[6.59] Accelerate series convergence

Most mathematical functions can be expressed as sums of infinite series, but often those series converge far too slowly to be of computational use. It turns out that there are a variety of methods that can be used to accelerate the convergence to the limit. Different methods are effective for different series, and an excellent overview is presented by Gourdon and Sebah in reference [1] below. As that paper is quite thorough, I will rely on it for background and just present a TI Basic program which implements one algorithm for accelerating convergence: Aitken's δ^2 -process.

Aitken's method constructs a series of partial sums, which may converge faster then the original terms. It is basically an extrapolation process which assumes geometric convergence, so, the closer the convergence is to geometric, the better Aitken's method works. If s_0, s_1 and s_2 are three successive partial sums (not terms!) of a series, then Aitken's acceleration of the sum is

$$s' = s_0 - \frac{(s_0 - s_1)^2}{s_0 - 2 \cdot s_1 + s_2}$$

Some references give a different but algebraically equivalent form of the this equation, but this one is less susceptible to round-off errors, according to reference [3] below. You can actually repeat this process on the series of s' terms, but that usually doesn't increase the accuracy much.

```
aitkend2(l)
Func
©(terms list); returns {sn,delta}
©Accelerate series w/Aitken's δ^2 method
©11julØ2/dburkett@infinet.com

local i,n,s,sØ,s1,s2,s3,sa,sb

dim(l)→n

seq(sum(left(l,i)),i,1,n)→s          © Series of partial sums

s[n]→sØ                              © Save the individual sums we need
s[n-1]→s1
s[n-2]→s2
s[n-3]→s3

sØ-(sØ-s1)^2/(sØ-2*s1+s2)→sb         © Find the last sum
s1-(s1-s2)^2/(s1-2*s2+s3)→sa         © Find the next-to-last sum

return {sb,sb-sa}                    © Return the last sum, and the difference

EndFunc
```

aitkend2() is called with a list of series terms, and returns the last accelerated sum, and the difference between the last two sums. The difference is useful to get an idea of the convergence; if it is large compared to the sum, then convergence is poor. Try using more terms.

A much-used example for Aitken's method is this series, and I'll use it, too:

$$\frac{\pi}{4} = \sum_{k=0}^{\infty} \frac{(-1)^k}{2 \cdot k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$$

To create an input series of 11 terms for *aitkend2()*, use

```
seq((-1)^k/(2*k+1),k,Ø,1Ø)→s1
```

then

```
aitkend2(s1)
```

returns {0.7854599, 0.0001462}.

which results in an absolute error, compared to pi, of 0.000247. In comparison, the final difference of the original 11 terms is about 0.1, and the absolute error is also about 0.1. With the same 11 terms, Aitken's method improves the final result by nearly three orders of magnitude.

No convergence acceleration works well if the series terms are increasing. In this case, find the point at which the terms begin decreasing, sum the increasing terms, then apply the acceleration to the remaining terms.

References:

[1] Convergence acceleration of series, Xavier Gourdon and Pascal Sebah, 2002
http://numbers.computation.free.fr/Constants/Miscellaneous/seriesacceleration.html
A concise, thorough survey of convergence methods, starting with a good basic introduction. Covers seven different acceleration methods, including the (popular) Euler's method for series with alternating signs, and an interesting method based on the zeta function. Highly recommended.

[2] Numerical Methods for Scientists and Engineers, Richard W. Hamming, Dover, 1973.
A short but typically pithy description of Aitken's method, which Hamming oddly calls Shanks' method.

[3] Numerical Recipes in Fortran, 2nd edition, William H. Press et al, Cambridge University Press, 1992. http://www.nr.com/
Section 5.1 is aptly titled *Series and Their Convergence*, and Press and his cohorts first mention Aitken's method, but then provide code for Euler's method.

(Credit to Ralf Fritzsch)