Subject: Re: Avoiding a for cicle
Posted by Craig Markwardt on Tue, 11 Apr 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Benno Puetz <puetz@mpipsykl.mpg.de> writes:
> "J.D. Smith" wrote:
>> Ricardo Fonseca wrote:
>>>
>>> Hi
>>>
>>> I'm looking for a more efficient way of implementing the following (i.e.
>>> avoiding the for cycle) which is a routine for finding local maximuns
>>>
...
>>> for i = 1, nx-1 do $
>>> if ((Data[i] gt Data[i-1]) and (Data[i] gt Data[i+1])) then $
>>> max_pos = [[max_pos],i]
>>>
>>> ; and then throw away the first element...
>>
>> max_pos = where(data gt median(data,3))
>>
>
> While this is rather efficient concerning code length,
>
>
> maxpos = WHERE(TEMPORARY(data[0:nx-3]) LT TEMPORARY(data[1:nx-2]) AND $
>
> TEMPORARY(data[1:nx-2]) GT TEMPORARY(data[2:nx-1])) + 1
>
> should execute faster, especially for longer arrays

And the code-shortened version of this is:

maxpos = where((data LT data(1:*)) AND (data(1:*) GT data(2:*))) + 1

There are two key things to note here. First, TEMPORARY is not needed
when you are indexing an array, since subscripted array expressions
are already considered temporary. Second, IDL automatically truncates
1-D arrays in binary operations. So, the finite difference expression
normally written like this:

diff = data(1:nx-1) - data(0:nx-2)

can be written like this:

diff = data(1:*) - data

The two data arrays are of different length, so IDL takes the shortest
of each. Saves some keystrokes, and the calculation of NX.

Craig

--

---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------