Subject: Re: Finding the closest value in an array...
Posted by timrobishaw on Wed, 31 Mar 2004 08:41:59 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@as.arizona.edu> wrote in message
> For monotonic arrays, you know either one or the other of the two
> bracketing values is the closest. VALUE_LOCATE is faster than
> MIN(ABS()) since it relies on the monotonicity to skip rapidly through
> the vector using bisection. This doesn't address your aesthetic
> concerns, but it's much more efficient:
>
> j=value_locate(r,find)
> mn=min(abs(r[j:j+1]-find),pos)
> pos+=j
>
> When compared to:
>
> mn=min(abs(r-find),pos)
>
> the former can be *much* faster, especially for long arrays. While
> the latter is linear in N, the former is logarithmic.

Hi JD. Thanks for the advanced cleverness. That is great! That factor of 130,000 in speed is wicked awesome! So, if I do a few tests and find that the MIN(ABS()) method is faster for the case when FIND only has one element, should I (would you) add an if/then to check for this case and perform the two-line MIN(ABS()) evaluation so that the slower SORT/MIN/ABS/REBIN method is avoided? I haven't really been too aware of efficiency issues, but I'm starting to do LOTS of reduction on BIG data sets, so I'd better start thinking about this stuff! Thanks a bunch -Tim.

```
> ;; Find indices closest to find values in vector, which must be
> ;; monotonically increasing or decreasing, otherwise a sort vector
> ;; should be passed. Find can be a vector itself.
> function closest,vector,find,SORT=s
>  nf=n_elements(find)
>  sort=keyword_set(s) || arg_present(s)
>  if sort && n_elements(s) ne n_elements(vector) then s=sort(vector)
>  j=value_locate(sort?vector[s]:vector,find)
>  b=[[j>0],[(j+1)<(n_elements(vector)-1)]]
>  mn=min(abs((sort?vector[s[b]]:vector[b])- $
>          rebin([find],nf,2)),DIMENSION=2,pos)
>  pos=j>0+pos/nf
>  return,sort?s[pos]:pos
> end
```