CDP Spectral Information

(with Command Line Usage)

Functions to obtain information on spectral data

(Names in brackets mean that these are separate programs. The others are sub-modules of SPECINFO.)

CHANNEL
Returns PVOC channel number corresponding to frequency given

FREQUENCY
Returns centre frequency of the PVOC channel specified

[GET_PARTIALS]
Extract relative amplitudes of partials in a pitched source

LEVEL
Convert (varying) level of analysis file to a pseudo-soundfile, for viewing (1 window -> 1 sample)

OCTVU
Text display of the time varying amplitude of the spectrum, within octave bands

PEAK
Locate time varying energy centre of spectrum (text display)

[PEAK EXTRACT]
Extract spectral peaks from analysis file and write to a text file

PRINT
Print data in an analysis file as text to file

REPORT
Text report on location of frequency peaks in the evolving spectrum

WINDOWCNT
Returns the number of windows in the *infile*

Technical Discussion
FFT Analysis DATA

SPECINFO CHANNEL – Returns PVOC channel number corresponding to a given frequency

Usage

specinfo channel *infile frequency*

Parameters

infile – input analysis file made with PVOC
frequency – frequency in Hz

Understanding the SPECINFO CHANNEL Function

This function displays the number of the analysis channel in which the specified frequency is most likely to occur. Note however that in practice a frequency may be displaced by up to 8 channels on either side of the anticipated channel.

Note that this does not mean that the frequency you give is actually present in that channel, only that this is the channel it *would* be in if it were present. To determine which frequencies in a soundfile for which you might want channel information, you could use **REPITCH GETPITCH**, extracting the pitch data as a breakpoint file so that it can be examined easily. However, do note that there will be a weighting towards harmonic partials, though not exclusively, because GETPITCH is focused on 'pitch' as such; 'pitches' usually sound as they do because of the focusing effect of harmonically related partials (i.e., integer multiples of the fundamental).

Musical Applications

This function may have become largely redundant since CDP Release 4. It was needed earlier to specify a 'frequency divide channel number', but this is now handled internally by the relevant functions. Still, it may be instructive to look at which channels various frequencies occur in, for example when deciding on the number of analysis bands to have (frequency resolution).

End of SPECINFO CHANNEL

SPECINFO FREQUENCY – Returns the centre frequency of the PVOC channel specified

Usage

specinfo frequency *infile analysis_channel_number*

Parameters

infile – input analysis file made with PVOC
analysis_channel_number – number of the analysis channel for which you would like to know the corresponding frequency. (Range: from 1 to the number of channels in the analysis, as given for the **-N** flag of PVOC.)

Understanding the SPECINFO FREQUENCY Function

The same caution applies as was stated for SPECINFO CHANNEL: the frequency returned may or may not actually be present in that channel. This function simply identifies the frequencies that correspond to a given analysis channel. It's a mathematical correlation, not an actual analysis of a real sound.

Musical Applications

As with SPECINFO CHANNEL, largely redundant since CDP Release 4, but possibly useful for noting frequency and channel correlations.

End of SPECINFO FREQUENCY

GET_PARTIALS HARMONIC – Extract relative amplitudes of partials in a pitched source

Usage

get_partials harmonic 1-2 *inanalfile outfile fundamental threshold* [**-v**]
get_partials harmonic 3-4 *inanalfile outfile fundamental threshold time* [**-v**]

Example command line to get harmonic partials :

```
get_partials harmonic 1 four1grabbed.ana fourpartials.txt 100 0.054
```

Modes

1 Convert single analysis window to FRQ and amplitude list
2 Convert single analysis window to MIDI and amplitude list
3 Convert window at time *time* to FRQ and amplitude list
4 Convert window at time *time* to MIDI and amplitude list

Parameters

inanalfile – input analysis file; for Modes **1-2** it must have only a single analysis window (see SPEC GRAB)
outfile – output text file (see **-v** flag below)
fundamental – fundamental frequency for the harmonics
threshold – (relative) level below which partials are ignored – lower values capture more partials (Range: 0.000002 to 1.0)
time – (multiple window files only): time of window to use
-v – gives output in format for FILTER VARIBANK file

Understanding the GET_PARTIALS HARMONIC Function

The output for the above example command line as contained in *fourpartials.txt* (Mode **1**) is:

```
harm_partial    amplitude
200.000000      0.690151
400.000000      1.000000
500.000000      0.789301
700.000000      0.130861
900.000000      0.356995
1100.000000     0.166859
```

The input was *four.ana*, the word 'four' extracted from *count.wav*. It is 1.026 sec long, and a single window was 'grabbed' at time 0.5, forming the input file for GET_PARTIALS.

When the **-v** flag is invoked, the output is a text file in the format required as an input for FILTER VARIBANK, and looks like this:

```
0       100.000000  1.0
100000  100.000000  1.0
#
0     2 0.690151  4 1.000000  5 0.789301  7 0.1310861  9 0.356995  11 0.166859
10000 2 0.690151  4 1.000000  5 0.789301  7 0.1310861  9 0.356995  11 0.166859
```

See *CDP File Formants & Codes* for more information about this format.

