Getting started with STRAIGHT in command mode

Hideki Kawahara
Faculty of Systems Engineering, Wakayama University, Japan

May 5, 2007

Contents

(note: This document is still under construction at 6 May 2007. It will be updated in a near future.)

1 Introduction

STRAIGHT[1] is a high-quality speech analysis, modification and synthesis system based on a simple channel VOCODER concept (in other words a source filter model).

This text is prepared for providing a basic knowledge to use STRAIGHT in command mode and to provide tips to write your own code using STRAIGHT. This text is based on the latest STRAIGHT (V40_006b) and new four wrapper functions (exstraightsource, exstraightspec, exstraightsynth and exstraightAPind). Underlying concept and basic principles of operations of STRAIGHT and its applications can be found in a review article [2] published in *Acoustical Science and Technology*. The following hyper text link (only visible and accessible from the HTML version of this document) provides a direct access to the paper.

> Please skip sections with (*) on its title for the first time. They are sometimes too detailed for beginners. However, once you understand the basic operations of STRAIGHT, please have time to check them when you have questions. They provide technical details and practical workarounds useful for some situations.

1.1 Highly reliable new F0 extractor and notes on aperiodicity (*)

The latest STRAIGHT directory also consists of a new F0 extraction program (MulticueF0v14). The underlying concept of the program is presented at INTERSPEECH2005 Lisbon [5]. It is recommended to use this new F0 extractor instead of using the default F0 extractor of STRAIGHT, however the new F0 extractor was not integrated into the new wrapperfunction (exstraightsource) yet. An introduction how to use the new extractor in the course of STRAIGHT analysis, modification and resynthesis procedures can be found later in this document (5).

Aperiodicity index is also discussed in the appendix in some details. A new set of procedures to extract aperiodic information is under development and will be introduced by the end of May 2007. In the new aperiodicity extractor, interfering factors such as frequency modulation (FM) and amplitude modulation (AM) of constituent harmonic components and temporal variations in the vocal tract transfer function, as well as probabilistic fluctuations of estimates are taken into account in aperiodicity calculation. The new procedure found to yield more reliable estimate of aperiodicity index than the default procedure built in current STRAIGHT implementation. Please note that the current aperiodicity index is not highly dependable, however, far better than nothing.

2 Installing STRAIGHT

You can get the latest STRAIGHT from http://www.wakayama-u.ac.jp/~kawahara/puzzlet/STRAIGHTtipse/. However, please be aware that the page is protected based on the IP-address of the accessing machine. If you are unable to make access, please mailto:kawahara@sys.wakayama-u.ac.jp

Please download STRAIGHT by clicking the link written as The downloaded file is compressed (using tar+gzip). Please expand it to get a folder consisting of STRAIGHT codes. Please add the folder to your Matlab search path. This completes installation of STRAIGHT.

3 Required environment

STRAIGHT is implemented on Matlab with signal processing toolbox installed. It has been tested on Matlab v7 (7.0.4.352 (R14) Service Pack 2). It is reported that STRAIGHT is also compatible with Matlab v5 or v6. mailto:kawahara@sys.wakayama-u.ac.jp if you find anything defective on these environment. For information about Matlab, please check http://www.mathworks.com/

All the codes of STRAIGHT is written in Matlab. It means that STRAIGHT is basically platform independent. (Platform dependence can be found only when manipulating RAW data format files.) STRAIGHT is reported to be functional on Max OS X, Windows (2000, XP), UNIX. It was reported that it is not impossible to use STRAIGHT on octave (an open source Matlab clone?) with minor modifications.

Current version of STRAIGHT is an experimental program evolved from a collection of elementary procedures. It consists of many unoptimized codes, they are intentionally left unchanged to preserve history of updates. This is the reason why current STRAIGHT requires a huge amount of memory. It is highly recommended to run it on machines having 1GB memory or more. (However, this does not imply that STRAIGHT is a memory eager algorithm. The STRAIGHT algorithm is originally designed for realtime applications in mind and needs only 100ms of memory duration. Please expect a memory efficient version in the near future.)

4 Using default parameters

This section introduces the simplest analysis and re-synthesis procedure.

Reading speech from a file

The first step is to read a file consisting of a speech signal. A file `vaiueo2d.wav` is used to illustrate the following steps. The file is located in the STRAIGHT directory. The following command reads the speech signal into the variable `x` and sets the sampling frequency (Hz) to `fs`.

```
[x,fs]=wavread('vaiueo2d.wav');
```
use audioread() instead

Extracting excitation source parameters

The next step is to extract source parameters. Source parameter consists of a fundamental frequency `f0raw` and periodicity indices `ap` at each frequency band. The following command does it.

```
[f0raw,ap]=exstraightsource(x,fs);
```

The fundamental frequency information is used to guide the following spectral information extraction.

Extracting smoothed time-frequency representation (STRAIGHT spectrum)

A time-frequency representation `n3sgram`, that is an F0 adaptively smoothed spectrogram, is extracted using the following command.

```
n3sgram=exstraightspec(x,f0raw,fs);
```

The algorithm for eliminating interferences caused by a periodic excitation based on an extended pitch synchronous analysis and an optimum F0 adaptive smoothing based on a spline theory is the core of STRAIGHT (and is a stateless algorithm). Let's call the time-frequency representation as STRAIGHT spectrogram later on. Accordingly, let's call its time slice as a STRAIGHT spectrum.

Re-synthesizing speech

Speech re-synthesis without modification is just calling the following command.

```
sy = exstraightsynth(f0raw,n3sgram,ap,fs);
```

The synthesized speech signal is stored in the variable `sy`. The following Matlab command reproduces it from an audio output.

```
soundsc(sy,fs);
```

4.1 Reading and writing audio files

Output rms level of the STRAIGHT synthesis procedure `exstraightsynth` is normalized to -22dB (relative to maximum level of signed 16bit integer). The rms value is calculated using (roughly speaking) voiced part only. (This is due to some historical reason.) Consequently, it is necessary to reduce the amplitude to meet requirements of the Matlab `wavwrite` function when storing it in WAVE format. The following command is an example implemented in the old GUI-STRAIGHT.

```
wavwrite(sy/32768,fs,16,'test.wav');
```

STRAIGHT also supports input and output of (a subset of) AIFF format. No scaling is necessary, because the writing function `aiffwrite` writes 1 in Matlab as 1 LSB in the output file.

```
aiffwrite(sy,fs,16,'test.aiff');
```

(This scaling introduced a bug in STRAIGHT spectral analysis. The analysis procedure was optimized to signals in a 16bit integer range. This bug is fixed in the analysis function `exstraightspec` of this STRAIGHT release (V40).)

5 Using a new F0 extractor for STRAIGHT

A new F0 extractor using multiple F0 cues was introduced recently and reported at INTERSPEECH2005 [5]. The new F0 extractor can be used in STRAIGHT instead of using its default F0 extractor. The new F0 extractor is especially useful when manipulating expressive speech, where irregular vocal fold vibration patterns are more frequently found. Please use the following steps to use this new procedure.

Reading speech data from a file
This part is the same as the default procedure.

```
[x,fs]=wavread('vaiueo2d.wav');
```

Extracting source infromation
This part should be replaced by the following procedures.

```
f0raw = MulticueF0v14(x,fs);
```

Then you have to calculate aperiodicity index using the fourth routine `exstraightAPind`.

```
 ap = exstraightAPind(x,fs,f0raw);
```

Extract spectral information
This part is the same as the default procedure.

```
n3sgram=exstraightspec(x,f0raw,fs);
```

Resynthesizing speech
This part is the same as the default procedure.

```
sy = exstraightsynth(f0raw,n3sgram,ap,fs);
```

Note:
Typing the name of the new F0 extractor shows how to use it in more detailed manner.

5.1 Comparison of the new F0 extractor and the defaul method

This section shows how the new F0 extractor fixes problems found in processing expressive speech using the default STRAIGHT F0 extractor. The following script performs STRAIGHT analysis and synthesis using both the default F0 extractor and the new F0 extractor.

```
[x,fs]=wavread('../Sample/em001c1013a.wav');
[f0raw,vuv,auxouts,prmouts]=MulticueF0v14(x,fs);
[ap,analysisParams]=exstraightAPind(x,fs,f0raw);
[n3sgram,prmP]=exstraightspec(x,f0raw.*vuv,fs);
[f0rawFixp,apFixp,analysisParams]=exstraightsource(x,fs);
[n3sgramFixp,prmP]=exstraightspec(x,f0rawFixp,fs);
[sy,prmS] = exstraightsynth(f0raw.*vuv,n3sgram,ap,fs);
[syFixp,prmS] = exstraightsynth(f0rawFixp,n3sgramFixp,apFixp,fs);
```

The original and synthesized sounds are linked below. (List of links only accessible in the HTML version of this document.)

The extracted F0 of the speech is shown in Figure 1. The red line shows the F0 trajectory extracted by the default extractor. The blue line shows the F0 trajectory extracted by the new F0 extractor. As you can see the new F0 extractor provides non-zero trajectory even in unvoiced or silent segments. The voiced/unvoiced distinction information is given in the binary value 'vuv'. The segments detected as voiced are represented as black boardered boxes in Figure 1.

The default F0 trajectory has several glitches in voiced portion. Those glitches are disappeared in the new F0 trajectory. Especially glitch around 800 ms produces noticeable defect in resynthesized speech by the default method. The following figure shows waveforms around 800 ms. From top to bottom, the default resynthesis, the new resynthesis and the original waveforms are displayed.

5.1.1 Default analysis parameter of the new F0 extractor (*)

The default values are highly tuned. Please use this default values.

