Subject: Re: what is the dist function's mean?
Posted by Benjamin Hornberger on Wed, 18 May 2005 18:57:45 GMT
View Forum Message <> Reply to Message

Benjamin Hornberger wrote:
>

> if you do a 2-dimensional
> F.T. (say, of an N x N array), DIST() will give you the frequency
> indices. Then you can create an array with spatial frequencies by
>
> freq = dist(N, N) / (N * delta)
>
> where delta is your real space sampling interval.
>

As a side note, I realized now that the IDL-provided DIST function has a limitation in that it can't handle cases which are not symmetric in X and Y. Even though you can specify N and M separately, if you want to create an array of frequencies by scaling DIST's result by a factor, you can't do it unless the real-space field of view is a square. In other words, N * dx = M * dy is required (dx, dy are the real-space sampling intervals).

I wrote a replacement which can do this. It can also return the frequencies (rather than frequency indices) directly if you pass sampling intervals or the Nyquist frequency in X and Y. In case somebody's interested ...

Benjamin

```
;+
; NAME:
;
;    BH_DIST
;
;
; PURPOSE:
;
;    This function is a more versatile replacement for the
;    IDL-provided DIST() function. If a real-space sampling interval
;    or a maximum (Nyquist) frequency is given, it can calculate the
;    frequency array directly. The number of pixels and the sampling
;    intervals can be different in the x and y directions.
;
;    Note: If you only specify nx, and possibly ny, this function
;    does exactly the same as the IDL-provided DIST() function.
;
```

```
;
; AUTHOR:
;
;    Benjamin Hornberger
;    benjamin.hornberger@stonybrook.edu
;
;
; CATEGORY:
;
;    General programming, frequency analysis
;
;
; CALLING SEQUENCE:
;
;    Result = BH_DIST(nx, ny, dx, dy)
;
;
; RETURN VALUE:
;
;    Returns a rectangular array in which the value of each element is
;    equal to its frequency index. If dx, and optionally dy, are
;    passed, the array will contain frequencies rather than frequency
;    indices.
;
;
; INPUT PARAMETERS:
;
;    nx: Number of pixels in the X direction.
;
;
; OPTIONAL INPUT PARAMETERS:
;
;    ny: Number of pixels in the Y direction. If not passed, it will
;       be set equal to nx.
;
;    dx: Real-space sampling interval in the X direction. If this
;       parameter is passed, the function will return an array of
;       frequencies rather than frequency indices. If additionally the
;       keyword FMAX is set, dx is interpreted as maximum frequency in
;       the X direction (Nyquist frequency).
;
;    dy: Real-space sampling interval in the Y direction. If dy is not
;       passed, but dx is, it is assumed to be equal to dx. If
;       additionally the keyword FMAX is set, dy is interpreted as
;       maximum frequency in the Y direction (Nyquist frequency).
;
;
; INPUT KEYWORDS:
```

```
;
;    CENTER: If this keyword is set, the result will be shifted so
;       that the zero frequency is at nx/2, ny/2 (which means right in
;       the center for odd numbers of pixels).
;
;    FMAX: If this keyword is set, dx and dy are interpreted as
;       maximum (Nyquist) frequencies in the X or Y direction, rather
;       than real space sampling intervals. This keyword has no effect
;       if neither dx nor dy are given.
;
;
; OUTPUTS KEYWORDS:
;
;    FX, FY: Set these keywords to named variables which will contain
;       one-dimensional arrays of frequencies in the X and Y
;       directions.
;
;
; EXAMPLE:
;
;    To calculate a 100 x 120 array of spatial frequencies, with
;    real-space sampling intervals of 1.2 and 1.4 (arbitrary units --
;    the units for the frequencies will just be the inverse):
;
;    Freq = BH_DIST(100, 120, 1.2, 1.4)
;
;
; MODIFICATION HISTORY:
;    Written: BH 2005-05-16
;-

FUNCTION bh_dist, nx, ny, dx, dy, $
                  fx=fx, fy=fy, $
                  fmax=fmax, $
                  center=center

  compile_opt idl2
  on_error, 2

  IF n_elements(nx) EQ 0 THEN message, 'must specify at least nx'
  IF n_elements(ny) EQ 0 THEN ny = nx

  nx1 = long(nx)
  ny1 = long(ny)

  ;; shift parameters
  sx = keyword_set(center) ? 0 : -nx1/2
  sy = keyword_set(center) ? 0 : -ny1/2
```

```
  ;; 1d frequency arrays
  fx = shift(findgen(nx1)-nx1/2, sx)
  fy = shift(findgen(ny1)-ny1/2, sy)

  ;; If sampling intervals are passed, we calculate frequencies rather
  ;; than frequency indices. If only dx has been passed, dy is assumed
  ;; to be the same. If the keyword FMAX is set, we take dx and dy as
  ;; maximum (Nyquist) frequencies.
  IF n_elements(dx) GT 0 THEN BEGIN
     IF n_elements(dy) EQ 0 THEN dy = dx
     fx *= (keyword_set(fmax) ? (1.*dx/(nx/2)) : (1./(nx1*dx)))
     fy *= (keyword_set(fmax) ? (1.*dy/(ny/2)) : (1./(ny1*dy)))
  ENDIF

  ;; 2d frequency arrays
  fx2d = rebin(fx, nx1, ny1)
  fy2d = rebin(reform(fy, 1, ny1), nx1, ny1)

  return, sqrt(fx2d^2.+fy2d^2.)

END
```