Subject: A better way to find a dip (local minimum with certain conditions)
Posted by DavidPS on Wed, 16 Dec 2009 15:47:16 GMT
View Forum Message <> Reply to Message

Hi all!!

First of all, this is a question of programming in an IDL way. I'm not too concerned on the speed (the one I have does the work fast enough for what I need, but speed suggestions are always welcomed), but I'm sure that there is a better (and more elegant) way to do avoid all the IF statements I used.

I want to extract from an 1D array the position where a local minimum bigger than a certain value and which four elements before and after are in decreasing and increasing order respectively. So, in pseudo code the element I'm trying to find should met the next conditions:

array[i] > value
array[i-4] > array[i] and array[i+4] > array[i]
array[i-4] >= .....>=array[i-1]>=array[i]<=array[i+1]<=....<=array[i+4]

What I have is shown below. It's a modified part of a code done by a college which I'm trying to debug and restructure.

```
;====================== CODE========================
FUNCTION finddip,array,minim
  start=4
  finish=n_elements(array)-5
  pos=[0]
  FOR i=start,finish DO BEGIN
    IF array[i] GE minim THEN BEGIN
      IF array[i-4] GT array[i] AND array[i+4] GT array[i] THEN
BEGIN
        IF array[i-4] GE array[i-3] AND $
          array[i-3] GE array[i-2] AND $
          array[i-2] GE array[i-1] AND $
          array[i-1] GE array[i]   AND $
          array[i]   LE array[i+1] AND $
          array[i+1] LE array[i+2] AND $
          array[i+2] LE array[i+3] AND $
          array[i+3] LE array[i+4] THEN BEGIN
          pos=[pos,i]
        ENDIF   ; (DIP)
      ENDIF    ; (GT ARRAY(+-4))
    ENDIF     ; (GE MIN)
  ENDFOR     ;(I)
  RETURN, n_elements(pos) gt 1 ? pos[1:*] : -1
```

END
;================== END CODE======================

All suggestions are welcome!! Thanks in advance!!

David

Subject: Re: A better way to find a dip (local minimum with certain conditions)
Posted by James[2] on Thu, 17 Dec 2009 22:57:37 GMT
View Forum Message <> Reply to Message

what about:

FUNCTION finddip, array, min, range

; find the derivative
deriv = array - SHIFT(array, -1)

; reduce the derivative to negative and positive
pos = deriv gt 0
neg = deriv lt 0

; create convolution kernels for the elements before and after
beforekern = [replicate(1,range),intarr(range-1)]
afterkern = [intarr(range+1),replicate(1,range)]

; find places where the leading derivatives are non-positive
; and the trailing derivatives are non-negative
dip = logical_and(convol(pos, beforekern) eq 0, convol(neg, afterkern)
eq 0)

; find where the value exceeds the threshold
thresh = array gt min

; put it all together
localmin = logical_and(dip, thresh)

return, localmin

end

does what you want, it might be a little heavy on the memory but it
doesn't use any loops.