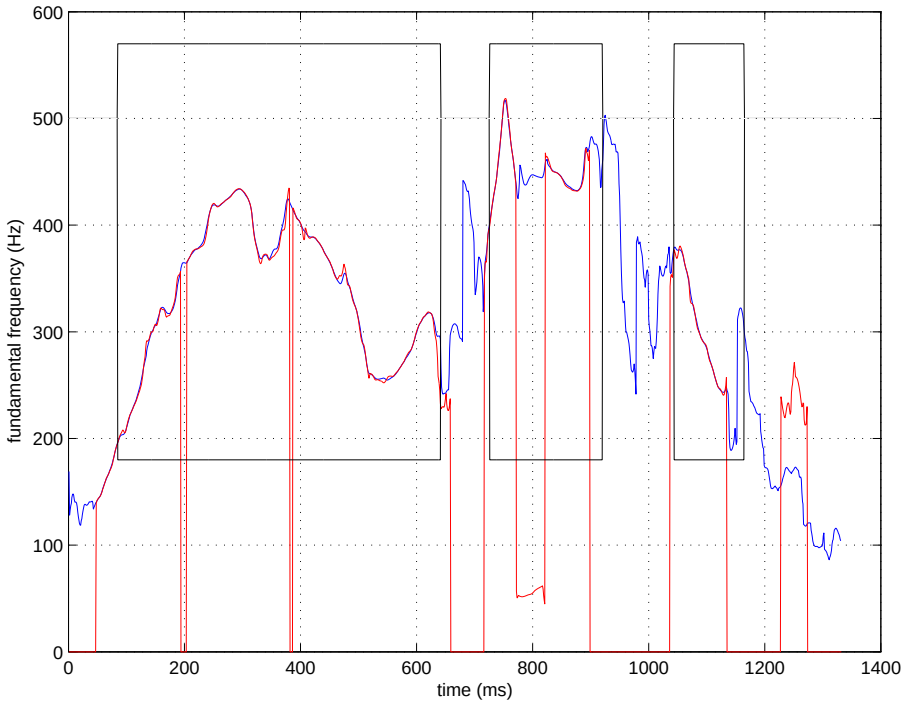

Figure 1: F0 trajectories extracted by the new and the default methods

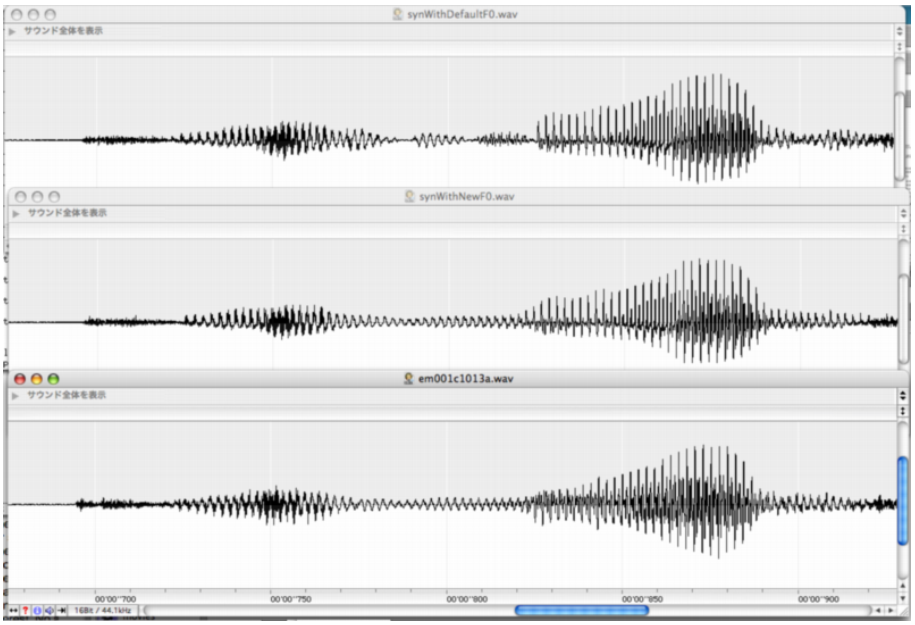

Figure 2: Resynthesized and the original waveforms. Top to bottom: default, new and the original.

```
                F0searchLowerBound: 40
                F0searchUpperBound: 800
           F0frameUpdateInterval: 1
             NofChannelsInOctave: 24
                 IFWindowStretch: 1.2000
                    DisplayPlots: 0
          IFsmoothingLengthRelToFc: 1
          IFminimumSmoothingLength: 5
        IFexponentForNonlinearSum: 0.5000
IFnumberOfHarmonicForInitialEstimate: 1
  TimeConstantForPowerCalculation: 10
              ACtimeWindowLength: 60
```

```
      ACnumberOfFrequencySegments: 8
     ACfrequencyDomainWindowWidth: 2200
   ACpowerExponentForNonlinearity: 0.5000
ACamplitudeCompensationInShortLag: 1.6000
          ACexponentForACdistance: 4
             AClagSmoothingLength: 1.0000e-04
        ACtemporalSmoothingLength: 20
              ThresholdForSilence: 3
                  ThresholdForVUV: 0.6000
      WeightForAutocorrelationMap: 1
      WeightForInstantaneousFqMap: 1
                VUVthresholdOfAC1: -0.1000
     SDforNormalizeMixingDistance: 0.3000
       SDforTrackingNormalization: 0.2000
     MaxumumPermissibleOctaveJump: 0.4000
           ThresholdToStartSearch: 0.3000
            ThresholdToQuitSearch: 0.3500
       ThresholdForReliableRegion: 0.2500
                          WhoAmI: 'MulticueF0v14'
```

6 Reading default analysis parameters

STRAIGHT has many preset parameters in its analysis and synthesis procedures. It is recommended to use preset default values, because they are optimized for general use. However, access and control procedures are provided to enable customization by the users to meet their special needs. This section introduces a way how to get preset values.

Assume that the working directory is set to the same directory in the previous section. Please input the following commands.

```
[x,fs]=wavread('vaiueo2d.wav');
[f0raw,ap,prmF0]=exstraightsource(x,fs);
[n3sgram,prmP]=exstraightspec(x,f0raw,fs);
[sy,prmS] = exstraightsynth(f0raw,n3sgram,ap,fs);
```

This does the same analysis and re-synthesis task. Important difference is that this set of commands provide readouts of preset parameters. They are stored in three structure variables (`prmF0, prmP, prmS`). The preset values stored in these structured variables can be reviewed by simply typing their variable names in the Matlab command window. The following sections introduces those preset values.

6.1 Default values in source information extraction

The following output is displayed by typing `prmF0` and return in the Matlab command window. These represent field names of a Matlab structured variable `prmF0` and their preset values for source information extraction. The field names are designed to be self-descriptive. (IF in field names stands for 'instantaneous frequency'.)

```
prmF0 =

                    F0searchLowerBound: 40
                    F0searchUpperBound: 800
                  F0defaultWindowLength: 40
                  F0frameUpdateInterval: 1
                    NofChannelsInOctave: 24
                        IFWindowStretch: 1.2000
                           DisplayPlots: 0
               IFsmoothingLengthRelToFc: 1
               IFminimumSmoothingLength: 5
             IFexponentForNonlinearSum: 0.5000
  IFnumberOfHarmonicForInitialEstimate: 1
                       refineFftLength: 1024
           refineTimeStretchingFactor: 1.1000
       refineNumberofHarmonicComponent: 3
        periodicityFrameUpdateInterval: 5
                                  note: ' '
```

6.2 Default values in spectral information extraction

The following output is displayed by typing `prmP` and return in the Matlab command window. These represent field names of a Matlab structured variable `prmP` and their preset values for STRAIGHT spectral analysis. The field names are designed to be self-descriptive.

```
prmP =

                     DisplayPlots: 0
                defaultFrameLength: 40
            spectralUpdateInterval: 1
         spectralTimeWindowStretch: 1.4000
  spectralExponentForNonlinearity: 0.6000
   spectralTimeDomainCompensation: 0.2000
```

6.3 Default values in synthesis

The following output is displayed by typing `prmS` and return in the Matlab command window. These represent field names of a Matlab structured variable `prmS` and their preset values for STRAIGHT synthesis The field names are designed to be self-descriptive.

```
prmS =

               spectralUpdateInterval: 1
          groupDelayStandardDeviation: 0.5000
           groupDelaySpatialBandWidth: 70
  groupDelayRandomizeCornerFrequency: 4000
            ratioToFundamentalPeriod: 0.2000
                  ratioModeIndicator: 0
          levelNormalizationIndicator: 1
                      headRoomToClip: 22
             powerCheckSegmentLength: 15
                timeAxisMappingTable: 1
    fundamentalFrequencyMappingTable: 1
           frequencyAxisMappingTable: 1
            timeAxisStretchingFactor: 1
```

7 Changing default values for analysis and synthesis

These preset values can be modified by assigning values to specific fields by specifying their field names. The following commands provide a examples how to do it for each STRAIGHT function. It is enough to set the revised value to a structured variable with the same field name that is to be modified.

```
[f0raw,ap,prmF0]=exstraightsource(x,fs prmF0in);
[n3sgram,prmP]=exstraightspec(x,f0raw,fs,prmPin);
[sy,prmS] = exstraightsynth(f0raw,n3sgram,ap,fs,prmS);
```

The following example uses the source information extraction function. Examples with the synthesis function are illustrated in the next section, because they are many enough to fill up numbers of sections.

The first thing is to display extracted source information and underlying information by setting 1 to the field `DisplayPlots`. The following commands make the source information extraction function to display the panel under the commands.

```
prminF0.DisplayPlots=1;
prminF0.note='viaueo2d.wav';
figure;[f0raw,ap]=exstraightsource(x,fs,prminF0);
```

The second line sets the read file name to the note field. It is a recommended practice.

