Subject: Re: Looking for more ideas on code ...
Posted by JD Smith on Tue, 01 Oct 2002 15:12:12 GMT
View Forum Message <> Reply to Message

On Mon, 30 Sep 2002 17:55:25 -0700, Craig Markwardt wrote:

```
> jeyadev@wrc.xerox.bounceback.com (Surendar Jeyadev) writes:
>
>> I have a question about how best (style and function, if possible!) to
>> write code for a function that has limits that have to be treated in a
>> special way. Consider the function
>>
>>     f(x) = sin(x)/x
>>
>> as an example. Now, if x is always a scalar, then on just tests to see
>> if it is zero, and then handle that special case using a if .. then ..
>> else construct. But, what if x can also be scalar? I have the following
>> code that works:
>>
>> -----------------------------------------
>>
>> function sinc, y
>>
>>
>> if(n_elements(y) eq 1) then begin              ; y is a scalar
>>    if(y eq 0.0) then profile = 1.0 else begin
>>       profile = sin(y)/y
>>    endelse
>>  endif else begin                              ; y is a vector
>>    zeros = where(y eq 0.0, ind)
>>    if(ind gt 0) then y(zeros) = 1.0e-10      ; set zeroes to a small
>>    quantity profile = sin(y)/y
>> endelse
>>
>> profile = profile*profile/a0
>>
>>
>> return, profile
>>
>> end
>>
>> -----------------------------------------
>>
>> I guess the one can always set
>>
>>    profile(zeros) = 1.0
>>
```

>>  to handle the more general cases. But, the real question is there a
>>  better way than
>>
>>     zeros = where(y eq 0.0, ind)
>>     if(ind gt 0) then y(zeros) = "special values" notzeros = where(y ne
>>     0.0, ind)
>>     if(ind gt 0) then y(notzeros) = "general definition"
>>
>>  I do understand that one should not compare reals, etc., but I will
>>  clean up the numerics later.
>
>  First of all, this is a perfectly good time to compare reals.  The
>  discontinuity only exists at zero, no where else.
>
>  Second of all, you can simplify your logic a little, by pre-filling the
>  array with the "special case:"
>
>    profile = y*0 + 1.   ;; Tricky way to get array filled with zeroes wh

That certainly the canonical "tricky" way to get an array of 1's, and, at
least on my machine, it's actually faster for most array sizes than:

profile=make_array(n_elements(y),/FLOAT,VALUE=1.)

I started to write this to demonstrate how certain tricks like this can be
inefficient, only to find it's actually *more* efficient in most cases.

Hmmph.  Live and learn.

JD