Subject: Signal processing in IDL
Posted by roy.hansen on Wed, 04 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi,

Are there any libraries / web-pages related to signal processing in IDL?
I am specifically looking for spectral estimation routines
and time-frequency / time-scale analysis routines.

PS! I know the trivial answer: Use matlab....

--RoyH

Subject: Re: Signal processing in IDL
Posted by R.G. Stockwell on Wed, 04 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <oTSp3.1211$Ffg.181237760@news.telia.no>,
  roy.hansen@triad.no (Roy E. Hansen) wrote:
> Hi,
Hi!

> I am specifically looking for spectral estimation routines
> and time-frequency / time-scale analysis routines.
> --RoyH

In article <oTSp3.1211$Ffg.181237760@news.telia.no>,
  roy.hansen@triad.no (Roy E. Hansen) wrote:
> Hi,
> (snip)
> I am specifically looking for spectral estimation routines
> and time-frequency / time-scale analysis routines.

Below is code that calculates the S-Transform.
(NOTE: I think I'll have text-wrap problems posting
from Dejanews. If so, email me and I'll send the code
through email)

input: a timeseries (1-D array)
output: a 2-D complex valued matrix
output with keyword /abs or /pow, a 2-D float matrix
output with keyword /samplingrate, a structure with
 three arrays,(1)1-D time array (2) 1-D frequency array
 (3) the ST 2-D matrix.

USAGE:
IDL> r = s_trans(/example) ; plots an example timeseries and ST

IDL> r = s_trans(/help) ; prints information

IDL> contour,s_trans(cos(findgen(100)),/abs)

email me for futher info.

reference:
@article{R_G_Stockwell_1996_44_998_1001,
Author = {R.G. Stockwell and L. Mansinha and R.P. Lowe},
Title = {Localization of the Complex Spectrum: The S Transform},
Year = {1996},
Journal = {IEEE Transactions on Signal Processing},
Volume = {44},
Number = {4},
Pages = {998-1001}
}

--

R.G. Stockwell
Colorado Research Associates
3380 Mitchell Lane
Boulder Colorado
(mysurname)(at)co-ra(dot)com
^^

```
; The S transform function
; written by bob stockwell, 1993
;
; Returns a matrix if succesful
; a structure  if a sampling rate is given
;    The structure is {st:, Freq:, Time:}
;  where st is the S Transform, freq is a vector containing the
;    frequencies
;
;    Optional Paramters
;   WIDTH   size of time resolution
;   if not set, width = 1
;
;
;    Keywords
;   \HELP    explains all the keywords and
```

parameters
;   \VERBOSE   flags errors and size
; \SAMPLINGRATE   if set returns array of frequency
; \MAXFREQ
; \MINFREQ
; \FREQSAMPLINGRATE
; \PIECEWISENUMBER   divides the time series, and passes back
array
; \POWER   returns the power spectrum
; \ABS   returns the absolute value
spectrum
; \REMOVEEDGE   removes the edge with a 5% taper, and
takes
;   out
least-sqares fit parabola

; added a "mask out edges" keyword,
; which STs a line (ie st of edges) and thresholds the returned st
matrix.
; The value of masked edges is the percent at which to make the
threshold
; default = 5 %.

; added an EXAMPLE keyword, will display a time series and the
; amplitude of the ST to the current graphics device
; WARNING, will call !P.multi=0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; modified hilbert transform
;;;;;;;;;;;;;;;;;;;;;
;+
; NAME:
;   HILBERT
;
; PURPOSE:
;   Return a series that has all periodic terms shifted by 90
degrees.
;

