From Fourier Series to Analysis of Non-stationary Signals – VII

Miroslav Vlček, Jan Přikryl

November 18, 2019

Department of Applied Mathematics, CTU FTS

Contents

Short Time Fourier Transform

An overview of Fourier transforms

	continuous in time	discrete in time periodic in frequenc
continuous in frequency	$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\omega) e^{j\omega t} d\omega$	$x(n) = \frac{T}{2\pi} \int_{-\pi/T}^{+\pi/T} X(e^{j\omega T}) e$
	$X(j\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt$	$X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x(n) e^{-}$
	Fourier transform	Fourier transform $t =$
y		

What is wrong with Fourier Transform?

Let's consider basis functions: $\sin(\omega t)$, $\cos(\omega t)$ and $\delta(t)$

support region	in time	in frequency
$\sin(\omega t)$	$(-\infty, \infty)$	0
$\cos(\omega t)$	$(-\infty, \infty)$	0
$\delta(t)$	0	$(-\infty, \infty)$

- The basis functions $\sin(\omega t)$ and $\cos(\omega t)$ are not localized in time!
- The $\delta(t)$ is not localized in frequency!

We have learned to localize a signal in time domain by windowing
$\Rightarrow$ short time Fourier transform (STFT)

Short Time Fourier Transform

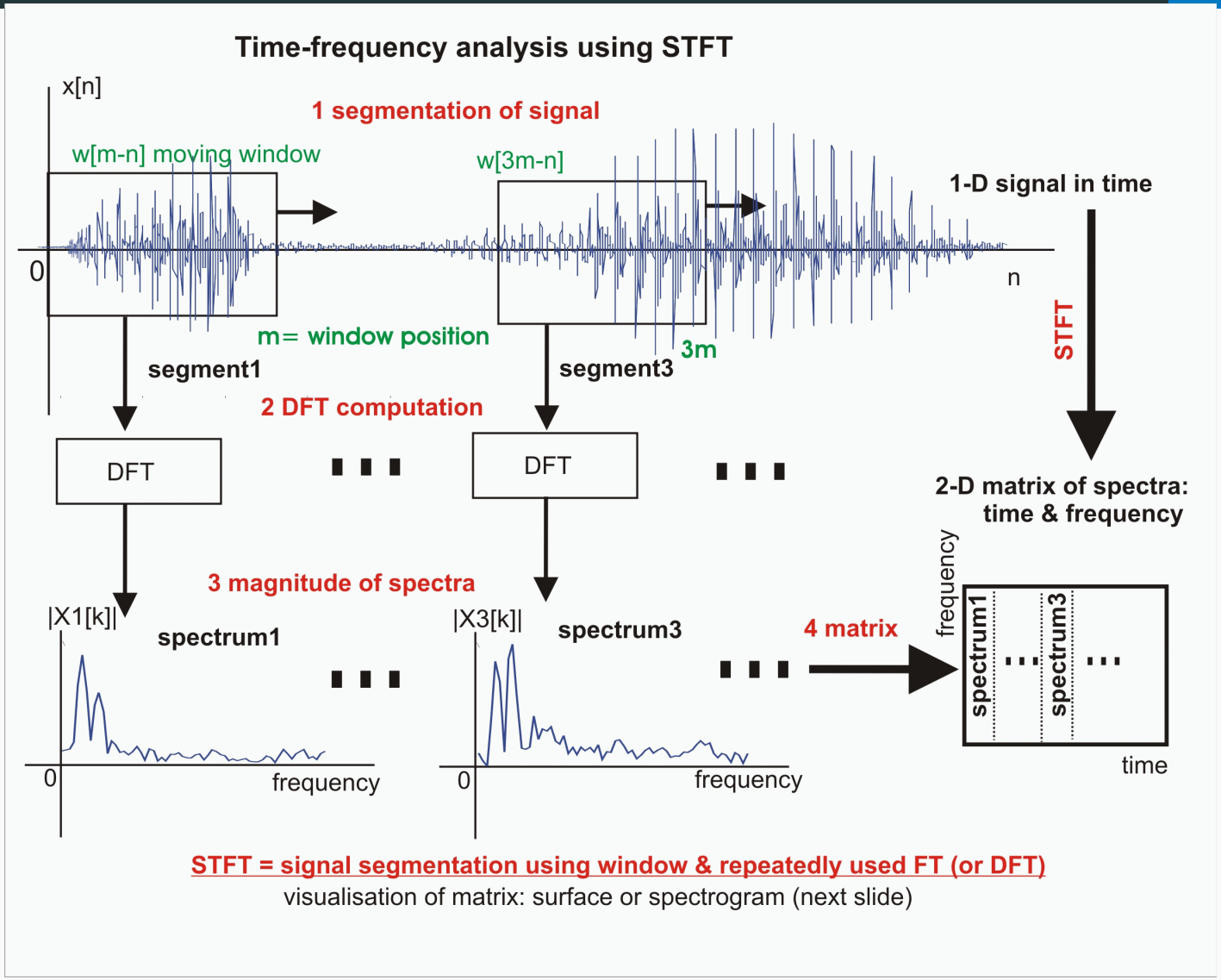
- When sampling an audio signal at a sampling rate 44.1 kHz
- ... 1 hour of stereophonic music would be $44100 \times 2 \times 60 \times 60 = 317520000$ samples !
- If we want to compute DFT, the closest power-of-two FFT is $2^{28} = 268435456$ per channel.
- A better approach is to break the long signal into small segments and analyze each one with FFT $\Rightarrow$ short time Fourier transform