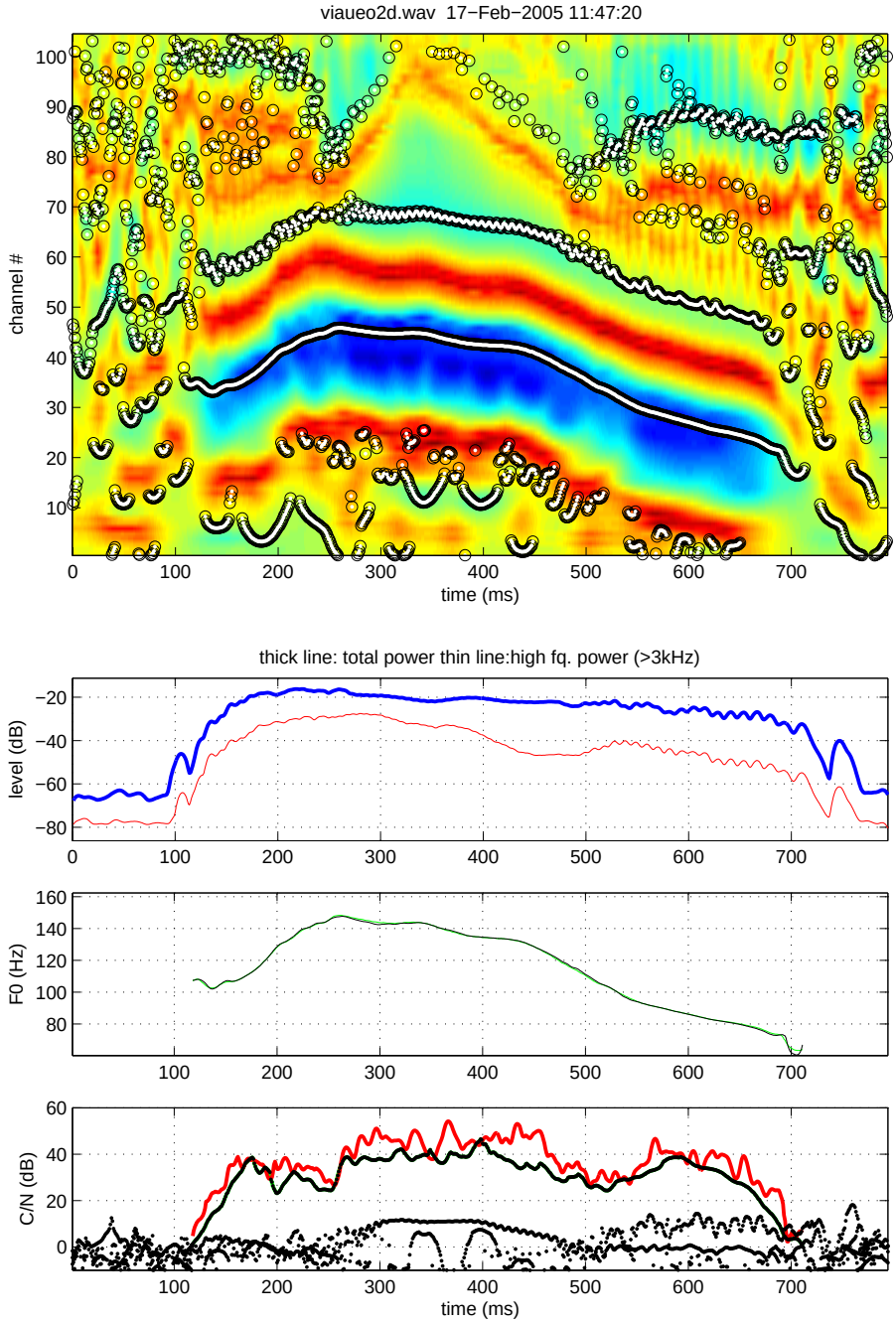

Figure 3: Displayed source information

7.1 Using other F0 extractors or re-doing analysis

Sometimes, F0 extraction fails. This failure introduces noticable degradation in resynthesized speech. It also introduces errors in spectrum estimation because of pitch synchronous nature of STRAIGHT spectral analysis. To alleviate this problem, you may manually edit the F0 information or you may use the other F0 ectractors. Once a fixed F0 information is ready and is stored in a variable, for example in 'f0fix', you can calculate the aperiodicity index using the following function.

```
ap=exstraightAPind(x,fs,f0fix);
```

8 Controlling synthesis parameters

Synthesis parameters are the most frequently modified parameters. The following sections illustrate examples of usage and effects.

8.1 Group delay parameters (*)

One unique (at least in 1996) feature of STRAIGHT is the group delay manipulated excitation source signal. The signal is controlled by the three parameters, `groupDelayStandardDeviation, groupDelayRandomizeCornerFrequency` and `groupDelaySpatialBandWidth`. The following three subsections illustrates effect of controlling each parameter.

Please keep the flag variable `ratioModeIndicator` unchanged, because the other group delay parameter `ratioToFundamen` was found to be problematic and is usually not in use.

8.1.1 Parameter: groupDelayStandardDeviation (*)

The default value is 0.5 ms. This determines the amount of the group delay dispersion in terms of its standard deviation on the frequency axis. The effects of manipulating this parameter is shown in the following plots.

The first example shows the excitation source signal when this parameter is set to 0. The source and the spectral information (`f0raw,n3sgram,ap`) are extracted from the vowel sequence used in the previous sections (`vaiueo2d.wav`). (A small trick is used here to observe the excitation source signal. All elements of the matrix variable `n3sgram` are set to a positive constant value. All elements of the matrix variable `ap` are set to a small (-60 for example) constant value.)

The excitation source signal for 0 ms group delay dispersion is calculated by calling synthesis function with the field `groupDelayStandardDeviation` to be set to a positive small value. The other parameters are set default.

```
prminS.groupDelayStandardDeviation=0.000001;
sy0 = exstraightsynth(f0raw,n3sgram*0+100,ap*0-80,fs,prminS);
```

The excitation source signal and the spectrogram are displayed by using the following commands. Please note that the default frame update interval is 1ms. The `specgram` function of Matlab system is used to calculate the spectrogram.

```
figure;
plot((1:length(syZ))/fs*1000,sy0);
axis([650 700 -30000 45000]);
set(gca,'fontsize',16);
xlabel('time (ms)');title('delsp=0, cornf=4000');
figure;
nsgnnZ=specgram(syZ,512,fs,90,85);
[nr nc]=size(nsgnnZ);
imagesc([0 nc*5000/fs],[0 fs/2],max(45,dB(abs(nsgnnZ))));axis('xy')
axis([650 700 0 fs/2])
xlabel('time (ms)');ylabel('frequency (Hz)'); title('delsp=0, cornf=4000');
```

The following plot shows the excitation source waveform around the area where the fundamental frequency goes down to 75Hz. Small blobs close to excitation pulses are side effects of the linear group delay manipulation for implementing sub-sampling accuracy in pulse interval control. The cause of the side effects is intuitively understood by the nest spectrographic display of the same signal.

The following plot is a spectrographic display of the same signal. The sampling frequency of the signal is 22,050Hz, in this case. Please note that group delay represents temporal energy centroid at each frequency. The pseudo color representation shows high energy area using red to brown and shows low energy area using deep blue. Roughly speaking, the red to brown area shows the shape of group delay function. Taking into account of this interpretation, it is easy to see that the spectrogram illustrates that the blobs are due to bending of group delay close to the Nyquist frequency.

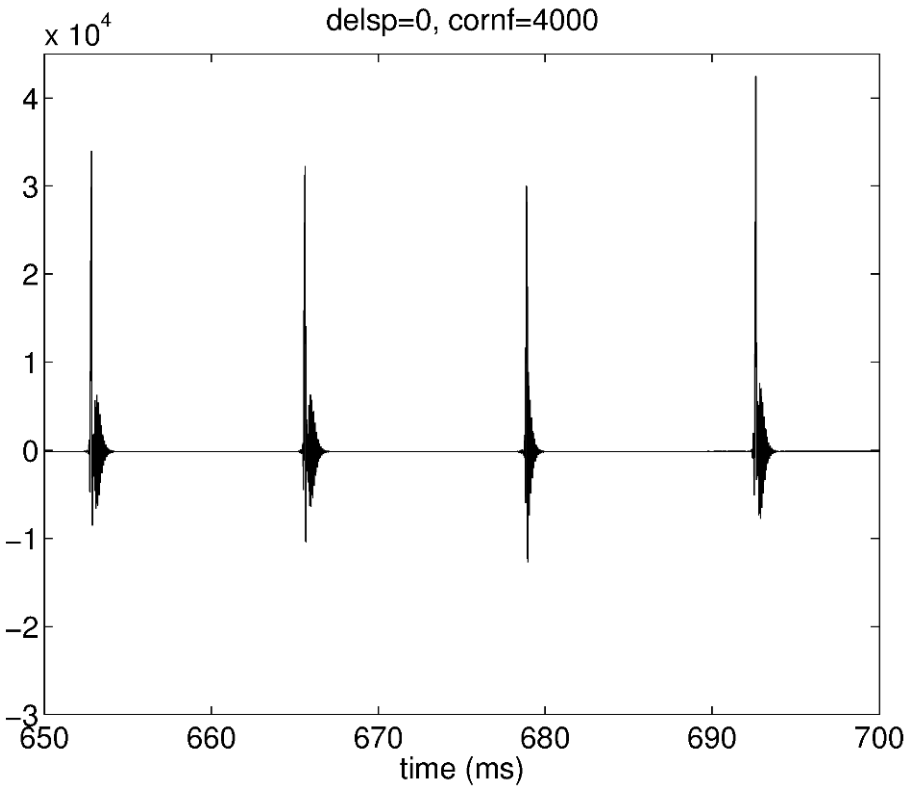

Figure 4: Waveform of the excitation source signal with 0 ms group delay dispersion.