;^^^ ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
; REVISION HISTORY:
;   JUNE, 1985,   Written, Leonard Kramer, IPST (U. of Maryland)
on site
;   contractor to NASA(Goddard Sp. Flgt. Cntr.)
;-
;== ============
=========
; MODIFIED: APRIL 20 1994 by RGS
;   - returns real function only

```
;                 - added keyword    /ANALYTIC
;                   to return the analytic signal



FUNCTION HILBERT,X,D, ANALYTIC = a   ; performs the Hilbert transform of
some data.
       ON_ERROR,2          ; Return to caller if an error occurs
       Y=FFT(X,-1)         ; go to freq. domain.
       N=N_ELEMENTS(Y)

       I=COMPLEX(0.0,-1.0)

       IF N_PARAMS(X) EQ 2 THEN I=I*D
       N2=ceil(N/2.)-1            ; effect of odd and even # of
elements
                          ; considered here.
       y(0) = complex(0,0)  ; zero the DC value (required for hilbert
trans.)
       Y(1)=Y(1:N2)*I       ; multiplying by I rotates counter c.w. 90
deg.
    if (n mod 2) eq 0  then Y(N2+1) = complex(0,0)     ; don't
need this
       N2=N-N2
  Y(N2)=Y(N2:N-1)/I
       Y=float(FFT(Y,1))  ; go back to time domain
       if keyword_set(a) then y = complex(x,y)
 RETURN,y

END




 ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^
; Gaussian Window Function
; Accepts variables length and width
; Returns the spectrum of the Gaussian window
; note this is length/2piw 11ll11ll
FUNCTION gaussian_window, l, w
 if w ne 0.0 then sigma = l/(2*!PI*w) else MESSAGE,'width is zero!'
 g = fltarr(l)
 iarr = findgen(l)
 ex = (iarr - l/2)^2/(2*sigma^2)
 wl = where(ex lt 25)  ; probably not needed, I ran into a problem about
6 years ago, and I do not remeber what it was!!
 g(wl) = exp(-ex(wl))
 g = shift(g,-l/2)
```

```
 return, complex(g,0)
end


 ;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^^
; the S Transform function
FUNCTION s_trans, ts, factor, HELP =
HELP,VERBOSE=VERBOSE,SAMPLINGRATE=SAMPLINGRATE $
    ,MAXFREQ=MAXFREQ,MINFREQ=MINFREQ  $
    ,FREQSAMPLINGRATE=FREQSAMPLINGRATE     $
    ,POWER=POWER,ABS=ABS,REMOVEEDGE=REMOVEEDGE   $
    , maskedges=maskedges,EXAMPLE=EXAMPLE


if keyword_set(EXAMPLE) then begin  ; show example of a chirp function
 ex_len = 512
 ex_time = findgen(ex_len)
 ex_freq = 5;ex_len/16
 ex_ts =  cos(2*!Pi*ex_freq*ex_time/ex_len)
 ex_ts =  cos(2*!Pi*(ex_len/5+2*ex_ts)*ex_time/ex_len)
 ; crossed chirp example commented out
  ;ex_ts =
cos(2*!Pi*ex_freq*ex_time*(1+2*ex_time/ex_len)/ex_len)
  ;ex_ts = ex_ts + reverse(ex_ts)
 !P.multi=[0,1,2]
 plot,ex_ts,xtitle='Time (units)',title='Time Series [h(t) =
cos(cos(wt))]'
 s = s_trans(ex_ts,/samp, /abs)  ; returns structure, amplitudes
only returned
 contour,s.st,ex_time,s.freq,nlevels=14,/fill,xtitle='Time
(units)', $
  ytitle='Frequency (1/unit)',title='Amplitude of
S-Transform'
 return,0
 !P.multi=0
endif

if keyword_set(HELP) then begin
 Print,"S_TRANS()  HELP COMMAND"
 Print,"S_trans() returns a matrix if succesful or a structure if
required"
 Print,"S_trans() returns  - 1 or an error message if it fails"
 Print,"USAGE::    localspectra = s_trans(timeseries)
 Print," "
 Print,"Optional Parameters"
 Print,"WIDTH  -size of time resolution"
 Print,"       -if not set, default WIDTH = 1"
 Print," "
```

```
 Print,"Keywords:
 Print,"\HELP          -explains all the keywords and parameters
 Print,"\VERBOSE       -flags errors and size, tells time left
etc.
 Print,"\SAMPLINGRATE -if set returns array of frequency
 Print,"\MAXFREQ
 Print,"\MINFREQ
 Print,"\FREQSAMPLINGRATE
 Print,"\POWER           -returns the power spectrum