In Mode **2**, the partials are displayed in MIDI Pitch Values (MPV), the above output becomes:

```
harm_partial    amplitude
55.349957       0.690151
67.349953       1.000000
71.213089       0.789301
77.038216       0.130861
81.389053       0.356995
84.863136       0.166859
```

In Modes **3** & **4**, the input does not have to be a single analysis window. Thus we can use the original analysis file from which we grabbed a window for Mode **1**, *four.ana*. If we give GET_PARTIALS the same time as we used for the grab (at 0.54 sec), we should get the same result:
`get_partials harmonic 3 four.ana fourpartials.txt 100 0.05 0.54`
The same partials are grabbed, but the amplitudes are very marginally different, e.g., 0.687556 instead of 0.690151. Again, the **-v** option is available.

Mode **4** does the same with the partials expressed in MIDI values.

Musical Applications

The importance of this function is that it is looking for *harmonic* partials, i.e., those related by an integer multiple. This is easily seen with the display in frequency values (200, 400, 500 etc.), but not so easily seen with the MIDI values, though they are accurate. (See our ***Equivalent Pitch Notations*** Chart.)

If you wanted the harmonics of a specific fundamental, e.g., MIDI 48 (Low C), you can look it up on the above-mentioned Chart (it's 130.81) or use **SNDINFO UNITS** to convert from MIDI to frequency. E.g., MIDI 50.2 = 148.538494. The output from GET_PARTIALS then enables you to work with the harmonics specifically related to that fundamental. Without the **-v** flag, the output text file can be used with **FILTER USERBANK**, and when the **-v** flag is invoked, the output text file can be used with **FILTER VARIBANK**, which can handle a time-varying filterbank.

ALSO SEE: **HARMONIC** in the Technical Glossary.

End of GET_PARTIALS HARMONIC

SPECINFO LEVEL – Convert (varying) level of the analysis file to a pseudo-soundfile (1 window -> 1 sample)

Usage

specinfo level *infile outsndfile*

Parameters

infile – input analysis file made with PVOC
outsndfile– a 'pseudo-soundfile' which can be viewed with a soundfile display program; it displays the amplitude shape of the sequence of windows. **Warning:** Do not attempt to play this file!

Understanding the SPECINFO LEVEL Function

LEVEL constructs a pseudo-soundfile in which the amplitude level of each analysis window becomes the amplitude level of each 'sample' in the display. This pseudo-soundfile can then be viewed with most sound waveform display programs. Each window is in fact a frame time: 1/analysis_rate. See the *Phase Vocoder* Reference Manual for a discussion of frames.

Musical Applications

The purpose of this program is to enable the user to get an overall picture of where in the analysis file the strongest signal lies.

The CDP **VIEWSF** program, available on the PC, will display the level of each analysis window on the 0 to 32767 scale. Use the zoom function to get a sample level display. The time shown is reckoned by the VIEWSF program to relate to samples, not to analysis windows, so it does not actually display the real time in the analysis file.

Each vertical line in the SPECINFO LEVEL pseudo-soundfile actually relates to *sample_rate/analysis_rate* samples. For example, 22050 (the sample rate) divided by 172 (the analysis rate) = 128. (Note that in the **DIRSF** display, the analysis rate is shown in the number-of-channels column.) **VIEWSF** thinks that this is only one sample (of sound) and will show the time accordingly. Therefore, to convert to the real time in the soundfile, you need to multiply the time displayed by the number of samples it really represents:

```
real_time = time_displayed * (sample_rate/analysis_rate).
```

This information indicates where in the sound the frequencies have the greatest amplitude, which may be useful when determining how and when to apply filtering to the sound.

End of SPECINFO LEVEL

PEAK EXTRACT – Extract spectral peaks from analysis file and write to a text file

Usage

peak extract 1 *inanalfile outtextfile winsiz peak floor lo hi* [**-h**tune] [**-a**] [**-m**] [**-q**] [**-z**]
peak extract 2 *inanalfile outtextfile winsiz peak floor lo hi* [**-h**tune] [**-a**] [**-m**] [**-q**] [**-z**] [**-f**]
peak extract 3 *inanalfile outtextfile winsiz peak floor lo hi* [**-h**tune] [**-a**] [**-m**] [**-q**] [**-z**] [**-f**]
peak extract 4 *inanalfile outtextfile winsiz peak floor lo hi* [**-h**tune] [**-a**] [**-m**] [**-q**] [**-z**] [**-f**]

Example command lines:

```
peak extract 1 four fourtxtm1.txt 2 2 0.001 86 1000
peak extract 1 four fourtxtm1.txt 2 2 0.001 86 1000 -m -q
peak extract 2 four fourtxtm1.txt 2 2 0.001 86 1000 -a -q -f
```

Modes

1 List the spectral peaks as they vary in time.
2 Stream the spectral peaks, using the maximum number of peaks found.
3 Stream the spectral peaks, using the most prominent pitches found.
4 List the most prominent peaks found as fixed average values.

Parameters

inanalfile – input analysis file
outtextfile – output text file containing the specified peak information
winsiz – the process divides each analysis window into a number of (overlapping) subwindows, and *winsize* determines the width of this subwindow, in semitones. Smaller values reduce the number of peaks found. Range: 1 - 96. (Recommended 12)
peak – in each subwindow, the process finds the median amplitude of the channels. It then looks for those channels whose amplitude is *peak* times louder than the median and initially assumes these are spectral peaks. Range: 1 - 1000. (Recommended 1.5+)
floor – the peak amplitudes must also exceed a minimum *floor* amplitude before they are accepted as true peaks – *floor* sets this level. Range: 0.0001 - 1. (Recommended ca .001)
lo – determines the lowest frequency acceptable for a retained peak. Range: *analysis channel width* to *nyquist*.
hi – determines the highest frequency acceptable for a retained peak. Range: *analysis channel width* to *nyquist*.
-h*tune* – If tune is **non-zero**, the process searches the peaks to discover if any of them are harmonics of other peaks. If they are, **the harmonic-peaks are excluded**. Peaks are judged to be harmonics if they are in tune with the expected frequency of a harmonic. The accuracy of tuning required (in semitones) is set by the value of *tune*. If tune is set to **zero**, **all** peaks are retained. Range: 0 - 6. (Recommended 1).

