

---

Subject: Re: Finding the closest value in an array...  
Posted by [JD Smith](#) on Wed, 31 Mar 2004 18:07:06 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Wed, 31 Mar 2004 00:41:59 -0800, Tim Robishaw wrote:

```
> 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.
```

I suppose that's reasonable, but it will be very machine specific.  
Here's what I'd recommend: if you're always looking for just a few  
values in long unordered vectors, it's probably not worth a fancy  
SORT()/VALUE\_LOCATE()-based solution. You won't beat a linear search.

What does "a few values" mean? Since sorting (the good kind anyway)  
is an operation of order  $N \log(N)$ , linear search is of order  $N$ , and  
bisection search on an ordered list is of order  $\log(N)$ , to sort+bisect  
 $k$  values is of order  $N \log(N) + k \log(N)$ , whereas a straight search on  $k$   
values is of order  $kN$ . So when  $kN/(N+k) > a \log(N)$  you should switch to  
pre-sorting. Here 'a' is a pre-factor which should be of order 1  
(meaning .1-10 or so). If  $N$  is always large compared to  $k$ , this  
simplifies to  $k > a \log(N)$ .

What this argument fails to capture, however, is the tremendous speedup gained by performing your loop over  $k$  values inside of `VALUE_LOCATE`: the  $k$  in linear search (performed in IDL), and  $k$  in bisection search (performed internally in `VALUE_LOCATE`) are not actually equivalent. This is a harder to quantify, but nonetheless real speedup. For  $N \gg k$ , it may just translate into a different pre-factor.

JD

---