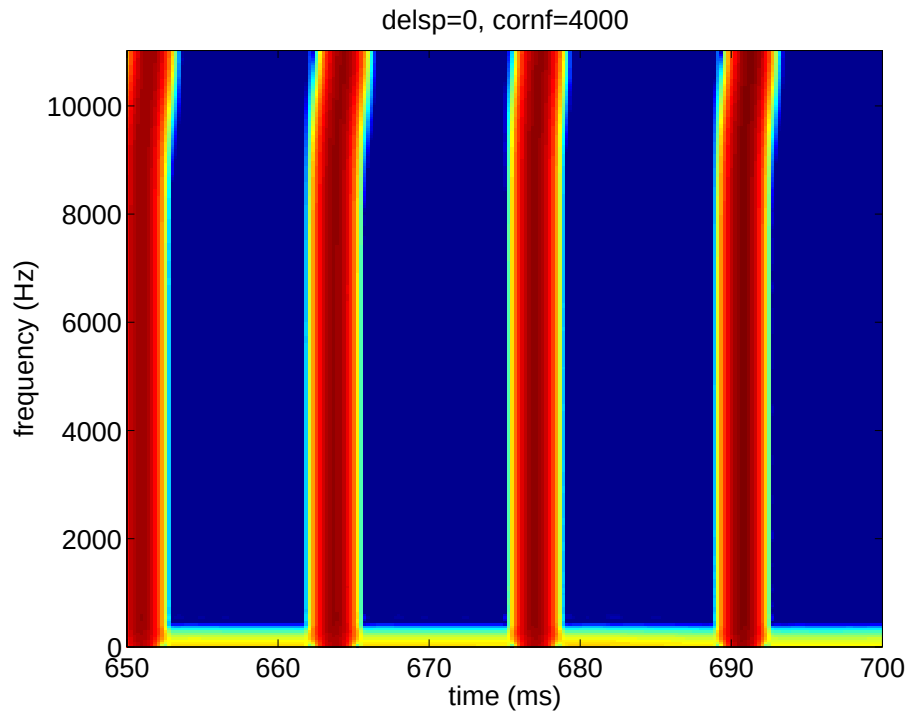

Figure 5: Spectrogram of the excitation source signal with 0 ms group delay dispersion.

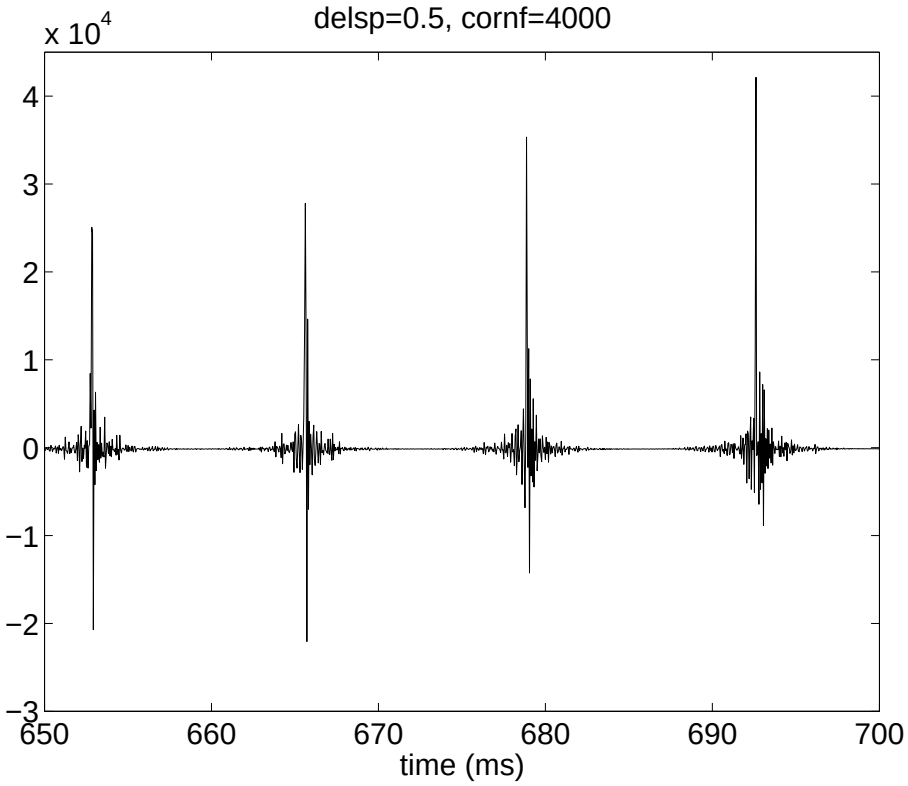

Figure 6: Waveform of the excitation source signal with 0.5 ms group delay dispersion.

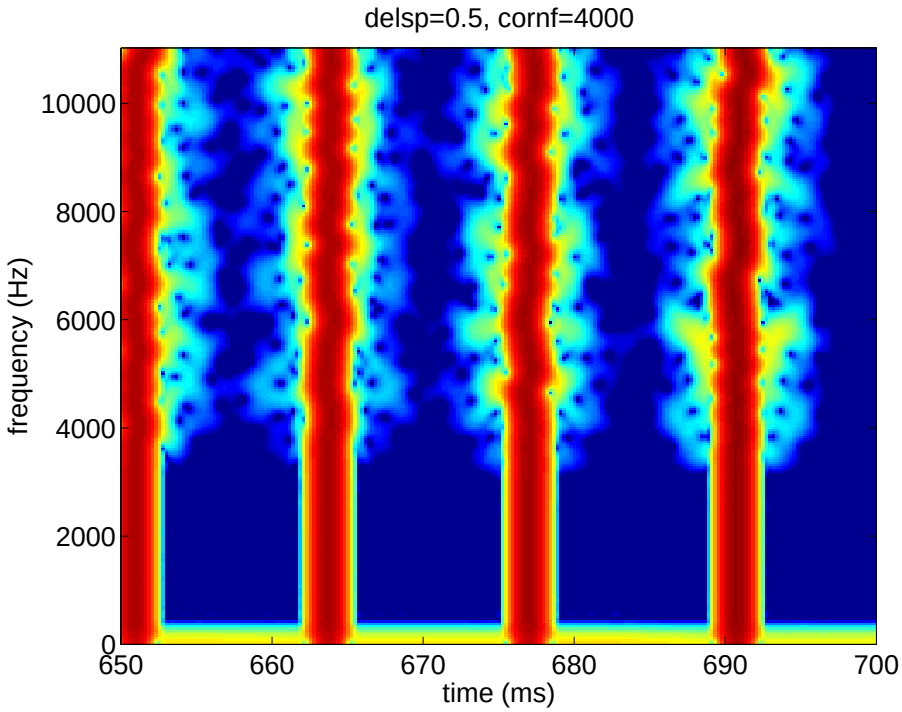

Figure 7: Spectrogram of the excitation source signal with 0.5 ms group delay dispersion.

groupDelayStandardDeviation=0.5 ms (default)

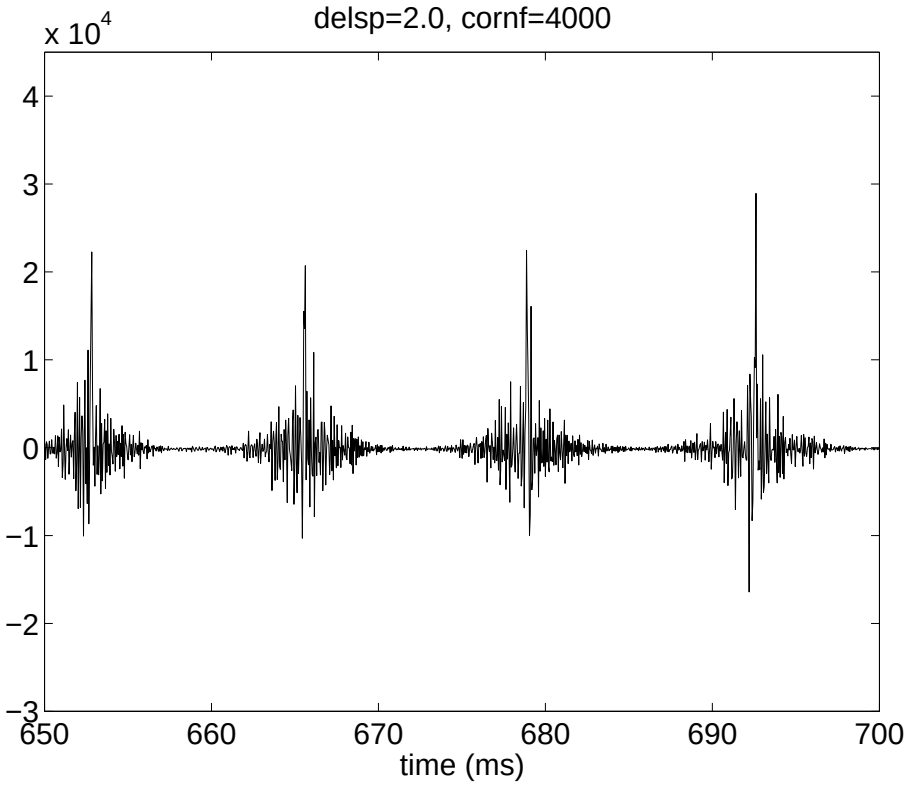

Figure 8: Waveform of the excitation source signal with 2 ms group delay dispersion.