Important ways to specify how the peak data should be displayed (many examples of various settings and combinations of settings are given below):
-a – suppress amplitude information in the output
-m – express frequency data as (posibly fractional) MIDI values
-q – quantise frequency or midi output to the nearest quarter-tones of the tempered scale
-z – show peakfree segments in the data, by printing zeros in the output
-f – format the output as a data file for the FILTER VARIBANK process. Also see ***Files & Codes*** for more information on this file format.

- The **nyquist** frequency is half the sampling rate of the original soundfile from which the analysis file was derived.
- The **analysis channel width** is the nyquist divided by half of the analysis-channels parameter used to create the analysis file. The analysis-channels parameter is the **FFTsize**, such as 1024, the default value. Thus the **analysis channel width** is (SR/2) / (AnalChannels/2): (44100/2) / 1024/2) + 22050 / 512 = 43.066 (Hz). This would be the normal minimum value for *lo*, but it would be larger if there were fewer analysis channels (e.g., 512), and smaller if there were more. e.g., 2048 or 4096), and would also change if the sample rate were different.

Table of Optional Flag Combinations

Flags	Results
-m & **-q**	Frq in MIDI vals quantised to nearest ¼-tone
-a & **-f**	FILTER VARIBANK format, amplitudes set to 1.0 (so the field is not missing)
-q & **-f**	FILTER VARIBANK format, frq in Hz quantised to the nearest ¼-tone
-a & **-m** & **-f**	FILTER VARIBANK format, amplitudes set to 1.0, and frqs in MIDI vals
-a & **-q** & **-f**	FILTER VARIBANK format, amplitudes set to 1.0, and frqs in Hz
-m & **-q** & **-f**	FILTER VARIBANK format, frqs in MIDI vals quantised to the nearest ¼-tone, and all amplitudes are 0.0
-a & **-m** & **-q** & **-f**	FILTER VARIBANK format, frq in MIDI vals quantised to the nearest ¼-tone, and amplitudes set to 1.0

Understanding the PEAK EXTRACT function

PEAK EXTRACT searches for the peaks in the spectrum in an analysis file. Unlike REPITCH GETPITCH, it is not searching for a specific pitch in each window, but merely whatever peaks it can find. It is therefore more appropriate for finding the pitch content of multipitched sources.

The process is based on a method using median amplitudes, suggested by Bill Sethares (verbal communication at CCMIX in Paris) together with a best fit algorithm and semitone-bin assignment devised by T Wishart.

The various modes of the process are used as follows.

Mode 1 – Finding and Listing Peaks The window-time and the peaks in the window are listed, one window at a time. Output might look like this, where the columns represent time frq 1 amp1 [frq2 amp2 ....]:

```
0.072562 356.607666 0.000719
0.075465 354.271942 0.000862
0.078367 354.491180 0.001014
0.081270 353.553528 0.001173 435.571716 0.000764
0.087075 354.383575 0.001534
0.089977 356.247406 0.001930
0.092880 355.165039 0.001831
0.095782 349.447784 0.003151
0.101587 348.444855 0.004650 436.394318 0.002715
0.104490 349.419006 0.005525 379.522858 0.002046
0.107392 350.585388 0.006362
```

Note that different windows may have different numbers of peaks.

With the **-a** flag set, the amplitude information is suppressed, and output would look like this:
time frq1 [frq2 ...]

```
0.072562 356.607666
0.075465 354.271942
0.078367 354.491180
0.081270 353.553528 435.571716
0.087075 354.383575
0.089977 356.247406
0.092880 355.165039
0.095782 349.447784
0.101587 348.444855 436.394318
0.104490 349.419006 379.522858
0.107392 350.585388
```

With the **-z** flag set, any window with no peaks at all ('peakfree') will be shown in the output by zero-entries. Output would look like this:

```
0.000000 0 0
0.072562 356.607666 0.000719
0.075465 354.271942 0.000862
0.078367 354.491180 0.001014
0.081270 353.553528 0.001173 435.571716 0.000764
0.084172 0 0
0.087075 354.383575 0.001534
0.089977 356.247406 0.001930
0.092880 355.165039 0.001831
0.095782 349.447784 0.003151
0.098685 0 0
0.101587 348.444855 0.004650 436.394318 0.002715
0.104490 349.419006 0.005525 379.522858 0.002046
0.107392 350.585388 0.006362
```

With the **-m** flag set, the frequency values are converted to MIDI values. Output would look like this:

```
0.072562 65.362007 0.000719
0.075465 65.248245 0.000862
0.078367 65.258949 0.001014
0.081270 65.213097 0.001173 68.824883 0.000764
0.087075 65.253700 0.001534
0.089977 65.344513 0.001930
0.092880 65.291832 0.001831
0.095782 65.010880 0.003151
0.101587 64.961121 0.004650 68.857544 0.002715
0.104490 65.009453 0.005525 66.440201 0.002046
0.107392 65.067146 0.006362
```

