Subject: Re: unwrap modulo 2pi
Posted by tam on Wed, 07 Feb 2001 19:18:30 GMT

View Forum Message <> Reply to Message

Perhaps I'm confused but it seems to me you have
a misunderstanding of what the modulo function does.
Since you indicate that omatrix has an 'unwrapping'
function I figure I'm still missions something...

If

    c = a mod b

and I have the values of c and b, then all I know about
a -- absent any other information -- is that

    a element_of   {c + n*b}

where n ranges over the integers. [Some languages
treat modulos of negative numbers differently so
we might be restricted to n >= 0). The modulus function
by definition gives values that repeat with a cycle
length of b, so the easiest solution for an
inverse of it is simply the identity function.
I.e., taking your fourth example.

    8 mod 2*!pi = 1.71681

but

     1.71681 mod 2*!pi = 1.71681

You've lost information using the mod function and
you can get it back...

The one thing I'm guessing is that in omatrix your
'unwrapping' function works on arrays and requires
that the value of the array increase monotonically.
Presumably the delta's between the array values
are not constant -- or the unwrapping is trivial --
so one is still not guaranteed that the unwrapping
proceeds correctly. However in the case where the spacing
between array values is never greater than b you should be able
to write a relatively simple unwrapping function...

Something like [untested and I've no doubt
someone can do it without loops.]:

```
   function unwrap, a, modval

       len  = n_elements(a)
```

```
        if (len lt 2) then begin  ; Need to have multiple elements
           b = a
           return, b
        endif

        w    = where (a[0:len-2] gt a[1:len-1])  ; Find the wrapping points
        if (w[0] eq -1) then begin               ; If none just return a copy
of input
           b = a
           return, b
        endif

        z = a          ; Get a copy of the input
        z[*] = 0       ; Set all values to nil
        z[w] = modval   ; Set deltas at wrapping points.
        for i=1, n_elements(z)-1 do begin  ; Loop to add deltas.
          z[i] = z[i] + z[i-1]
        endfor

        return, a + z   ; Add the deltas back in.
    end
```

Again I note that this inversion of the modulus is possible only given the additional information that the input array was monotonic and had sufficiently small increments. If your additional information is different, then a different inverse may be appropriate (e.g., if you know the input values are integral). In general the function is not invertible.

Regards,
Tom McGlynn

graham_wilson@my-deja.com wrote:

```
>
>  My appologies for not being explicit enough...
>
>  IDL> a=[2,4,6,8,10,12]
>  IDL> a=[2.,4.,6.,8.,10.,12.]
>  IDL> b=2*!PI
>  IDL> c=a mod b
>  IDL> print, c
>        2.00000  4.00000   6.00000   1.71681  3.71681   5.71681
>
>  What I mean by 'unwraping' is: Given I know 'c' and 'b' how do I
>  explicitly find a?
```

>
> GW
>
> In article <3A817F92.1DBCDCCA@noaa.gov>,
> "Pavel A. Romashkin" <pavel.romashkin@noaa.gov> wrote:
>> I am sorry, but I am still a little behind here, please bear with me.
>> Must be the lack of coffe. What is the "unwrapping of the function
>> mod(x,y)" ? I might think of a solution if I knew what I am looking
> at.
>>
>> Cheers,
>> Pavel
>
> Sent via Deja.com
> http://www.deja.com/