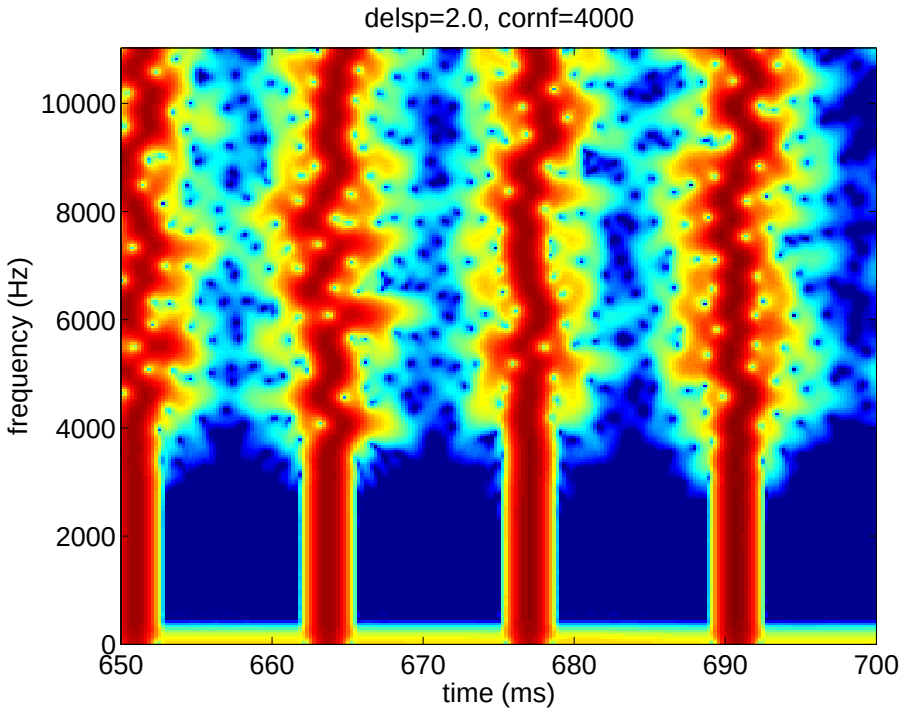

Figure 9: Spectrogram of the excitation source signal with 2 ms group delay dispersion.

groupDelayStandardDeviation=2 ms

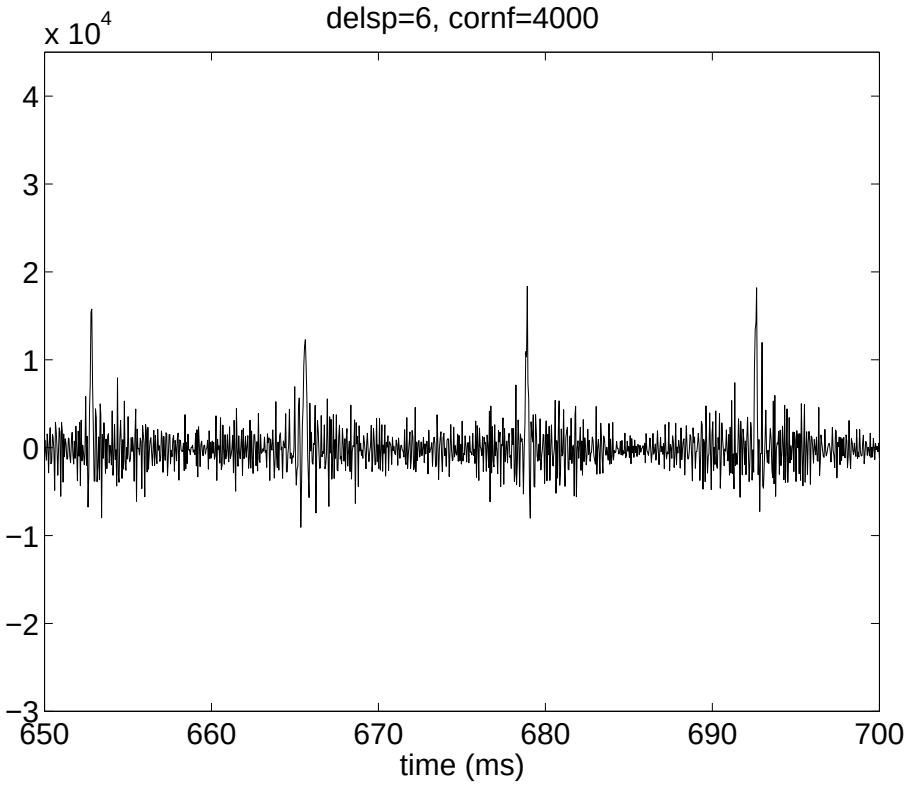

Figure 10: Waveform of the excitation source signal with 6 ms group delay dispersion.

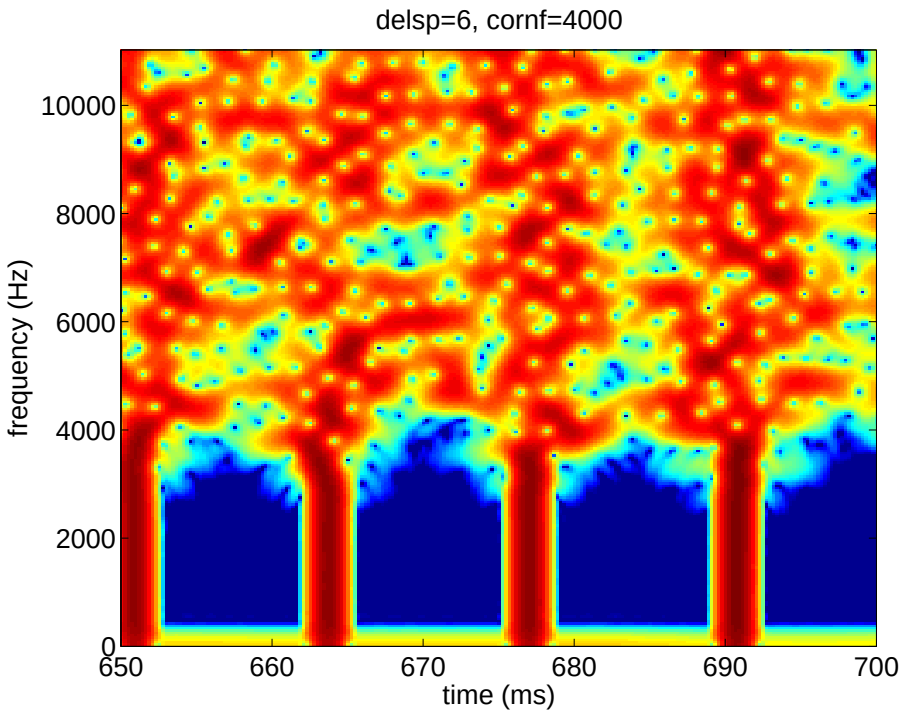

Figure 11: Spectrogram of the excitation source signal with 6 ms group delay dispersion.

groupDelayStandardDeviation=6 ms

8.1.2 Parameter: groupDelayRandomizeCornerFrequency (*)

The default value of this parameter is 4000Hz. This parameter sets the boundary frequency between the randomized group delay region and the constant group delay region.

The same speech example is used here and the same conditions are assumed. The following commands modifies the boundary frequency value to 8000Hz by setting the field `groupDelayRandomizeCornerFrequency`. Please note that the 2 ms group delay dispersing is used here (instead of using its default 0.5 ms) to enhance visual salience of the effect of this parameter.

```
prminS.groupDelayStandardDeviation=2;
prminS.groupDelayRandomizeCornerFrequency=8000;
sy8000 = exstraightsynth(f0raw,n3sgram*0+100,ap*0-80,fs,prminS);
```

The effect is not clearly visible in the waveform representation.

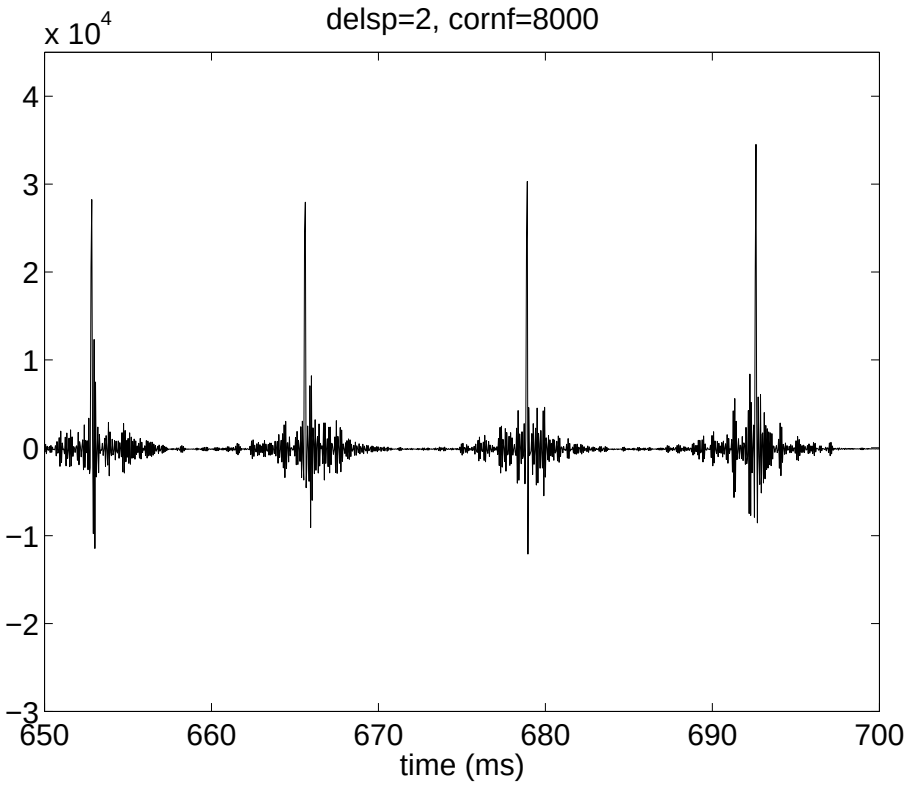

Figure 12: Waveform of the excitation source signal with the 8000Hz boundary frequency.