With the **-m** and **-q** flags set, the frequency values are converted to MIDI *and* quantised to the nearest quartertone. The output would look like this:

```
0.072562 65.5 0.000719
0.075465 65.0 0.000862
0.078367 65.5 0.001014
0.081270 65.0 0.001173 69.0 0.000764
0.087075 65.5 0.001534
0.089977 65.5 0.001930
0.092880 65.5 0.001831
0.095782 65.0 0.003151
0.101587 65.0 0.004650 69.0 0.002715
0.104490 65.0 0.005525 66.5 0.002046
0.107392 65.0 0.006362
```

With only the **-q** flag set, the frequency values are converted to the nearest quartertone, but output as frequency values. The output would look like this:

```
0.072562 359.461395 0.000719
0.075465 349.228241 0.000862
0.078367 359.461395 0.001014
0.081270 349.228241 0.001173 440.000000 0.000764
0.087075 359.461395 0.001534
0.089977 359.461395 0.001930
0.092880 359.461395 0.001831
0.095782 349.228241 0.003151
0.101587 349.228241 0.004650 440.000000 0.002715
0.104490 349.228241 0.005525 380.836090 0.002046
0.107392 349.228241 0.006362
```

(Compare this with the first table of output.)

Modes 2 to 4: Streaming the data

- **NB:** In some of the streamed data displays, below, the lines may be so long that they "wrap-around" because they will not fit on the displayed page on your computer. However, the data itself does not wrap around, and each row contains the same number of columns, despite appearances to the contrary.
- The aim of streaming the data is to track the peaks as they pass from one window to the next.
- A stream will corresponds to a pair of columns of data (frequency + amplitude) in the output.
- In general not all the window will have the same number of peaks.
- PEAK EXTRACT has two methods to assign peaks to streams: Mode **2** uses the **maximum number of peaks** found, and Mode **3** uses the **most prominent pitches** found.

Mode 2: Stream the spectral peaks, using the maximum number of peaks found

This process first looks for the windows with the maximum number of peaks. It uses this as the number of streams it will output.

However, some window may have less than the maximum number of peaks. For these windows, the process predicts where the centre frequency of the streams might be and then assigns the actual peaks to the available streams to get the best fit of the data: i.e., the deviation of the *actual peak frequency* from the *predicted stream frequency* is at a minimum.

The remaining streams (which have no corresponding peaks) are given a **zero amplitude** at this time and assigned to the *predicted frequency* of the stream.

The output might look like this time frq1 amp1 [frq2 amp2 ...]:

```
0.197370 122.435066 0.000000 353.447144 0.076715 433.644592 0.000000
         525.059875 0.025843 572.628113 0.000000
0.200272 130.600586 0.001569 355.624725 0.071684 433.644592 0.000000
         533.026001 0.000000 572.628113 0.000000
0.203175 122.435066 0.000000 354.864227 0.068599 433.644592
0.000000          526.530640 0.017870 572.628113 0.000000
0.206077 122.435066 0.000000 249.924759 0.047720 353.834137 0.067311
         533.026001 0.000000 572.628113 0.000000
0.208980 122.435066 0.000000 246.898758 0.052282 350.662476
0.067316          533.026001 0.000000 572.628113 0.000000
0.211882 122.435066 0.000000 348.227264 0.063584 406.975525 0.024158
         525.834778 0.034635 572.628113 0.000000
0.214785 122.435066 0.000000 249.414642 0.069781 346.354675
0.055594          533.026001 0.000000 572.628113 0.000000
0.217687 122.435066 0.000000 251.439682 0.077112 433.644592 0.000000
         533.026001 0.000000 703.172241 0.000754
0.220590 122.435066 0.000000 251.685486 0.078068 433.644592
0.000000          533.026001 0.000000 572.628113 0.000000
0.223492 122.435066 0.000000 248.102814 0.075569 433.644592 0.000000
         527.619629 0.037400 572.628113 0.000000
```

Note that there are the same number of entries in each row, and the data corresponds to 5 streams. Also note that most entries in streams 1

and 5 have zero amplitude. At those times, these streams have no corresponding peak.

Mode 3: Stream the spectral peaks, using the most prominent pitches found

This process assumes that the spectrum is relatively stable. It scans the entire file, and counts the peaks occurring within every semitone interval. It then looks for the most prominent (the most frequently visited) semitones, the peaks in the '*semitone-spectrum*'. The number of these peaks is used as the number of streams, and the peak-frequencies determine the predicted centres of the streams. The peaks in each window are then assigned to these streams.

In this process, some window may have less peaks than the number of streams. The peaks are assigned to the streams using the best-fit approach described above. Also, some window may have more peaks than the number of streams. Again, a best fit procedure determines which peaks to retain and which to discard.

