Otras funciones útiles

wavread

(Original de octave) Lee archivos wav.

```
[Y, FS, BITS] = wavread (FILENAME)
      Y     = muestras de audio (cada canal en una columna).
      FS    = frecuencia de muestreo (Hz).
      BITS  = bits por muestra.
```

wavwrite

(Original de octave) Escribe archivos wav.

```
wavwrite (Y, FS, BITS, FILENAME)
         Write Y to the canonical RIFF/WAVE sound file FILENAME with sample
         rate FS and bits per sample BITS.  The default sample rate is 8000 Hz
         with 16-bits per sample.  Each column of the data represents a
         separate channel.
```

fftfilt

(Original de octave).

```
fftfilt (b, x, n)
         filters x with the FIR filter b using the FFT.
```

filter

(Original de octave).

```
y = filter (b, a, x)
         In terms of the z-transform, y is the result of passing the discrete-
         time signal x through a system characterized by the following rational
         system function:
                      M
                     SUM d(k+1) z^(-k)
                     k=0
           H(z) = ----------------------
                        N
                  1 + SUM c(k+1) z^(-k)
                      k=1

         where c = a/a(1) and d = b/a(1).
```

freqz

(Original de octave).

```
[h, w] = freqz (b, a, w, Fs)
         Return the complex frequency response h of the rational IIR filter
         whose numerator and denominator coefficients are b and a,
         respectively. Evaluate the response at the specific frequencies in the
         vector w. Return frequencies in Hz instead of radians assuming a
         sampling rate Fs.
```

impz

(Original de octave).

```
usage: [x, t] = impz(b [, a, n, fs])

          Generate impulse-respo,nse characteristics of the filter. The filter
          coefficients correspond to the the z-plane rational function with
          numerator b and denominator a.  If a is not specified, it defaults to
          1. If n is not specified, or specified as [], it will be chosen such
          that the signal has a chance to die down to -120dB, or to not explode
          beyond 120dB, or to show five periods if there is no significant
          damping. If no return arguments are requested, plot the results.
```

```
usage: zplane(b [, a]) or zplane(z [, p])

 Plot the poles and zeros.  If the arguments are row vectors then they
 represent filter coefficients (numerator polynomial b and denominator
 polynomial a), but if they are column vectors or matrices then they
 represent poles and zeros.

 This is a horrid interface, but I didn't choose it; better would be
 to accept b,a or z,p,g like other functions.  The saving grace is
 that poly(x) always returns a row vector and roots(x) always returns
 a column vector, so it is usually right.  You must only be careful
 when you are creating filters by hand.

 Note that due to the nature of the roots() function, poles and zeros
 may be displayed as occurring around a circle rather than at a single
 point.

 The transfer function is

        B(z)   b0 + b1 z^(-1) + b2 z^(-2) + ... + bM z^(-M)
 H(z) = ---- = ------------------------------------------
        A(z)   a0 + a1 z^(-1) + a2 z^(-2) + ... + aN z^(-N)

                 b0          (z - z1) (z - z2) ... (z - zM)
               = -- z^(-M+N) ------------------------------
                 a0          (z - p1) (z - p2) ... (z - pN)

 The denominator a defaults to 1, and the poles p defaults to [].
```

Function File: [*sos*, *g*] = **zp2sos** (*z*, *p*)

Convert filter poles and zeros to second-order sections.

INPUTS:

- *z* = column-vector containing the filter zeros
- *p* = column-vector containing the filter poles
- *g* = overall filter gain factor

RETURNED:

- *sos* = matrix of series second-order sections, one per row:
  sos = [*B1*.' *A1*.'; ...; *BN*.' *AN*.'], where
  B1`.'==[b0 b1 b2]` and *A1*`.'==[1 a1 a2]` for section 1, etc.
  b0 must be nonzero for each section.
  See `filter()` for documentation of the second-order direct-form filter coefficients *B*i and %*A*i, i=1:N.
- *Bscale* is an overall gain factor that effectively scales any one of the *B*i vectors.

EXAMPLE:

```
  [z,p,g] = tf2zp([1 0 0 0 0 1],[1 0 0 0 0 .9]);
  [sos,g] = zp2sos(z,p,g)

sos =
   1.0000    0.6180    1.0000    1.0000    0.6051    0.9587
   1.0000   -1.6180    1.0000    1.0000   -1.5843    0.9587
   1.0000    1.0000         0    1.0000    0.9791         0

g =
    1
```

rline = **dbstop** (*func*, *line*, ...)

Set a breakpoint in a function

`func`

String representing the function name. When already in debug mode this should be left out and only the line should be given.

`line`

Line number you would like the breakpoint to be set on. Multiple lines might be given as separate arguments or as a vector.

The rline returned is the real line that the breakpoint was set at.

dbcont

In debugging mode, quit debugging mode and continue execution.

dbquit

In debugging mode, quit debugging mode and return to the top level.