The effect is salient in the spectrogram. The boundary between straight line and the randomized pattern is clearly moved up to 8000Hz.

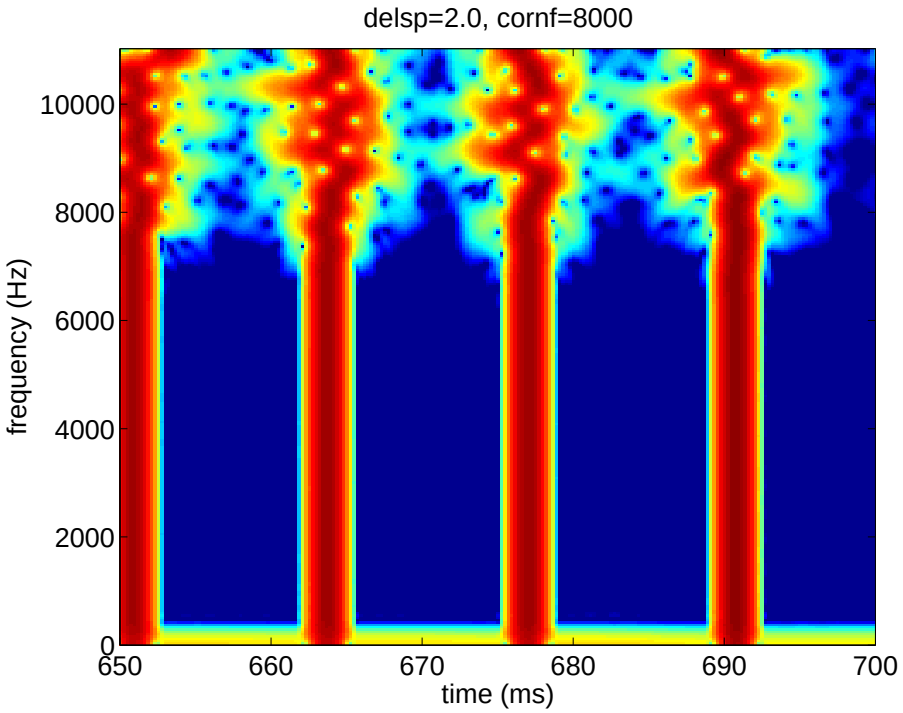

Figure 13: Spectrogram of the excitation source signal with the 8000Hz boundary frequency.

8.1.3 Parameter: groupDelaySpatialBandWidth (*)

The default value of this parameter is 70 Hz. This parameter defines the coarseness of the group delay trajectory along the frequency axis. The uncertainty relation between temporal resolution and frequency resolution introduces constraint, that makes energy to spread temporally when this parameter decreases. This parameter is used to control this temporal spread as (a kind of) side effect. This control is implemented as a moving averaging on the frequency axis. The following examples show effects when this parameter is one tenth and ten times of the default value. The group delay dispersion is also set 2ms here to illustrate effects more salient.

```
prminS.groupDelayStandardDeviation=2;
prminS.groupDelaySpatialBandWidth=700;
sy700 = exstraightsynth(f0raw,n3sgram*0+100,ap*0-80,fs,prminS);
```

The effects are not salient. The temporal spread is somewhat small.

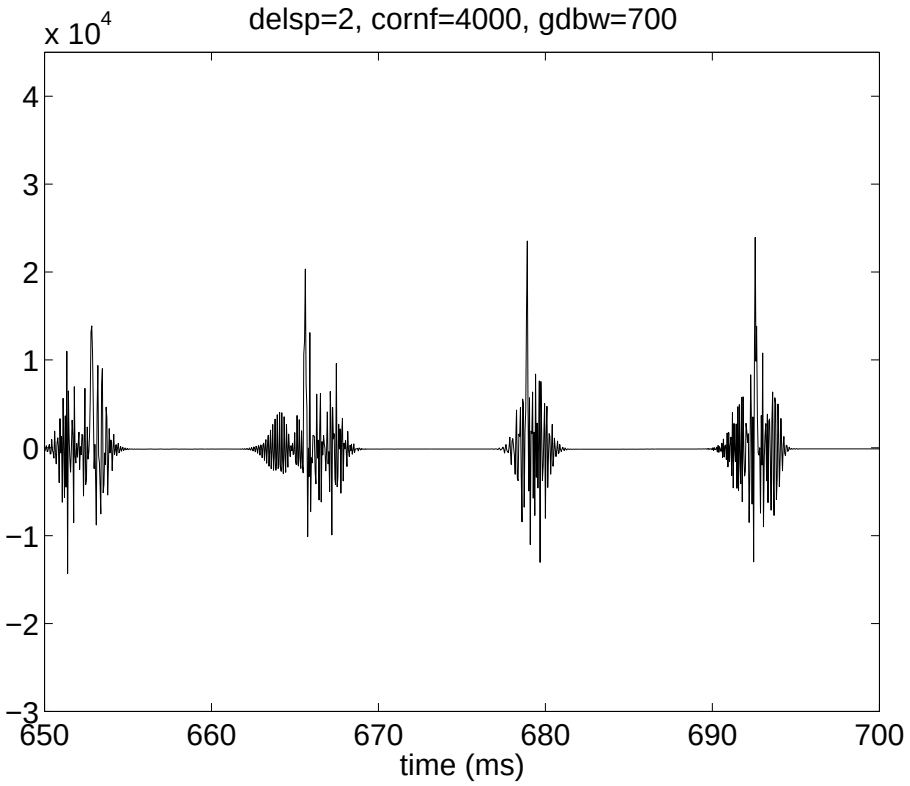

Figure 14: Waveform of the excitation source signal when the spatial frequency bandwidth is 700Hz.

The spectrographic display looks clearly different from the default case. A cloud like time-frequency spread of energy found in default case does not exist in this representation. The energy trajectory along the frequency axis looks smoother.

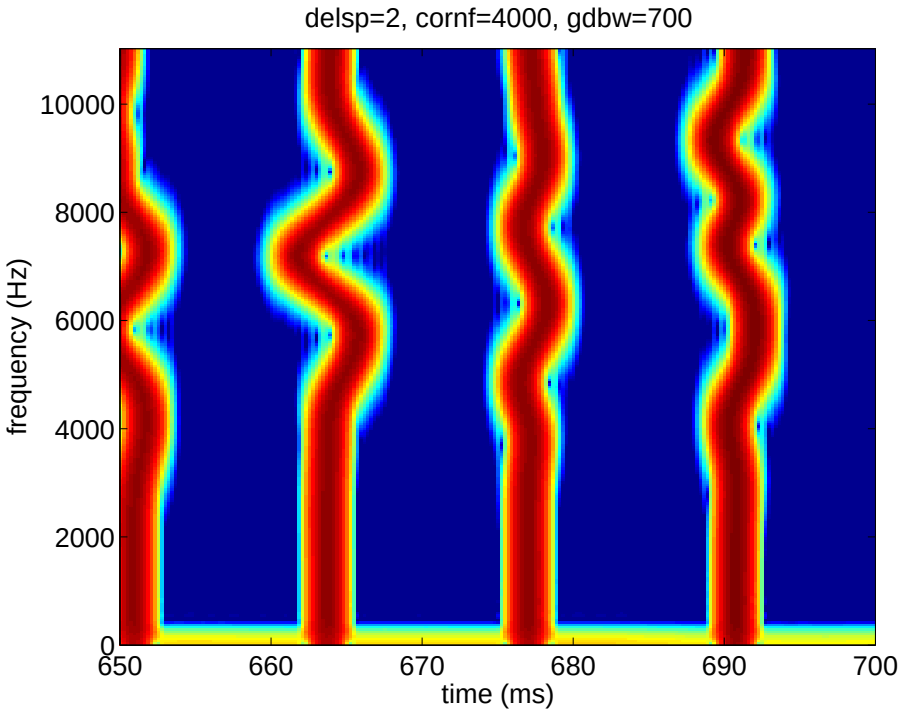

Figure 15: Spectrogram of the excitation source signal when the spatial frequency bandwidth is 700Hz.

groupDelaySpatialBandWidth = 7Hz The temporal spread is visible when the bandwidth is set 7 Hz, even though the group delay standard deviation parameter is kept same.