The output might look like this time frq1 amp1 [frq2 amp2 ....]:

```
0.197370 187.935364 0.000000 223.494080 0.000000 250.863602
0.000000          353.447144 0.076715 446.988159 0.000000 525.059875
0.025843           632.136719 0.000000
0.200272 187.935364 0.000000 223.494080 0.000000 250.863602 0.000000
         355.624725 0.071684 446.988159 0.000000 528.049866 0.027765
         632.136719 0.000000
0.203175 187.935364 0.000000 223.494080 0.000000 250.863602 0.000000
         354.864227 0.068599 446.988159 0.000000 526.530640 0.017870
         632.136719 0.000000
0.206077 187.935364 0.000000 223.494080 0.000000 250.863602 0.000000
         353.834137 0.067311 446.988159 0.000000 531.561462 0.000000
         632.136719 0.000000
0.208980 187.935364 0.000000 223.494080 0.000000 250.863602 0.000000
         350.662476 0.067316 446.988159 0.000000 531.561462 0.000000
         632.136719 0.000000
0.211882 187.935364 0.000000 223.494080 0.000000 250.863602 0.000000
         348.227264 0.063584 406.975525 0.024158 525.834778 0.034635
         632.136719 0.000000
0.214785 187.935364 0.000000 223.494080 0.000000 250.863602 0.000000
         346.354675 0.055594 446.988159 0.000000 531.561462 0.000000
         632.136719 0.000000
0.217687 187.935364 0.000000 223.494080 0.000000 251.439682 0.077112
         354.774719 0.000000 446.988159 0.000000 531.561462 0.000000
         632.136719 0.000000
0.220590 187.935364 0.000000 223.494080 0.000000 251.685486 0.078068
         354.774719 0.000000 446.988159 0.000000 531.561462 0.000000
         632.136719 0.000000
0.223492 187.935364 0.000000 223.494080 0.000000 248.102814 0.075569
         354.774719 0.000000 446.988159 0.000000 527.619629 0.037400
         632.136719 0.000000
```

Note that in both these cases, the output data is in the format of a FILTER VARIBANK data file. However, in general the amplitudes obtained from the channel peaks are very low. If the **-f** flag is set, these amplitudes are normalised in the output.

Format of the output with Modes 1-3

If the **-f** and **-a** flags are set, the amplitude data is discarded. However, the FILTER VARIBANK data format requires an amplitude be assigned to each pitch entry. So the amplitudes in the output **default to 1.0.**

Returning to Mode 2 (stream the spectral peaks, using the maximum number of peaks found)

The original output might look this time frq1 amp1 [frq2 amp2 ...]:

```
0.197370 122.435066 0.000000 353.447144 0.076715 433.644592
0.000000          525.059875 0.025843 572.628113 0.000000
0.200272 130.600586 0.001569 355.624725 0.071684 433.644592 0.000000
         533.026001 0.000000 572.628113 0.000000
0.203175 122.435066 0.000000 354.864227 0.068599 433.644592
0.000000          526.530640 0.017870 572.628113 0.000000
0.206077 122.435066 0.000000 249.924759 0.047720 353.834137 0.067311
         533.026001 0.000000 572.628113 0.000000
0.208980 122.435066 0.000000 246.898758 0.052282 350.662476
0.067316          533.026001 0.000000 572.628113 0.000000
0.211882 122.435066 0.000000 348.227264 0.063584 406.975525 0.024158
         525.834778 0.034635 572.628113 0.000000
0.214785 122.435066 0.000000 249.414642 0.069781 346.354675
0.055594          533.026001 0.000000 572.628113 0.000000
0.217687 122.435066 0.000000 251.439682 0.077112 433.644592 0.000000
         533.026001 0.000000 703.172241 0.000754
0.220590 122.435066 0.000000 251.685486 0.078068 433.644592
0.000000          533.026001 0.000000 572.628113 0.000000
0.223492 122.435066 0.000000 248.102814 0.075569 433.644592 0.000000
         527.619629 0.037400 572.628113 0.000000
```

With the **-f** (FILTER VARIBANK format) and **-a** (suppress amplitude information) flags set, it would look like this (note the default amplitudes at 1.0):

```
0.197370 122.435066 1.0 353.447144 1.0 433.644592 1.0 525.059875 1.0
         572.628113 1.0
0.200272 130.600586 1.0 355.624725 1.0 433.644592 1.0 533.026001 1.0
         572.628113 1.0
0.203175 122.435066 1.0 354.864227 1.0 433.644592 1.0 526.530640 1.0
         572.628113 1.0
0.206077 122.435066 1.0 249.924759 1.0 353.834137 1.0 533.026001 1.0
         572.628113 1.0
0.208980 122.435066 1.0 246.898758 1.0 350.662476 1.0 533.026001 1.0
         572.628113 1.0
0.211882 122.435066 1.0 348.227264 1.0 406.975525 1.0 525.834778 1.0
         572.628113 1.0
0.214785 122.435066 1.0 249.414642 1.0 346.354675 1.0 533.026001 1.0
         572.628113 1.0
0.217687 122.435066 1.0 251.439682 1.0 433.644592 1.0 533.026001 1.0
         703.172241 1.0
0.220590 122.435066 1.0 251.685486 1.0 433.644592 1.0 533.026001 1.0
         572.628113 1.0
0.223492 122.435066 1.0 248.102814 1.0 433.644592 1.0 527.619629 1.0
         572.628113 1.0
```

