Subject: Re: speeding up code for fitting spectra for doppler map
Posted by Jeremy Bailin on Tue, 15 Mar 2016 15:28:52 GMT
View Forum Message <> Reply to Message

On Tuesday, March 8, 2016 at 1:37:49 PM UTC-5, Krishna Mooroogen wrote:
> Hi,
>
> I'm new to using the IDL google groups so please forgive me if my question is not formatted to the norm.
>
> I have written some code to create doppler images from a 4d image array (x,y,w,t) where w is the wavelength. Currently, the code takes nearly 3 hours to make one frame of the doppler image. I have 79 frames!
>
> To do this, I loop over each pixel in the image and fit a function to the spectra found in each pixel to find the centroid, the central peak position is important to make the doppler map.
>
> I use a rudimentary peak finding algorithm at the start to find the peaks as the profile shifts around from pixel to pixel and sometimes flattens out. Then fit a function around the peak.
>
> I have clauses that change the start parameters if the fit is not satisfactory and also to record positions that could not be fitted. I would like to retain these features if possible.
>
> Below is the code in question. In this version, it is set up to do a single frame and so I do not loop in time just in position. If anyone has any ideas how I can speed up the procedure I will be very grateful! I am using IDL 7.
>
> pro vel_map3,data,map,coords
>
> sz=size(data)
>
> x=([-0.29000000,-0.21700000,-0.14500000,-0.073000000,0.00000
00,0.073000000,0.14500000,0.21700000,0.29000000]+8542)/10.0
>
> coords=0
>
> map=fltarr(861, 481, /nozero)
> grad=190000
>
> FOR j=0 , sz(2)-1 DO BEGIN
> FOR e=0 , sz(1)-1 DO BEGIN
>
> d_cut=reform(data[e,j,3:11])
> const=max(d_cut)
> d_cut=d_cut-const
>
> FOR i=2, 6 DO BEGIN
> minval=min(d_cut[i-2:i+2],xi)

```
>  ;Wait till maximum is at centre of search bar
>  IF xi EQ 2 THEN BEGIN
>  in=i
>  BREAK
>  ENDIF
>  in=i
>  ENDFOR
>
>  res1=(d_cut[in]-d_cut[in-2])/(x[in]-x[in-2])
>  res2=(d_cut[in+2]-d_cut[in])/(x[in+2]-x[in])
>
>  IF (res1 LT -1*grad) AND (res2 GT grad) AND (xi EQ 2) THEN BEGIN
>
>  x2=x[in-2:in+2]
>  d2=d_cut[in-2:in+2]
>  der=sqrt(d2)
>
>  cent=total(double(d2)^2*x2)/total(double(d2)^2)
>
>  p=[min(d2,n),cent,0.8,600]
>
>   f=mpfitpeak(x2,d2,res,nterms=4,errors=der,estimates=p,perror
=perr,/quiet,/moffat,/negative,chisq=chi)
>
>  IF chi GT 1e8 THEN BEGIN
>  p[2]=0.6
>   f=mpfitpeak(x2,d2,res,nterms=4,errors=der,estimates=p,perror
=perr,/quiet,/moffat,/negative,chisq=chi)
>  ENDIF
>
>  IF chi GT 1e8 THEN BEGIN
>  p[2]=0.5
>   f=mpfitpeak(x2,d2,res,nterms=4,errors=der,estimates=p,perror
=perr,/quiet,/moffat,/negative,chisq=chi)
>  ENDIF
>
>  map[e,j]=res(1)
>
>  ENDIF ELSE BEGIN
>  coords=[temporary(coords),e,j]
>
>  ENDELSE
>  ENDFOR
>  ENDFOR
>
>  coords=coords[1:*]
>
>  END
```

If you really need to fit each pixel individually using mpfitpeak, it will be difficult to do a full vectorization. However, I suspect that innermost loop at least could be vectorized if I could figure out what exactly it's doing -- could you explain it in words?

-Jeremy.