 Print,"\ABS             -returns the absolute value spectrum
 Print,"\REMOVEEDGE      -removes the edge with a 5% taper, and
takes
 Print,"                 -out least-squares fit parabola
 return, -1
endif

;print,'paramters',N_params()
;Check number of arguments.
CASE N_PARAMS() OF
 1: begin
  if n_elements(ts) eq 0 then MESSAGE, 'Invalid timeseries
(check your spelling).'
  factor = 1
 end
 2: if n_elements(factor) ne 1 then begin ;Two-argument case.
   factor = 1                    ;Make sure factor
is a number.
    if keyword_set(VERBOSE)then print,'Error in second parameter.
Using default values.'
    endif
ELSE: MESSAGE, 'Wrong number of arguments'
ENDCASE


if n_elements(ts) eq 0 then MESSAGE, 'Invalid timeseries (check your
spelling).'
time_series = ts ; don't change the original dataset passed to this
function

; check to see if it is a vector, not a 1 x N matrix
sz = size(time_series)
if sz(0) ne 1 then begin
    if sz(1) eq 1 and sz(2) gt 1 then begin
       time_series = reform(time_series)  ; a column vector, change it
       if keyword_set(VERBOSE)then print,'Reforming timeseries'
 endif else MESSAGE, 'Must enter an array of data'
endif
```

```
if keyword_set(VERBOSE) then print
if keyword_set(VERBOSE) then print,'Performing S transform:'

if keyword_set(REMOVEEDGE)  then begin
  if keyword_set(VERBOSE) then  print,'Removing edges'
   ind = findgen(n_elements(time_series))
   r = poly_fit(ind,time_series,2,fit,yband,sigma,AM)
   ts_power =
sqrt(total(float(time_series)^2)/n_elements(time_series))
   if keyword_set(VERBOSE) then  print,'Sigma
is:',sigma,'power',ts_power,'ratio',sigma/ts_power
   if keyword_set(VERBOSE) then print, 'total
error',total(yband)/n_elements(yband)
   time_series = time_series - fit
   sh_len = n_elements(time_series)/10
 if sh_len gt 1 then begin
  wn = hanning(sh_len)
   time_series(0:sh_len/2-1) =
time_series(0:sh_len/2-1)*wn(0:sh_len/2-1)
   time_series(n_elements(time_series)-sh_len/2:*) =
 time_series(n_elements(time_series)-sh_len/2:*)*wn(sh_len/2: *)
  endif
endif


; here its dimension is one
sz = size(time_series)
if sz(2) ne 6 then begin
 if keyword_set(VERBOSE) then print,'Not complex data, finding
analytic signal.'
 ;take hilbert transfrom
 time_series = (hilbert(time_series,/analytic))
 ; the /2 is because the  spectrum is  *2 from an. sig.
 ; note that the nyquist point and DC are 2*normal in AS(t)
endif


length = n_elements(time_series)
spe_length = length/2
b = complexarr(length)
gw = complexarr(length)
h = fft(time_series,-1)

; do the different sampling cases here:
if (keyword_set(MAXFREQ))  then begin
 if maxfreq lt 1 then maxfreq = fix(length*maxfreq)
endif
if not(keyword_set(MINFREQ))  then begin
```

```
  if keyword_set(VERBOSE) then print,'Minimum Frequency is
0.'
  MINFREQ = 0  ; loop starts at 0
endif else begin
  if MINFREQ gt spe_length then begin
    MINFREQ = spe_length
    print,'MINFREQ too large, using default
value'
  endif
  if keyword_set(VERBOSE) then print,strcompress('Minimum
Frequency is '+string(MINFREQ)+'.')
endelse
if not(keyword_set(MAXFREQ))  then begin
  if keyword_set(VERBOSE) then print,strcompress('Maximum
Frequency is '+string(spe_length)+'.')
  MAXFREQ = spe_length
endif else begin
  if MAXFREQ gt spe_length then begin
    MAXFREQ = spe_length
    print,'MAXFREQ too large, using default
value'
  endif
  if keyword_set(VERBOSE) then print,strcompress('Maximum
Frequency is '+string(MAXFREQ)+'.')
endelse
if not(keyword_set(FREQSAMPLINGRATE))  then begin