We can produce **MIDI value output** by *also* setting the **-m** flag, when the output might look like this:

```
0.197370 46.854156 0.000000 65.207893 0.076715 68.748116
0.000000          72.059738 0.025843 73.561134 0.000000
0.200272 47.971893 0.001569 65.314224 0.071684 68.748116
0.000000          72.320427 0.000000 73.561134 0.000000
0.203175 46.854156 0.000000 65.277161 0.068599 68.748116 0.000000
          72.108162 0.017870 73.561134 0.000000
0.206077 46.854156 0.000000 59.207882 0.047720 65.226837
0.067311          72.320427 0.000000 73.561134 0.000000
0.208980 46.854156 0.000000 58.996990 0.052282 65.070953
0.067316          72.320427 0.000000 73.561134 0.000000
0.211882 46.854156 0.000000 64.950302 0.063584 67.649261 0.024158
          72.085274 0.034635 73.561134 0.000000
0.214785 46.854156 0.000000 59.172508 0.069781 64.856956
0.055594          72.320427 0.000000 73.561134 0.000000
0.217687 46.854156 0.000000 59.312504 0.077112 68.748116
0.000000          72.320427 0.000000 77.116493 0.000754
0.220590 46.854156 0.000000 59.329422 0.078068 68.748116 0.000000
          72.320427 0.000000 73.561134 0.000000
0.223492 46.854156 0.000000 59.081215 0.075569 68.748116
0.000000          72.143936 0.037400 73.561134 0.000000
```

And we can **quantise** this MIDI data to the *nearest quartertone*, by also setting the **-q** flag (quantise to the nearest quarter-tone), when the output might look like this:

```
0.197370 47.0 1.0 65.0 1.0 68.5 1.0 72.0 1.0 73.5 1.0
0.200272 48.0 1.0 65.5 1.0 68.5 1.0 72.5 1.0 73.5 1.0
0.203175 47.0 1.0 65.5 1.0 68.5 1.0 72.0 1.0 73.5 1.0
0.206077 47.0 1.0 59.0 1.0 65.0 1.0 72.5 1.0 73.5 1.0
0.208980 47.0 1.0 59.0 1.0 65.0 1.0 72.5 1.0 73.5 1.0
0.211882 47.0 1.0 65.0 1.0 67.5 1.0 72.0 1.0 73.5 1.0
0.214785 47.0 1.0 59.0 1.0 65.0 1.0 72.5 1.0 73.5 1.0
0.217687 47.0 1.0 59.5 1.0 68.5 1.0 72.5 1.0 77.0 1.0
0.220590 47.0 1.0 59.5 1.0 68.5 1.0 72.5 1.0 73.5 1.0
0.223492 47.0 1.0 59.0 1.0 68.5 1.0 72.0 1.0 73.5 1.0
```

Or we can produce **quantised MIDI output** but retain the relative amplitudes of each peak by setting the **-f** (FILTER VARIBANK format), the **-m** (express frequency in MIDI values) and the **-q** (quantise to the nearest quarter-tone) flags. Now the output might look like this:

```
0.197370 47.0 0.000000 65.0 0.076715 68.5 0.000000 72.0 0.025843
          73.5 0.000000
0.200272 48.0 0.001569 65.5 0.071684 68.5 0.000000 72.5 0.000000
          73.5 0.000000
0.203175 47.0 0.000000 65.5 0.068599 68.5 0.000000 72.0 0.017870
          73.5 0.000000
0.206077 47.0 0.000000 59.0 0.047720 65.0 0.067311 72.5 0.000000
          73.5 0.000000
0.208980 47.0 0.000000 59.0 0.052282 65.0 0.067316 72.5 0.000000
          73.5 0.000000
```

```
0.211882 47.0 0.000000 65.0 0.063584 67.5 0.024158 72.0 0.034635
         73.5 0.000000
0.214785 47.0 0.000000 59.0 0.069781 65.0 0.055594 72.5 0.000000
         73.5 0.000000
0.217687 47.0 0.000000 59.5 0.077112 68.5 0.000000 72.5 0.000000
         77.0 0.000754
0.220590 47.0 0.000000 59.5 0.078068 68.5 0.000000 72.5 0.000000
         73.5 0.000000
0.223492 47.0 0.000000 59.0 0.075569 68.5 0.000000 72.0 0.037400
         73.5 0.000000
```

Again, we can also quantise the data, but keep it in frequency format. To do this we need to set the **-f** (FILTER VARIBANK format) and the **-q** (quantise to the nearest quarter-tone) flags, and the output would look like this:

```
0.197370 123.470825 0.000000 349.228241 0.076715 427.474060
0.000000         523.251160 0.025843 570.609375 0.000000
0.200272 130.812790 0.001569 359.461395 0.071684 427.474060 0.000000
         538.583557 0.000000 570.609375 0.000000
0.203175 123.470825 0.000000 359.461395 0.068599 427.474060
0.000000         523.251160 0.017870 570.609375 0.000000
0.206077 123.470825 0.000000 246.941650 0.047720 349.228241 0.067311
         538.583557 0.000000 570.609375 0.000000
0.208980 123.470825 0.000000 246.941650 0.052282 349.228241
0.067316         538.583557 0.000000 570.609375 0.000000
0.211882 123.470825 0.000000 349.228241 0.063584 403.481781 0.024158
         523.251160 0.034635 570.609375 0.000000
0.214785 123.470825 0.000000 246.941650 0.069781 349.228241
0.055594         538.583557 0.000000 570.609375 0.000000
0.217687 123.470825 0.000000 254.177597 0.077112 427.474060 0.000000
         538.583557 0.000000 698.456482 0.000754
0.220590 123.470825 0.000000 254.177597 0.078068 427.474060
0.000000         538.583557 0.000000 570.609375 0.000000
```

Mode 4: Finding the average peak-frequencies

If the analysis data represents a non-varying spectrum, we may simply want to know what the **frequencies of the peaks** are, **on average**.

