Subject: Re: Looking for more ideas on code ...
Posted by Craig Markwardt on Tue, 01 Oct 2002 00:55:25 GMT
View Forum Message <> Reply to Message

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 = where(y NE 0, ct)
profile(wh) = sin(y(wh))/y(wh)
```

That's not too bad. You can get trickier too, and in fact cheat with this little doozy, which only works for an even function:

```
sz = size(y) & isdouble = sz(sz(0)+1) EQ 5
ymin = (machar(double=isdouble)).xmin
yp = abs(y)+ymin

profile = sin(yp)/yp
```

This will get at maximum only 2 ulps of error, and even then only if your values of Y are near 10^{-308}. It works by adding a small value to ABS(Y), so that the sinc evaluation is never exactly equal to zero. The top two lines are needed to determine correct value to add, float vs. double. This is a bit too sneaky.

Craig

--

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

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

Subject: Re: Looking for more ideas on code ...
Posted by jeyadev on Tue, 01 Oct 2002 20:11:20 GMT
View Forum Message <> Reply to Message

In article <onr8fbugea.fsf@cow.physics.wisc.edu>,
Craig Markwardt <craigmnet@cow.physics.wisc.edu> wrote:
>
> jeyadev@wrc.xerox.bounceback.com (Surendar Jeyadev) writes:
>
>>   ....
>>
>> --
>>
>> 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

My mistake there. I am actually after sinc^2 ..... but it
has not caused any harm!

>> return, profile
>>
>> end
>>
>
> 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 = where(y NE 0, ct)
>   profile(wh) = sin(y(wh))/y(wh)

Nice one that, when the only exceptional value is the same for
all "problem" points.

thanks
--

Surendar Jeyadev   jeyadev@wrc.xerox.bounceback.com

Remove 'bounceback' for email address

Subject: Re: Looking for more ideas on code ...
Posted by Craig Markwardt on Wed, 02 Oct 2002 02:44:28 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@as.arizona.edu> writes:
> 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.

Interesting performance result! I do it because it allows me to

control the type and dimension of the output array pretty simply. The effects of the following statement can be pretty subtle:

```
y = x*0 + 1.
```

This statement guarantees that Y has the same dimensions as X (except for trailing unit dimensions darnit). But the other nice thing this does is guarantee a certain minimum data type for Y.

Because I am adding the floating point value "1.", Y is guaranteed to be at least floating point. *BUT* if X is double precision, then Y will be double precision as well. This is a nice way to keep the internal precision consistent without resorting to the awkward and error-prone "DOUBLE" keywords that pepper the IDL library.

Craig

--

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