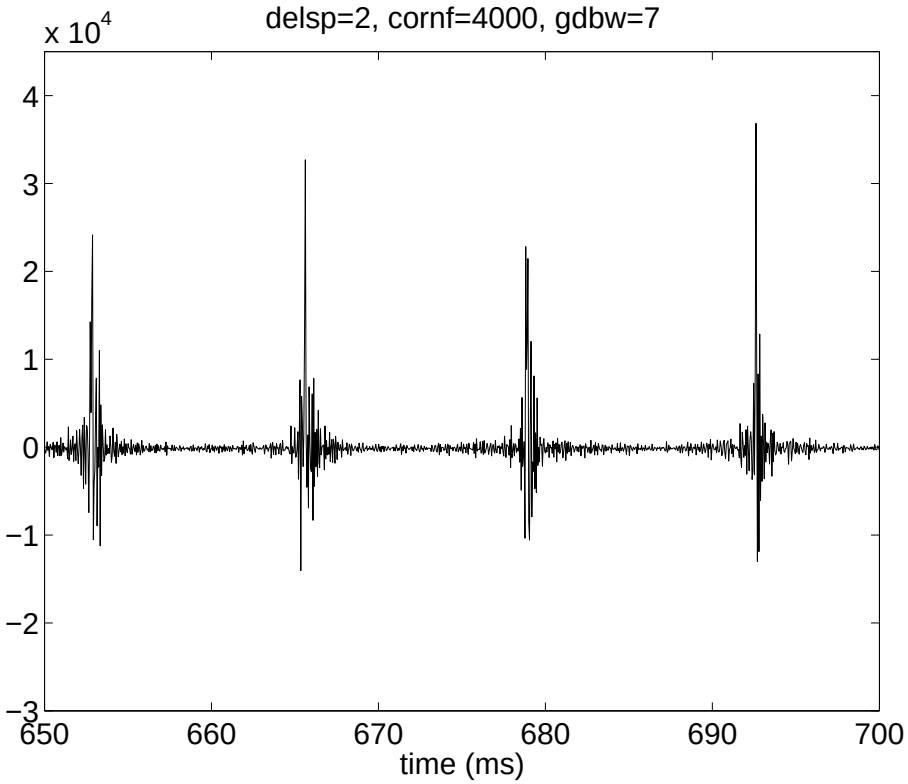

Figure 16: Waveform of the excitation source signal when the spatial frequency bandwidth is 7Hz.

The cloud like time-frequency spread is denser than default. The energy trajectory along the frequency axis does not seem to have smaller deviations.

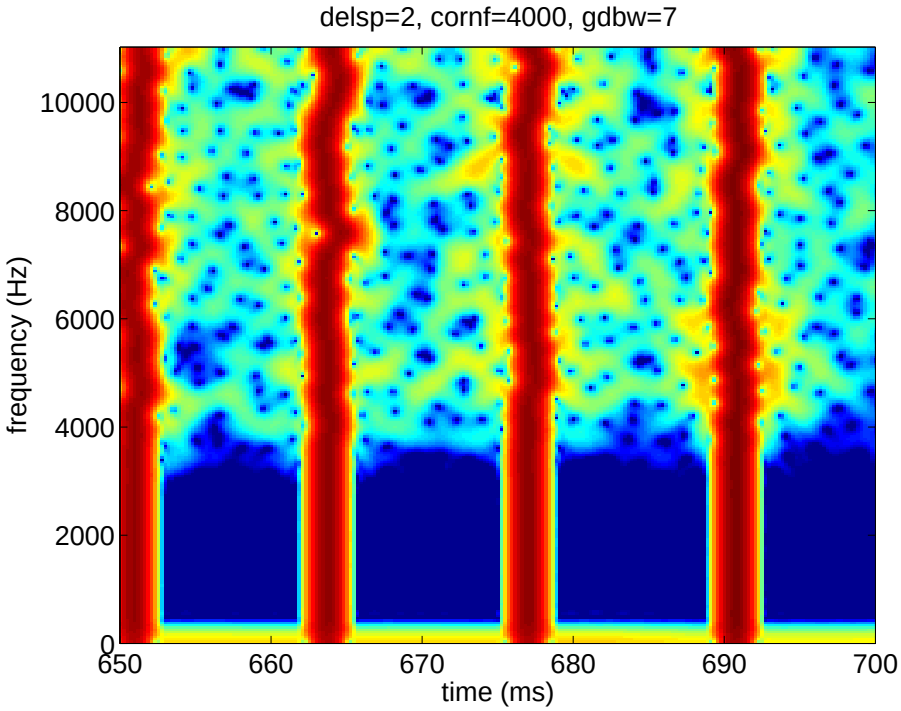

Figure 17: Spectrogram of the excitation source signal when the spatial frequency bandwidth is 7Hz.

8.2 How to assign the lower F0 limit for synthetic speech (*)

Acceptable lowest fundamental frequency in re-synthesis stage is introduced by the limited data length introduced by the FFT buffer size. It is a function of the FFT buffer length in samples and the sampling frequency. Current implementation sets the buffer length two times longer than the fundamental period of this lower F0 limit. (The default value is 40 Hz.) Users using recent versions (STRAIGHTv40_005b and later) can set this lower limit F0 explicitly.[1]

The following example shows how to assign the lower F0 limit to 20 Hz. It uses the optional structured variable for controlling synthesis conditions.

```
prm.lowestF0 = 20;
[sy,prmS] = exstraightsynth(f0raw,n3sgram,ap,fs,prm);
```

Explanation how synthesis stage responded to the assigned conditions can be retrieved by assigning a optional structured variable as one of outputs of the synthesis routine. The return values and their explanations are the following.

```
prmS.statusReport
```

ok Synthesis is completed successfully.

Minimum synthesized F0 exceeded the lower limit Synthesis is terminated without producing result.

The FFT length was inconsistent and replaced Synthesis is completed using the updated FFT buffer length. The length is calculated based on the newly assigned lowestF0 parameter.

Frequency axix mapping function is not consistent with lowestF0 Synthesis is completed using the updated FFT buffer length. The length is calculated based on the newly assigned lowestF0 parameter. The inconsistent frequency mapping table assigned is ignored.

[1] No API for F0 lower limit was available in the former versions. Moreover, they cause accidental termination of synthesis stage, when F0 value was lowered beyond the implicit limit (40Hz).

9 How to run your batch programs (UNIX and OS X) (*)

The three wrapper programs were developed to make programming easier by providing structured APIs. It is also suitable for background batch processing. Mac OS X is a variant of UNIX meaning the same procedure is applicable to run a batch job. Please refer to the support page of Mathworks and reference manual of Matlab. This section introduces one simple example.

The first thing is to clear the environmental variable DISPLAY.

```
unsetenv DISPLAY
```

The next step is to change the current directory to the directory where the Matlab script to run. Then, please issue the following command. Assume `test.m` is the program to be executed.

```
nohup matlab <  test.m >! logoftest.txt &
```

The redirection >! overwrites Matlab outputs to the text file `logoftest.txt`. The status of the background job can be monitored by using `jobs` or `ps` or `top`. The above procedure initiate a batch job that does not terminates when the user logouts.

References

[1] Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain de Cheveigné. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction. *Speech Communication*, Vol. 27, No. 3-4, pp. 187–207, 1999.

[2] Hideki Kawahara, STRAIGHT, exploitation of the other aspect of VOCODER: Perceptually isomorphic decomposition of speech sounds *Acoustical Science and Technology*, Vol. 27, No. 6, pp. 349-353, 2006.

[3] Hideki Kawahara and Hisami Matsui. Auditory morphing based on an elastic perceptual distance metric in an interference-free time-frequency representation. In *Proc. 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, Vol. I, pp. 256–259, Hong Kong, 2003.

[4] Hisami Matsui and Hideki Kawahara. Investigation of emotionally morphed speech perception and its structure using a high quality speech manipulation system. In *Eurospeech'03*, pp. 2113–2116, Geneva, 2003.

[5] Hideki Kawahara, Alain de Cheveigné, Hideki Banno, Toru Takahashi and Toshio Irino. Nearly Defect-free F0 Trajectory Extraction for Expressive Speech Modifications based on STRAIGHT. In *Interspeech'05*, pp. 537–540, Lisboa, 2005.

A GUI for STRAIGHT

The GUI for STRAIGHT that was introduced 1998 is still functional in this version. However, this old GUI will not be refined actively. It will be revised mainly for maintenance of compatibility and bug fixes. Please refer the following pages for introduction to the GUI.

- http://www.wakayama-u.ac.jp/~kawahara/puzzlet/STRAIGHTtipse/sourceinformation.html
- http://www.wakayama-u.ac.jp/~kawahara/puzzlet/STRAIGHTtipse/betterresynth.html
- http://www.wakayama-u.ac.jp/~kawahara/puzzlet/STRAIGHTtipse/F0modification.html
- http://www.wakayama-u.ac.jp/~kawahara/puzzlet/STRAIGHTtipse/timempl.html

Please note that the GUI is very old. The programming style and practice are terrible. Please use this GUI only for demonstrations and some preliminary tests. Please use command line mode and program your own project using new interface functions. Examples for parameter manipulation described in the links above are outdated and unnecessarily complex. Please consider using more flexible and powerful procedures such as http://www.wakayama-u.ac.jp/~kawahara/puzzlet/morphingWithSTRAIGHT/morphingWithSTRAIGHTe.html.

B Notes on STRAIGHT parameters

This appendix provides additional descriptions on speech parameters extracted using STRAIGHT.

B.1 Aperiodicity Index

STRAIGHT has two modes of controlling excitation source; group delay manipulation and mixed mode excitation. The aperiodicity index controls the noise to total energy ratio using pulse plus noise excitation source. The following schematic diagram shows steps to calculate the aperiodicity index.

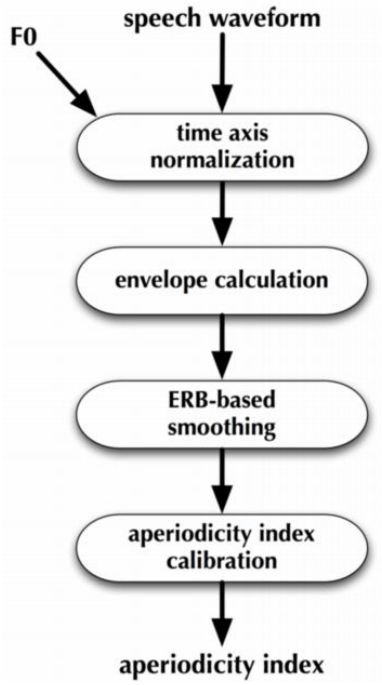

Figure 18: How to estimate aperiodicity index. part1.