Mode **4**: first use the streaming procedure of Mode **3**, then determine the **average pitch** of each stream. We also sum the amplitude of every window in each stream, then use the stream amplitude-sums to determine the **relative level** of each peak. For the same input data, the output would look like this (note that there is no *time* information, just average_frequency relative_level_of_amplitude_sums:

```
93.967637 0.000294
111.727164 0.000460
125.471817 0.000938
149.080771 0.003545
167.896870 0.006329
188.224409 0.207316
250.011853 0.386916
351.188075 1.000000
440.523979 0.117659
530.458239 0.047925
629.244392 0.001581
843.606908 0.000466
```

```
1192.913225 0.000074
```

Note that there is no timing information here as the spectrum is assumed not to vary in time.

With the **-q** (quantise to the nearest quarter-tone) and **-m** (use MIDI values for frequency) flags set, the output in MIDI values is quantised to the nearest quarter-tone, and looks like this:

```
42.5 0.000294
45.5 0.000460
47.5 0.000938
50.5 0.003545
52.5 0.006329
54.5 0.207316
59.0 0.386916
65.0 1.000000
69.0 0.117659
72.0 0.047925
75.0 0.001581
80.5 0.000466
86.5 0.000074
```

If the **-f** flag is *also* used in Mode **4**, the FILTER VARIBANK data format requires time information, so an *arbitrary* data end-time of 1000 seconds is set (16 minutes, 40.2 seconds). If a later end-time is required for the output data, the output file can be edited.

With the **-f** (FILTER VARIBANK format), the **-m** (use MIDI values for frequency), and the **-q** (quantise to the nearest quarter-tone) flags set, the output is **MIDI quantised to nearest quarter-tone**. And with the **-a** (suppress amplitude information in the output) flag *also* set, **amplitudes all are set to 1.0** and the output data would look like this time quantised_MIDIfrq1 amp1 [quantised_MIDIfrq2 amp2 ...]:

```
0.0   42.5 1.0 45.5 1.0 47.5 1.0 50.5 1.0 52.5 1.0 54.5 1.0 59.0 1.0
65.0 1.0 69.0 1.0 72.0 1.0 75.0 1.0 80.5 1.0 86.5 1.0
1000  42.5 1.0 45.5 1.0 47.5 1.0 50.5 1.0 52.5 1.0 54.5 1.0 59.0 1.0
65.0 1.0 69.0 1.0 72.0 1.0 75.0 1.0 80.5 1.0 86.5 1.0
```

Musical Applications

PEAK EXTRACT searches for the peaks in the spectrum in an analysis file, regardless of whether the spectrum is pitched or not. It is appropriate for finding the pitch foci of multi-pitched sources.

The output data can be used in processes where pitch or frequency data is required. In particular, the FILTER VARIBANK output can be used to create filters which tune material to the spectrum found in the analysis file.

ALSO SEE: the entry on the FILTER VARIBANK **file format** in *CDP File Formats & Codes*.

End of PEAK EXTRACT

SPECINFO OCTVU – Text display of the time-varying amplitude of the spectrum, within octave bands

Usage

specinfo octvu *infile outtextfile time_step* [**-f***fundamental*]

Parameters

infile – input analysis file made with PVOC
outtextfile – output text file
time_step – spread (in milliseconds) of the time points at which information is gathered; band energy is totalled over each *time_step*
-f*fundamental* – Octave bands are centred on octave transpositions of *fundamental*, given in Hz. Default: centred on divisions of the Nyquist frequency (i.e., ½ the sample rate).

Understanding the SPECINFO OCTVU Function

The display shows frequency bands horizontally, and the readings at the time points for each band vertically, underneath each frequency band heading. The fundamental will influence the frequency starting point for the display. Scanning vertically down the time point readings will quickly show in which band and at what time the maximum energy is located. (Using a mathematically 'easy' division of the second (e.g., a *time_step* of 250 for ¼ sec, or of 500 for ½ sec) will make it easier to visualise the energy across the time of the sound.)

Note that the reported values are **relative** levels only. The lowest band includes all energy down to '0Hz'.

Musical Applications

SPECINFO OCTVU provides an overview of how the spectrum is evolving. (Also see SPECINFO PEAK.) This information can be useful when making decisions, for example, about when and at which frequencies to apply filtering to a sound. Also, some functions, such as STRANGE SHIFT, have a 'frequency divide' parameter to locate processes above or below a certain frequency. SPECINFO OCTVU can be used to help work out what frequency value to give to this parameter. Note that the *time_step* parameter enables you to set the resolution in time of the information gathered – a 'zoom' control.

End of SPECINFO OCTVU

SPECINFO PEAK – Locate time-varying energy centre of the spectrum

Usage

specinfo peak *infile outtextfile* [**-c***cutoff_frq*] [**-t***timewindow*] [**-f***frqwindow*] [**-h**]

Parameters

infile – input analysis file made with PVOC
outtextfile – output text file of lines of data in this form:

WINDOW starttime endtime : PEB lofrq TO hifrq
(where 'PEB' means Peak Energy Band)

-c*cutoff_frq* above which spectral search begins (Default = 8.00Hz)
-t*timewindow* for energy averaging: in SECONDS (Default = 0.1000)
-f*frqwindow* for energy averaging: in OCTAVES (Min 0.02. Default = 0.50)
-h adjust result for sensitivity of ear

Understanding the SPECINFO PEAK Function

PEAK searches the spectrum for the time-varying peak energy, i.e., the highest amplitude within the given frequency band. You can define the size of the frequency bands you wish to compare (minimum is one channel width). You can also ask the program to average the energy calculations over a certain time-window, which you specify. This avoids tracking every micro-fluctuation of energy in the spectrum.

