Subject: Re: set all elements in 2d array between some range to 1
Posted by havok2063 on Mon, 08 Jun 2015 22:03:31 GMT
View Forum Message <> Reply to Message

It looks like that's exactly what is happening. Some are matched to the lower boundary, and some are actually matched to the upper boundary. And it seems like a precision issue. My wavelengths are to 4 sigfigs, but my skyline mask values are to 2 sigfigs, e.g. 3827.1299 vs 3827.13, so that kicks them to a different bin.

On Wednesday, June 3, 2015 at 3:31:36 PM UTC-4, Jeremy Bailin wrote:
>> Thanks a lot for your help. This is great. Exactly the solution I'm looking for. Hmm..my original loop is taking me ~30 seconds. Not sure why yours is much faster. Here is my code. Actually it seems it's because the 10.^loglam is being done every loop iteration. Changing this to a stored variable, the loop takes 2.1 seconds.
>
> Ah, that definitely makes sense! Yes, avoid doing redundant calculations in loops. :) This is always Good Advice... it looks like you got 15x faster just by doing that, which is almost as much as the speedup of going to the more sophisticated algorithm!
>
>> I really like your code, since it's faster and no loops. However, when I run it, I'm getting some small differences between the output from the original vs the new method.
>>
>> IDL> help, where(skylinemask)
>> <Expression> LONG = Array[226935]
>>
>> IDL> help, where(skylinemask_v2)
>> <Expression> LONG = Array[226933]
>>
>> IDL> help, where(skylinemask ne skylinemask_v2)
>> <Expression> LONG = Array[14]
>>
>> Printing these 14 elements for both the old and new mask shows that are flipped from each other. Could it be a boundary issue?
>
> Quite possibly. The Value_Locate code will mask when wave lies exactly at a lower boundary (but not at an upper boundary), while your code doesn't mask at either boundary. So I'd suggest checking what the exact wavelengths of those 14 discrepancies are -- I suspect you'll find that they're exactly at the lower boundary of one of the mask regions.
>
> -Jeremy.