  if keyword_set(VERBOSE) then print,'Frequency sampling
rate is 1.'
  FREQSAMPLINGRATE = 1
endif else if keyword_set(VERBOSE) then print,strcompress('Frequency
sampling rate is '+string(FREQSAMPLINGRATE)+'.')

if FREQSAMPLINGRATE eq 0 then FREQSAMPLINGRATE = 1     ; if zero then
use default

; check for errors in frequency parameters
if MAXFREQ lt MINFREQ then begin  ; if min > max, switch them
 temp = MAXFREQ
 MAXFREQ = MINFREQ
    MINFREQ= temp
    temp = 0
    print,'Switching frequency limits.'+strcompress(' Now, (MINFREQ =
'+string(MINFREQ) + ') and (MAXFREQ ='+string(MAXFREQ)+').')
endif

if MAXFREQ ne MINFREQ then begin
 if FREQSAMPLINGRATE gt (MAXFREQ - MINFREQ)   then  begin
```

```
 print,strcompress('FreqSamplingRate='+string(FREQSAMPLINGRAT E)+' too
big, using default = 1.')
  FREQSAMPLINGRATE = 1 ; if too big then use default
 endif
endif else FREQSAMPLINGRATE = 1 ; if there is only one frequency

spe_nelements = floor((MAXFREQ - MINFREQ )/FREQSAMPLINGRATE)+1
if keyword_set(VERBOSE) then print,strcompress('The number of frequency
elements is'+string(spe_nelements)+'.')


; Calculate the ST of the data
if keyword_set(ABS) and keyword_set(POWER) then begin
 print,'You are a moron! Defaulting to Local Amplitude Spectra
calculation'
 Power = 0
endif

if keyword_set(ABS) or keyword_set(POWER) then begin
    loc = fltarr(length,spe_nelements)
 if keyword_set(ABS)  then if keyword_set(VERBOSE) then
print,'Calculating Local Amplitude Spectra.'  $
  else if keyword_set(VERBOSE) then print,'Calculating
Local Power Spectra.'
 h = shift(h,-MINFREQ)
 if MINFREQ eq 0 then  begin
  gw = fltarr(length)
  gw(0) = 1
  loc(*,0) = abs(fft(h*gw,1))
 endif else begin
  f = float(MINFREQ)
     width = factor * length/f
  gw = gaussian_window(length,width)
  b = h * gw
     loc(*,0) = abs(fft(b,1))
 endelse
 for index = 1,spe_nelements-1 do begin
   f = float(MINFREQ) + index*FREQSAMPLINGRATE
     width = factor * length/f
  gw = gaussian_window(length,width)
  h = shift(h,-FREQSAMPLINGRATE)
     b = h * gw
     loc(*,index) = abs(fft(b,1))
  endfor
 if keyword_set(POWER) then loc = loc^2
endif else begin  ; calculate complex ST
 if keyword_set(VERBOSE) then print,'Calculating Local Complex
Spectra'
```

```
 loc = complexarr(length,spe_nelements)
 h = shift(h,-MINFREQ)
 if MINFREQ eq 0 then begin
  gw = fltarr(length)
  gw(0) = 1
  loc(*,0) = fft(h*gw,1)     ; 0 freq. equal to DC level
 endif else begin
  f = float(MINFREQ)
     width = factor * length/f
  gw = gaussian_window(length,width)
  b = h * gw
     loc(*,0) = fft(b,1)
 endelse
  for index = 1,spe_nelements-1  do begin
   f = float(MINFREQ) + index*FREQSAMPLINGRATE
  width = factor * length/f
     gw = gaussian_window(length,width)
     h = shift(h,-FREQSAMPLINGRATE)
     b = h * gw
     loc(*,index) = fft(b,1)
    endfor
endelse

if keyword_set(maskedges) then begin
 if maskedges eq 1 then maskthreshold=0.05
 if maskedges gt 0 and maskedges lt 1 then
maskthreshold=maskedges
 if maskedges gt 1 and maskedges le 100 then
maskthreshold=float(maskedges)/100. $
  else  maskthreshold=0.05
 edgets = findgen(length)/length
 st = s_trans(edgets,/abs)
 mask=where(st gt maskthreshold,maskcount) ; 5 % is good = 0.05
based on snooping around
 loc(mask) = 0
endif

if keyword_set(SAMPLINGRATE) then begin  ; make structure
;
 frequencies = (MINFREQ +
 findgen(spe_nelements)*FREQSAMPLINGRATE)/(SAMPLINGRATE*lengt h)
 time = findgen(length)*SAMPLINGRATE
 a = {st: loc, time: time,freq: frequencies}
 return,a
endif else return, loc
```

```
end ; end of function
```