Short Time Fourier Transform

- Assume that $x[m]$ is an infinitely long sequence
- in order to localize energy in time and frequency we segment the signal into short-time pieces and calculate DFT of each one
- sampled STFT for a window defined in the region $0 \leq m \leq M-1$ is given by

$$X[k, \ell L] = \sum_{m=0}^{M-1} x[\ell L - m] w[m] \exp\left(-\mathrm{j}2\pi \frac{km}{N}\right)$$

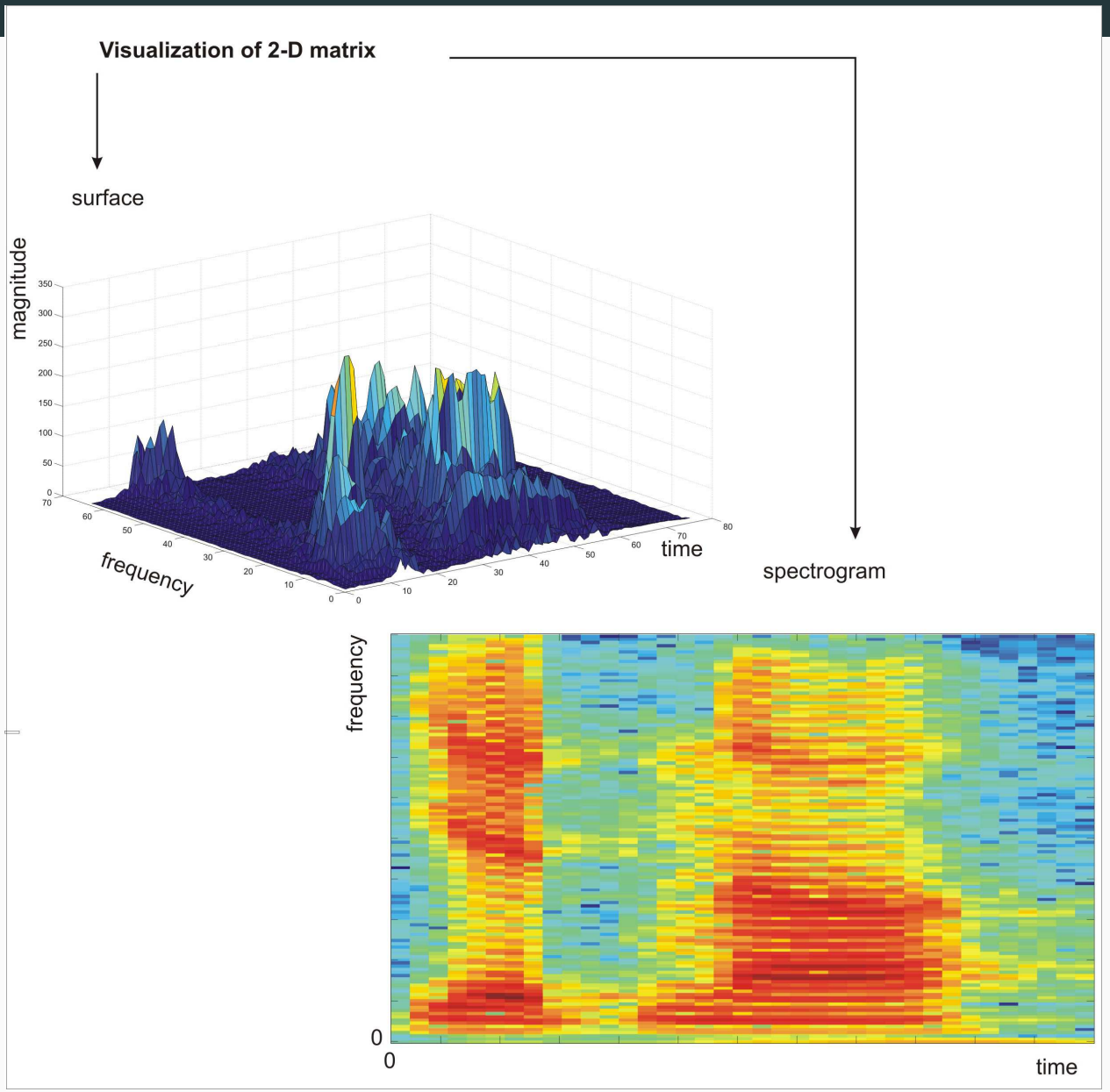
Short Time Fourier Transform



STFT = signal segmentation using window & repeatedly used FT (or DFT)

visualisation of matrix: surface or spectrogram (next slide)

Short Time Fourier Transform



Spectrograms

Spectrograms

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `x` is the signal specified by vector **x**.
- if `window` is an integer, `x` is divided into segments of length equal to that integer value
- otherwise, `window` is a Hamming window of length `nfft`
- `noverlap` is the number of samples each segment of `x` overlaps

Spectrograms

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `x` is the signal specified by vector **x**.
- if `window` is an integer, `x` is divided into segments of length equal to that integer value
- otherwise, `window` is a Hamming window of length `nfft`
- `noverlap` is the number of samples each segment of `x` overlaps

Spectrograms

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `x` is the signal specified by vector **x**.
- if `window` is an integer, `x` is divided into segments of length equal to that integer value
- otherwise, `window` is a Hamming window of length `nfft`
- `noverlap` is the number of samples each segment of `x` overlaps

