Subject: Re: extracting values from an array
Posted by Michael Wallace on Wed, 01 Jun 2005 19:49:07 GMT
View Forum Message <> Reply to Message

> Actually I thought maybe there is a shortcut like:
> B = A gt -1
>
> But if
> A = [2,-1, -1, 3,-1,-1, 4, -1]
> the command
> B = A gt -1
> leads to
> B = [1,0,0,1,0,0,1,0]
> which this doesn't help me as I want B = [2,3,4]

In this case, the "a gt -1" returns a boolean value based on whether the expression is true or not. That's why you see the array of only 1 (true) and 0 (false). Also note that whenever you do array operations, your resultant array will be the same size as the input array. You want your output array to be potentially smaller and therefore you need at least one call to a function such as "where" to handle the manipulation. By the way, the problem you describe is the very reason why where was invented.

If you want something a little cleaner, I'd suggest using the count parameter set by the where command rather than checking if w is -1 or not. I just find this a little easier to read.

```
a = [2,-1, -1, 3,-1,-1, 4, -1]
w = where(a gt -1, count)
if count gt -1 then b = a[w]
```

> The array A is never bigger than 50 elements.
> So I will keep my three lines of code.

All of us like to tinker and find some slick way of doing things, but if the code works as it should and there's no real gain to be achieved from optimizing, don't optimize! And never optimize just for the sake of optimization. Profile your code, find the major bottlenecks and optimize those first. Rinse, wash, repeat.

Of course, if you're just tinkering with something, playing around with it to see how it works, optimize the crap out of it. It's a good learning experience. I still have fond memories of taking one of my co-worker's programs and bringing its running time from about 20 minutes down to under a minute and even under 30 seconds in some cases. And I

could have taken it down further had I written some DLMs. :-)

-Mike