Subject: Re: Fast Implementation
Posted by Sven Geier on Tue, 09 Jul 2002 21:58:09 GMT
View Forum Message <> Reply to Message

[Humm - resending. My new newsreader seems to have small troubles...]

Isa Usman wrote:

```
> Hi,
>
> I have the bit of code below which calculates the number of points in all
> four quandrants of a 2d space. Unfortunately my arrays are very large and
> it takes quite a while to run.Is there a way of making the code faster.
>
> Thanks in advance!
>
> Isa
> @@@@@@@@@@@@
> ;Data samples
> for j=0L,n1-1 do begin
>  x0=X(j)
>  y0=Y(j)
>
>  index=where(X gt x0 and Y gt y0,count1)
>  index=where(X lt x0 and Y gt y0,count2)
>  index=where(X lt x0 and Y lt y0,count3)
>  index=where(X gt x0 and Y lt y0,count4)
>
>  na=count1
>  nb=count2
>  nc=count3
>  nd=count4
>
>
>  points(j,0:3)=float([na,nb,nc,nd])/n2
>
> endfor
```

Just to get that straight: You have n1 points, where n1 is large and for
each point you want to know how many of the other points are
above/below/left/right of it, right?

I am tacitly assuming here that you are using reals or doubles, i.e. there
is rarely ever an event with the exact same X[] or Y[] as another (right?
wrong?)

Step one: If you sort your arrays according to one of the coordinates, the answer for that coordinate will just be the index:

```
 Xs = sort(X)
 Xn = x[Xs]
 Yn = Y[Ys]
```

Then X[i] has (i-1) other points with X < X[i]

Step two: for each point you only need to know how many other points are greater in Y, since you know the total number of points (and whatever isn't greater would have to be less or equal). I.e. if

```
 na = where(Yn[0:i-1] lt Yn[i]) ; don't need to check the [i+1:*] elements
                              ; because the sorting put those in the
                              ; other X half-plane
```

Then

```
 nb = i - na
```

IFF you can tolerate a "le" for one half of your quadrants.