Spectrograms

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

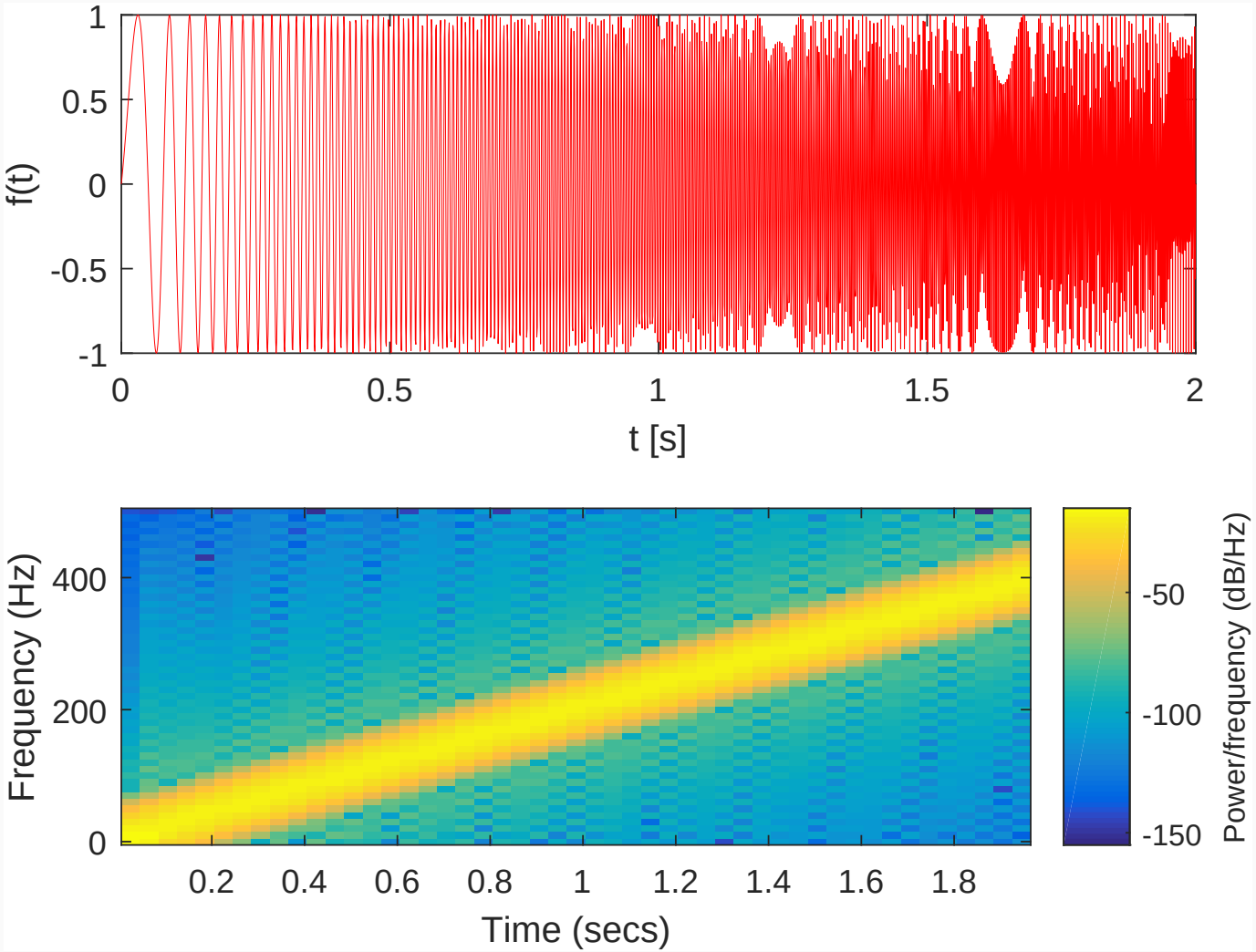
- `x` is the signal specified by vector **x**.
- if `window` is an integer, `x` is divided into segments of length equal to that integer value
- otherwise, `window` is a Hamming window of length `nfft`
- `noverlap` is the number of samples each segment of `x` overlaps

Spectrograms

- `nfft` is the FFT length and is the maximum of 256 or the next power of 2 greater than the length of each segment of `x`
- `fs` is the sampling frequency, which defaults to normalized frequency
- using `'yaxis'` displays frequency on the y-axis and time on the x-axis

We also use command `colorbar` which appends a color scale to the current axes.

Chirp signal analysis $\sin(2\pi(f_0 + \alpha t)t)$



Matlab project

MATLAB project – Spectrogram I

a) Start MATLAB. Load in the audio signal with commands

```
filename = 'flute-C4.wav';
[x1 sr1] = audioread(filename);
```

b) The sampling rate is 11 025 Hz, and the signal contains 36 250 samples.

Q: If we consider this signal as sampled on an interval $[0, T)$, what is the time duration of the flute sound ?

