both j and k, so that won't give much accuracy in a reasonable amount of time in this case.

Extrapolation is probably the best option, but because the rates of convergence are different in the j and k directions, extrapolation will have trouble. Like the sum function, doing separate extrapolations for j and k will probably take too long.

So we will combine groups of terms from the double sum to get one sum to extrapolate. It might help if we make the terms approach zero at the same rate for both $j \to \infty$ and $k \to \infty$. Even then, grouping the terms will leave us a complicated function that isn't easy to extrapolate.

$$S = \sum_{k=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{j^3 + k^2} = \sum_{k=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{k^3 + j^2}$$

$$2S = \sum_{k=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{j^3 + k^2} + \sum_{k=1}^{\infty} \sum_{j=1}^{\infty} \frac{1}{k^3 + j^2} = \sum_{k=1}^{\infty} \sum_{j=1}^{\infty} \left(\frac{1}{j^3 + k^2} + \frac{1}{k^3 + j^2} \right)$$

$$S = \frac{1}{2} \sum_{k=1}^{\infty} \sum_{j=1}^{\infty} \left(\frac{1}{j^3 + k^2} + \frac{1}{k^3 + j^2} \right)$$

f1 will compute these terms. $\mathrm{f1}(j,k) = \frac{1}{2} \left(\frac{1}{j^3 + k^2} + \frac{1}{k^3 + j^2} \right)$

Visualize these $\mathrm{f1}(j,k)$ terms in the double sum as being laid out in a 2-dimensional grid. To get a single sequence of partial sums for extrapolation, group all the terms in row i, columns 1 through i, and all the terms in column i, rows 1 through i.

$S_1 = \mathrm{f1}(1,1)$
$S_2 = \mathrm{f1}(2,1) + \mathrm{f1}(2,2) + \mathrm{f1}(1,2)$
$S_3 = \mathrm{f1}(3,1) + \mathrm{f1}(3,2) + \mathrm{f1}(3,3) + \mathrm{f1}(1,3) + \mathrm{f1}(2,3)$

Now that f1 is symmetric in i and j, we can skip almost half the function calls because $\mathrm{f1}(2,1) = \mathrm{f1}(1,2)$, $\mathrm{f1}(3,1) = \mathrm{f1}(1,3)$, $\mathrm{f1}(3,2) = \mathrm{f1}(2,3)$, etc. This gives

$S_1 = \mathrm{f1}(1,1)$
$S_2 = 2\,\mathrm{f1}(2,1) + \mathrm{f1}(2,2)$
$S_3 = 2\big(\mathrm{f1}(3,1) + \mathrm{f1}(3,2)\big) + \mathrm{f1}(3,3)$

The sum of S_i from $i = 1$ to n will add all the terms in the double sum that lie in an $n \times n$ square, with each S_i adding another row and column to the square.

Define f2(i) to compute S_i. f2(i) = 2(f1(i, 1)+ f1(i, 2) + $\cdots$ + f1(i, i − 1))+ f1(i, i).

f2(i) saves i in register 2, then sums f1 from 1 to $i - 1$, doubles it, then adds f1(i, i).

f1(j, k) uses j from register 2 and k from the input argument.

f1: 1, func, 1, sto, 3, y^x, 2, rcl, x^2, +, 1/x, 1, rcl, x^2, 2, rcl, 3, y^x, +, 1/x, +, 2, /

f2: 7, func, 2, sto, 1, x↔y, 1, −, 1, enter, 1, sum, 2, ∗, 3, sto, 2, rcl, 1, f_n, 3, rcl, +

Now do some extrapolation. The estimated errors will probably be less reliable than usual, so just increase the number of terms in the S_i sum and see if the extrapolated values seem to be converging. It will be slow, so we might need to increase the maximum time.

30, fix, 600, time, 7, func, 1, enter, 1, enter, 100, enter, 2, extr	3.508358892928161373800374222926
1, enter, 1, enter, 200, enter, 2, extr	3.508358892276014654390475244314
1, enter, 1, enter, 400, enter, 2, extr	3.508358900770560797075448025053
1, enter, 1, enter, 800, enter, 2, extr	3.508358892276804719543465180895
1, enter, 1, enter, 1200, enter, 2, extr	3.508358892276804719529586491454

These are gradually getting more digits of agreement, except for the $i = 400$ case, which looks worse than the ones above it. The last two agree to about 20 digits, and the extr key's estimated error for the last case was 1.9e−22, so we might hope to have about 20 digits correct for the double sum.

Checking with the more accurate value found below shows the last value above is actually correct to about 25 digits.

The terms of most double sums will be more complicated than this example, forcing us to use summation plus extrapolation as we did above. More complicated terms may also produce an extrapolated result that is not as accurate as this one.

But sometimes, as in this case, the terms have a simple enough form that a computer algebra system can give a closed form for the double sum itself, or one of the single sums in terms of j or k.

If we change the order of summation to put the k sum on the inside,

$$S = \sum_{j=1}^{\infty} \left(\sum_{k=1}^{\infty} \frac{1}{j^3 + k^2} \right)$$

Wolfram Alpha, available on the internet, and Mathematica both give a closed form for the inner sum. Either will accept this form of input for the k series: Sum[1/(j^3 + k^2), {k, 1, Infinity}]

They return a formula for the inner sum:

$$S(j) = \sum_{k=1}^{\infty} \frac{1}{j^3 + k^2} = \frac{\pi\, j^{3/2}\, \coth(\pi\, j^{3/2}) - 1}{2\, j^3}$$

Here $\coth(x)$ is the hyperbolic cotangent function, which we can get using hyperbolic tangent on screen 1: $\coth(x) = 1/\tanh(x)$.

This reduces the problem from a double sum to a single sum in terms of j, so that should make the calculation much simpler.

There is also a closed form for the sum over j, giving a single sum in terms of k, but each of these k terms involves finding the three roots of a cubic polynomial, two of which are complex numbers, and evaluating the polygamma function at those roots. That is probably too slow, so we will use

$$S = \sum_{j=1}^{\infty} \frac{\pi\, j^{3/2}\, \coth(\pi\, j^{3/2}) - 1}{2\, j^3}$$

The terms in this series are positive and decreasing, so we can use the sum function to approximate it. See the "Infinite sums" example page for more discussion.

Define f3(j) to be the j^{th} term, then do the sum and check the sum function's estimated relative error.

f3: 3, sto, 1, func, 1.5, y^x, π, $*$, tanh, 1/x, 3, rcl, 1.5, y^x, $*$, π, $*$, 1, $-$, 3, rcl, 3, y^x, 2, $*$, /

50, fix, 7, func, 1, enter, e9999, enter, 1, enter, 3, sum

3.50835889227680471952958628932015175450655672413588

10, sci, x↔y 2.406344561e−55

Being able to reduce the problem from a double sum to an equivalent single sum allowed us to get 50 digits for the value.