The output text file lists the start time of each window (based on the window durations specified by you) and indicates the lower and upper frequency of the band of maximum energy in each of these windows.

The window times shown in the *outtextfile* are generated automatically, starting at time zero and stepping through the file in time-chunks specified by you as *timewindow*. The *timewindow* parameter therefore operates as a zoom mechanism. PEB shows you the lower and upper frequency of the band of frequencies which has the peak energy in that particular window.

Musical Applications

The possibility of widening the time window chunks indicates how this function can return useful information. The objective is to locate which frequency band contains an energy peak in any given time period. *Timewindow* gives time resolution and *frqwindow* gives frequency resolution in parts of an octave.

Note that the *timewindow* is not the same as the analysis window. The duration of an analysis window is the value given for **-N** divided by the sample rate: e.g., 1024/44100 = an analysis window of 0.0232 sec duration. A *timewindow* of e.g. 0.1 sec will therefore average over about 5 analysis windows and will give 10 data entries per second of sound. This will show where on average in those 5 windows, the peak energy occurs.

The information in *outtextfile* will show which frequency bands contain high amplitude data, and where they occur, useful for CUT and FILTER operations.

The *cutoff_frq* parameter enables you to avoid gathering unncessary information.

End of SPECINFO PEAK

SPECINFO PRINT – Print data in an analysis file as text to file

Usage

specinfo print *infile outtextfile time* **-w***windowcnt*

Parameters

infile – input analysis file made with PVOC
outtextfile – name of text file in which to store the data
time – time in *infile* at which the printout begins
windowcnt – number of windows to print

Warning: using more than a few windows will generate a huge textfile.

Understanding the SPECINFO PRINT Function

SPECINFO PRINT provides a way to examine in detail the contents of a PVOC analysis file. It displays amplitude and frequency data for each channel, window by window, starting at *time*. The sum of all values in a single window will give the total amplitude of the sound in that window as a value between 0 and 1. The value in each channel therefore indicates the amplitude contribution of that channel to the total amplitude of the window.

Musical Applications

This function gives a complete listing of the amplitude and frequency content of an analysis window. The more prominent frequencies will presumably have higher amplitude values. Scanning this data should indicate where key sonic events occur.

End of SPECINFO PRINT

SPECINFO REPORT – Text report on location of frequency peaks in the evolving spectrum

Usage

specinfo report mode *infile outtextfile* **-f***N* | **-p***N* [**-i**] *peaks* [**-b***bt*] [**-t***tp*] [**-s***val*]

Modes

1 Report on frequency and time of spectral peaks
2 Report spectral peaks in loudness order
3 Report spectral peaks as frequency only (no time data)
4 Report spectral peaks in loudness order, as frequency only (no time data)

Parameters

infile – input analysis file made with PVOC
outtextfile – output text file
-f – extract formant envelope linear frequency-wise, using 1 point for every *N* equally-spaced frequency-channels
-pN – extract formant envelope linear pitchwise, using *N* equally-spaced pitch-bands per octave
-i – quicksearch for formants (less accurate)
peaks – (maximum) number of peaks to find. (Range: 1–16)
-b*bt* – bottom frequency to start peak search
-t*tp* – top frequency to stop peak search
-s*val* – number of windows over which the peaks are averaged. (Range 2–4097, Default 9.)
Warning: high values slow up the program.

Bottom frequency and top frequency may vary over time.

See **FORMANTS GET** for a discussion of the formant extraction options.

Understanding the SPECINFO REPORT Function

The report shows the times on the left and the *peaks* peaks requested in columns to the right of each time. By glancing over the data, one can tell at what time and in which frequency bands the peaks occur. This shows the time and frequency location of the spectral energy of the sound.

Musical Applications

SPECINFO REPORT gives more detailed information on where the spectral peaks are (especially the persistent ones, if you use the **-s** option). The information about spectral peaks will probably also tell you something about where formants are, or where strong audible pitch components are located.

End of SPECINFO REPORT

SPECINFO WINDOWCNT – Returns the number of analysis windows in *infile*

Usage

specinfo windowcnt *inanalfile*

Parameters

inanalfile – the only parameter, a PVOC analysis file

Understanding the SPECINFO WINDOWCNT Function

SPECINFO WINDOWCNT is a simple utility which reports on the number of (time) windows there are in the analysis file.

Musical Applications

Several spectral processes have a **WINDOWS** parameter. This function ensbles you to check the total number of windows, so that you don't exceed it.

End of SPECINFO WINDOWCNT

Technical Discussion about FFT Analysis DATA

An overview of the many aspects of this complex subject can be obtained by looking through relevant entries in the **CDP Technical Glossary**, especially **ANALYSIS**, **ANALYSIS SETTINGS**, **FRAMES**, **PARTIALS**, **SPECTRAL DOMAIN**, **SPECTRAL ENVELOPE**, **SPECTRUM**, **WINDOW**, and the various entries about **FFT** itself.

Regarding the forms of information about spectral data that the SPECINFO programs provide, a quick look at the respective differences can be had by examining the entries in **CDP File Formats & Codes** listed below:

- **OCTVU**,
- **PEAK**, and
- **REPORT**

End of Technical Discussion

Last Updated 18 Sep. 2016
Documentation: Archer Endrich, revised R. Fraser