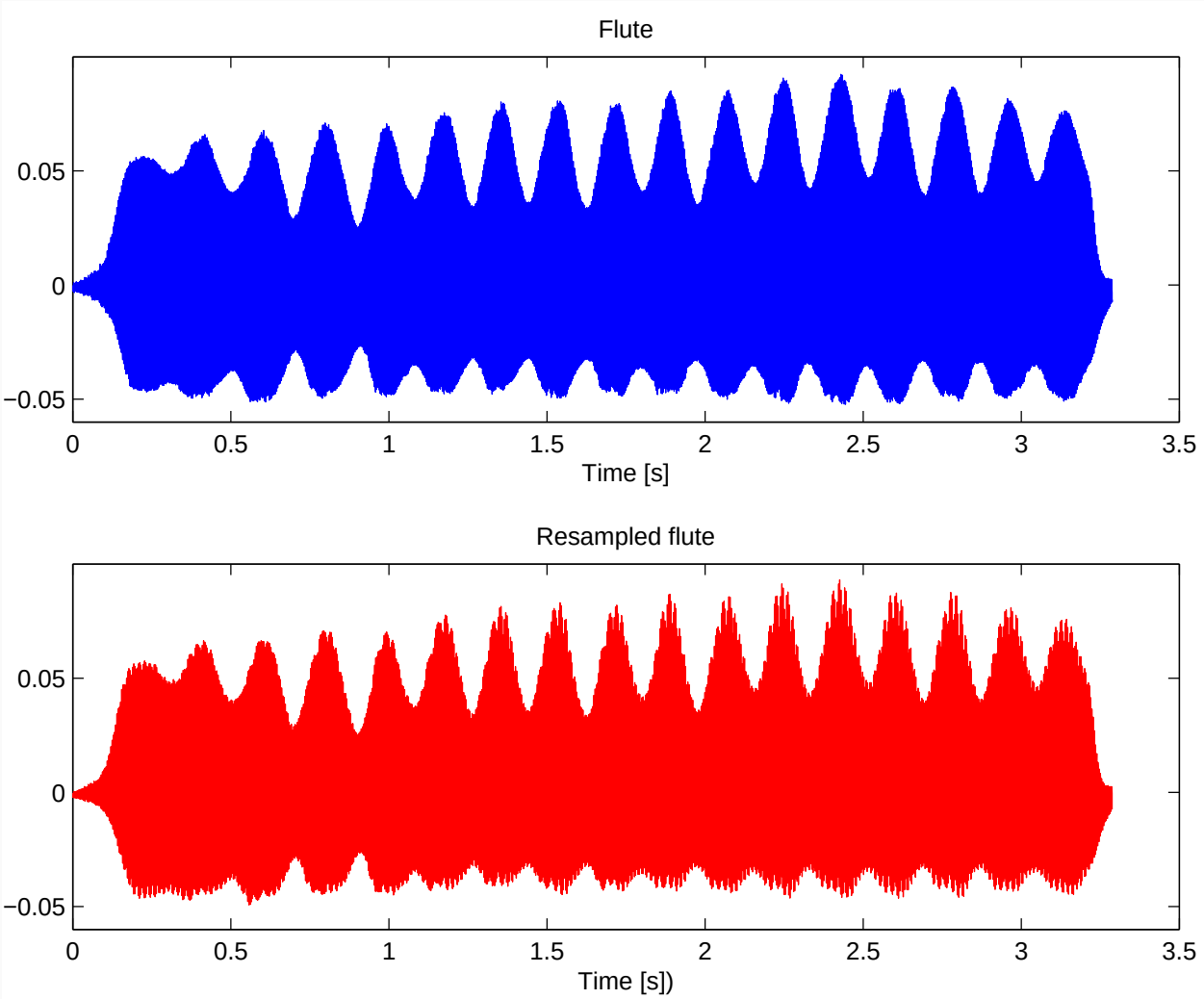
c) Use command `soundsc(x1,sr1)` to obtain flute sound

click to play

d) Resample the audio signal by $f_r = 4000$ Hz and write the sound file to disk using

```
audiowrite('flute-resampled.wav',x2, sr2);
```

MATLAB project – Spectrogram I



e) Compute the DFT of the signal with `X1 = fft(x1(1:1024));` and `X2 = fft(x2(1:1024));`

f) DFT of real-valued signals is always symmetric around `fr/2` so we only need to plot the first half. Display the magnitude of the Fourier transform using

```
plot(f1(1:end/2+1), abs(X1(1:end/2+1)))}
```

g) **Q:** What is the approximate fundamental frequency of the flute note C4?

MATLAB project – Spectrogram I

e) Compute the DFT of the signal with `X1 = fft(x1(1:1024));` and `X2 = fft(x2(1:1024));`

f) DFT of real-valued signals is always symmetric around `fr/2` so we only need to plot the first half. Display the magnitude of the Fourier transform using

```
plot(f1(1:end/2+1), abs(X1(1:end/2+1)))}
```

g) **Q:** What is the approximate fundamental frequency of the flute note C4? A: Find the bin corresponding to the first peak in the magnitude spectrum.

