Subject: Re: TOTAL gives totally different result on identical array
Posted by Fabzou on Fri, 08 Jul 2011 17:46:43 GMT
View Forum Message <> Reply to Message

Thank you very much for these functions.

Do you suggest to use PAIRWISE_SUM systematically? Is it possible to
make a generic function such as:

```
function robust_total, x

  if (*statement to define*) then return, PAIRWISE_SUM(X) $
    else return, total(X)

end
```

Thank you for your help!

Fabien

On 07/08/2011 07:25 PM, Liam Gumley wrote:
> On Jul 8, 10:44 am, Fï¿½LDY Lajos<fo...@rmki.kfki.hu> wrote:
>> It will be very slow. But it's IDL, vectorize it!
>
> The pairwise summation algorithm is sometimes recommended as a faster
> solution:
>
> http://en.wikipedia.org/wiki/Pairwise_summation
>
> Here is an IDL implementation (with very little testing!)
>

```
> FUNCTION PAIRWISE_SUM, X
> compile_opt idl2
> forward_function pairwise_sum
> np = 100
> nx = n_elements(x)
> if (nx le np) then begin
>    ;- Naieve summation
>    s = total(x, /double)
> endif else begin
>    ;- Divide and conquer: recursively sum two halves of the array
>    m = floor(nx / 2)
>    s = pairwise_sum(x[0:m-1]) + pairwise_sum(x[m:*])
> endelse
> return, s
> END
>
```

```
>  Increasing the value of NP makes the algorithm faster, but potentially
>  decreases accuracy. Here's a challenging test case:
>
>  PRO TEST
>
>  vec = 100 ^ (10.0 * randomu(123456, 10000000))
>  help, vec
>  print, 'MIN = ', min(vec), ' MAX = ',
>  max(vec)
>
>  print
>  print, 'TOTAL result'
>  print, total(vec) - total(reverse(vec))
>
>  print
>  print, 'TOTAL double precision result'
>  print, total(vec, /double) - total(reverse(vec), /double)
>
>  print
>  print, 'Kahan sum result'
>  t1 = systime(1)
>  result = kahansum(vec) - kahansum(reverse(vec))
>  t2 = systime(1)
>  print, result
>  print, 'Elapsed time (seconds) = ', t2 -
>  t1
>
>  print
>  print, 'Pairwise sum result'
>  t1 = systime(1)
>  result = pairwise_sum(vec) - pairwise_sum(reverse(vec))
>  t2 = systime(1)
>  print, result
>  print, 'Elapsed time (seconds) = ', t2 -
>  t1
>
>  END
>
>  Results of the test case:
>
>  IDL>  test
>  % Compiled module: TEST.
>  VEC             FLOAT     = Array[10000000]
>  MIN =       1.00000 MAX =   9.99996e+19
>
>  TOTAL result
>  % Compiled module: REVERSE.
>    -2.30584e+18
```

>
> TOTAL double precision result
>  -2.5769804e+10
>
> Kahan sum result
> % Compiled module: KAHANSUM.
>  0.00000
> Elapsed time (seconds) =  5.7468300
>
> Pairwise sum result
> % Compiled module: PAIRWISE_SUM.
>  0.0000000
> Elapsed time (seconds) =  0.89422798
>
> Cheers,
> Liam.
> Practical IDL Programming (in print for 10 years!)
> http://www.gumley.com/

---