Subject: Re: TOTAL gives totally different result on identical array
Posted by Liam Gumley on Fri, 08 Jul 2011 14:37:32 GMT
View Forum Message <> Reply to Message

[stuff deleted]

The online documentation for the TOTAL function in IDL warns that this may happen:

---quote begins---
You should be aware that when summing a large number of values, the result from TOTAL can depend heavily upon the order in which the numbers are added. Since the thread pool will add values in a different order, you may obtain a different — but equally correct — result than that obtained using the standard non-threaded implementation. This effect occurs because TOTAL uses floating point arithmetic, and the mantissa of a floating point value has a fixed number of significant digits. The effect is especially obvious when using single precision arithmetic, but can also affect double precision computations. Such differences do not mean that the sums are incorrect. Rather, they mean that they are equal within the ability of the floating point representation used to represent them.
---end quote---

In a previous posting, someone mentioned the Kahan algorithm for computing a numerically stable summation:

http://en.wikipedia.org/wiki/Kahan_summation_algorithm

A translation to IDL looks like this:

```
function kahansum, input
sum = 0.0
c = 0.0
for i = 0L, n_elements(input) - 1 do begin
  y = input[i] - c
  t = sum + y
  c = (t - sum) - y
  sum = t
endfor
return, sum
end
```

The IDL documentation for TOTAL includes the following example for demonstrating the impact of array ordering on summation:

```
IDL> vec = FINDGEN(100000)
IDL> PRINT, TOTAL(vec) - TOTAL(REVERSE(vec))
```

-87552.0

However the Kahan summation, while considerably slower, does the summation with much less error:

IDL> PRINT, kahansum(vec) - kahansum(reverse(vec))
0.00000

Cheers,
Liam.
Practical IDL Programming
http://www.gumley.com/