The following diagram illustrates how the envelope extraction for estimating the raw estimate of the aperiodicity index. The value of the index (ap) is conceptually a dB difference between lower envelope (Le) and upper envelope (Ue). (ap = Le -Ue)

This raw estimate is then smoothed on the frequency axis based on resolution about 1 ERB (Equivalent Rectangular Band width. Please consult with text book on psychology of hearing.) The following Matlab code defines the relation between frequency (x) and ERB (y).

```
y = 21.4*log10(4.37e-3*x+1);
```

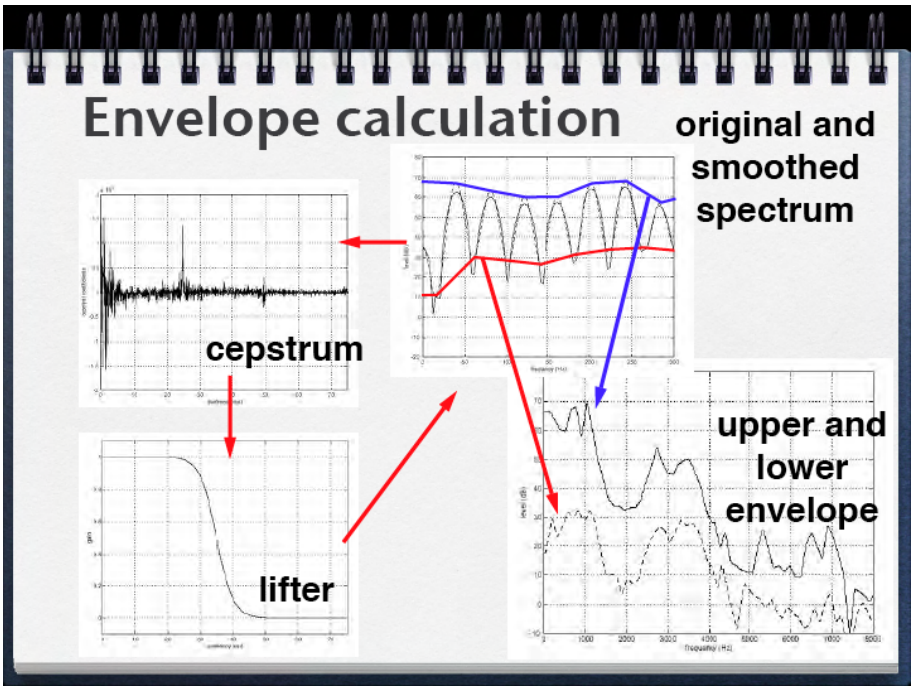

Figure 19: How to estimate aperiodicity index..

Finally the ap values are calibrated based on the calibration table that was prepared based on a series of simulations using synthetic vowels.

```
imagesc([0 794],[0 fs/2],10.0.^(ap/20));axis('xy')
```

The following figure is the extracted aperiodicity index using the formula above. The speech is a Japanese vowel sequence /aiueo/. This index is represented in dB. The value 0 dB indicates that the excitation is totally random because it represents contribution from energy of random component. When the index is set lower than -60 dB, the excitation is effectively purely periodic.

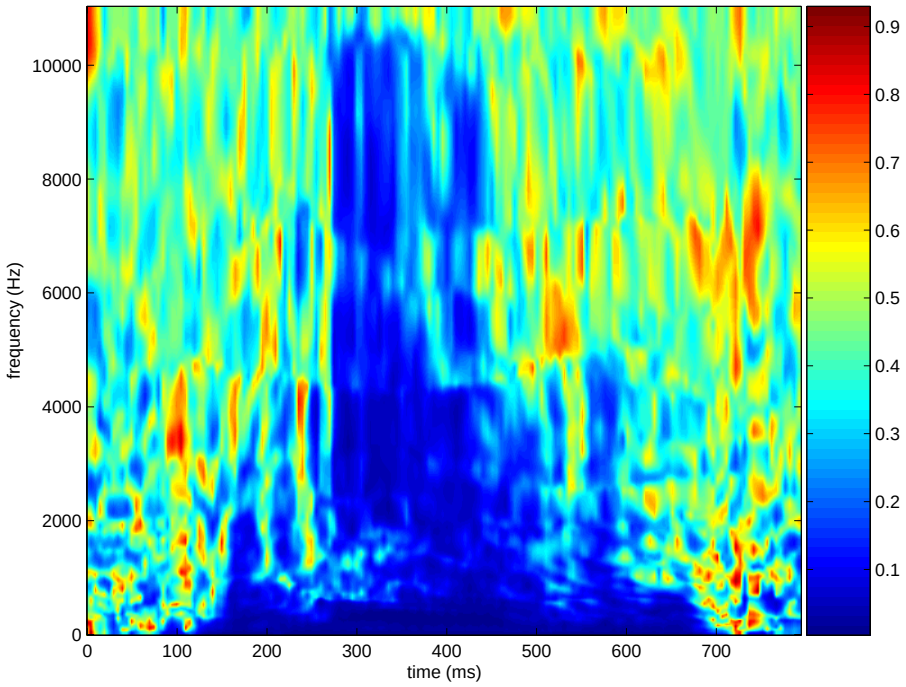

Figure 20: Extracted aperiodicity index using the formula above. The speech is a Japanese vowel sequence /aiueo/.

The following Matlab command line session record shows how to modify the apriodicity index to control excitation source.

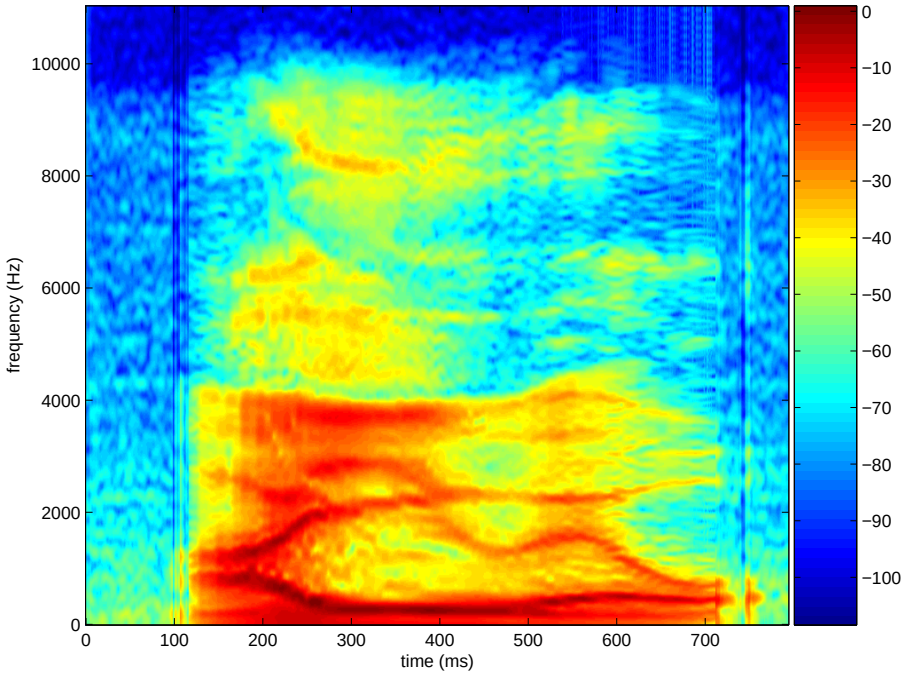

Figure 21: STRAIGHT spectrogram for the same sample. The speech is a Japanese vowel sequence /aiueo/.

```
>> [x,fs,nbs,ops]=wavread('vaiueo2d.wav');
>> [f0raw,ap]=exstraightsource(x,fs);
>> n3sgram=exstraightspec(x(:,1),f0raw,fs);
>> syOrg = exstraightsynth(f0raw,n3sgram,ap,fs);
>> syApr = exstraightsynth(f0raw,n3sgram,ap*0,fs);
>> syPpr = exstraightsynth(f0raw,n3sgram,ap*0-60,fs);
>> [syPprD0,prmS] = exstraightsynth(f0raw,n3sgram,ap*0-60,fs);
>> prmS

prmS =

               spectralUpdateInterval: 1
          groupDelayStandardDeviation: 0.5000
           groupDelaySpatialBandWidth: 70
  groupDelayRandomizeCornerFrequency: 4000
            ratioToFundamentalPeriod: 0.2000
                  ratioModeIndicator: 0
          levelNormalizationIndicator: 1
                      headRoomToClip: 22
             powerCheckSegmentLength: 15
                timeAxisMappingTable: 1
    fundamentalFrequencyMappingTable: 1
           frequencyAxisMappingTable: 1
            timeAxisStretchingFactor: 1
                        DisplayPlots: 0
                            lowestF0: 50
                        statusReport: 'ok'

>> prmS.groupDelayStandardDeviation = 0.001;
>> [syPprD0,prmS] = exstraightsynth(f0raw,n3sgram,ap*0-60,fs,prmS);
>> wavwrite(syOrg/32768,fs,16,'synAiueoOrg.wav');
>> wavwrite(syApr/32768,fs,16,'synAiueoApr.wav');
>> wavwrite(syPpr/32768,fs,16,'synAiueoPpr.wav');
>> wavwrite(syPprD0/32768,fs,16,'synAiueoPprD0.wav');
```

Examples synthesized using these commands are linked below. (Links are accessible only in the HTML version of this document.)

Index