MATLAB project Spectrogram I

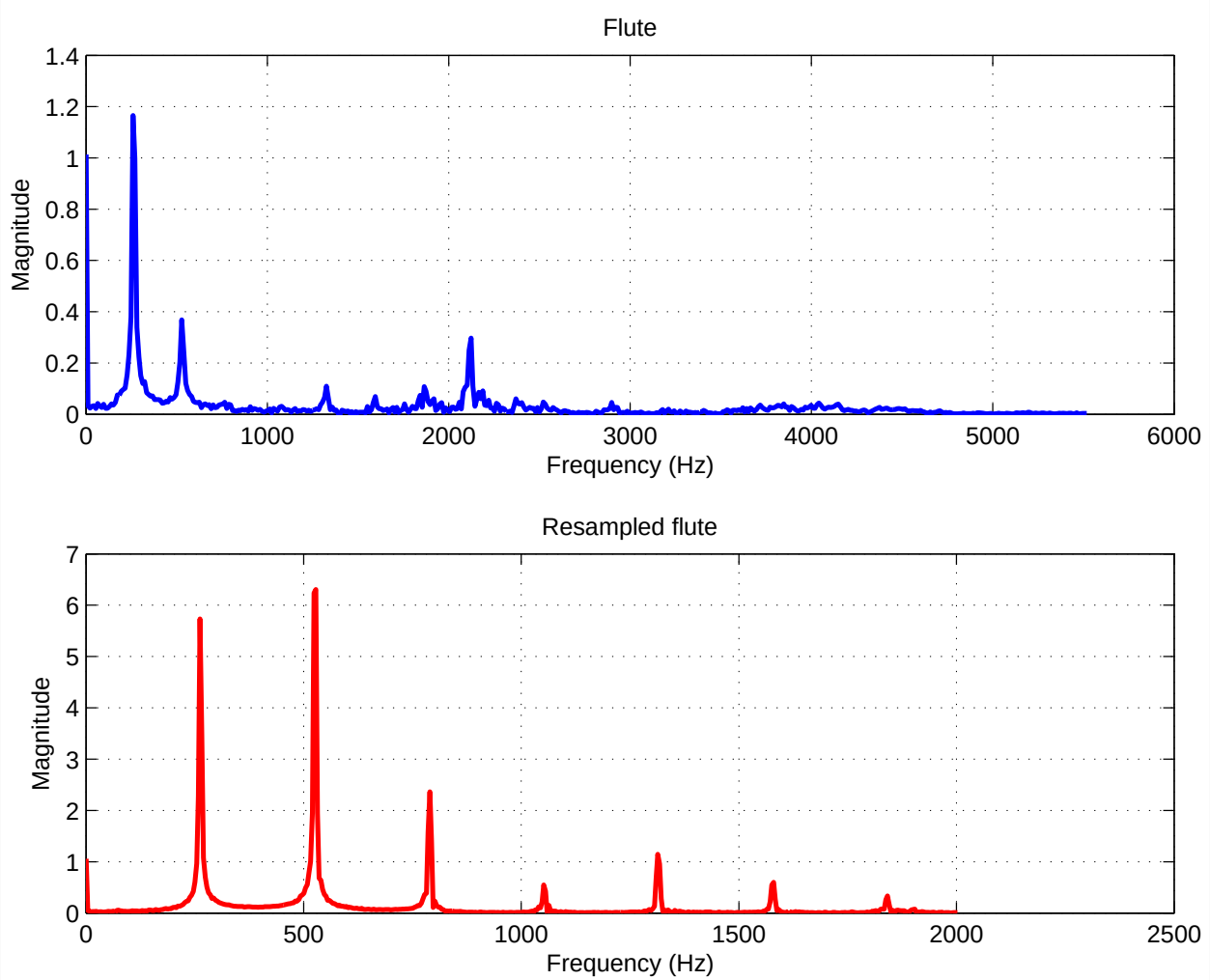
e) You can use a systematic way to find the frequency of the peaks in spectrum `abs(X2)` using following commands:

```
% find local maxima
mag = abs(X2);
mag = mag(1:end/2+1);
peaks = (mag(1:end-2) < mag(2:end-1)) &
(mag(2:end-1) > mag(3:end));
```

f) Then evaluate the peaks at corresponding frequencies above a threshold:

```
peaks = peaks & mag(2:end-1) > 0.5;
fmax = f2(peaks)
```

MATLAB project Spectrogram I

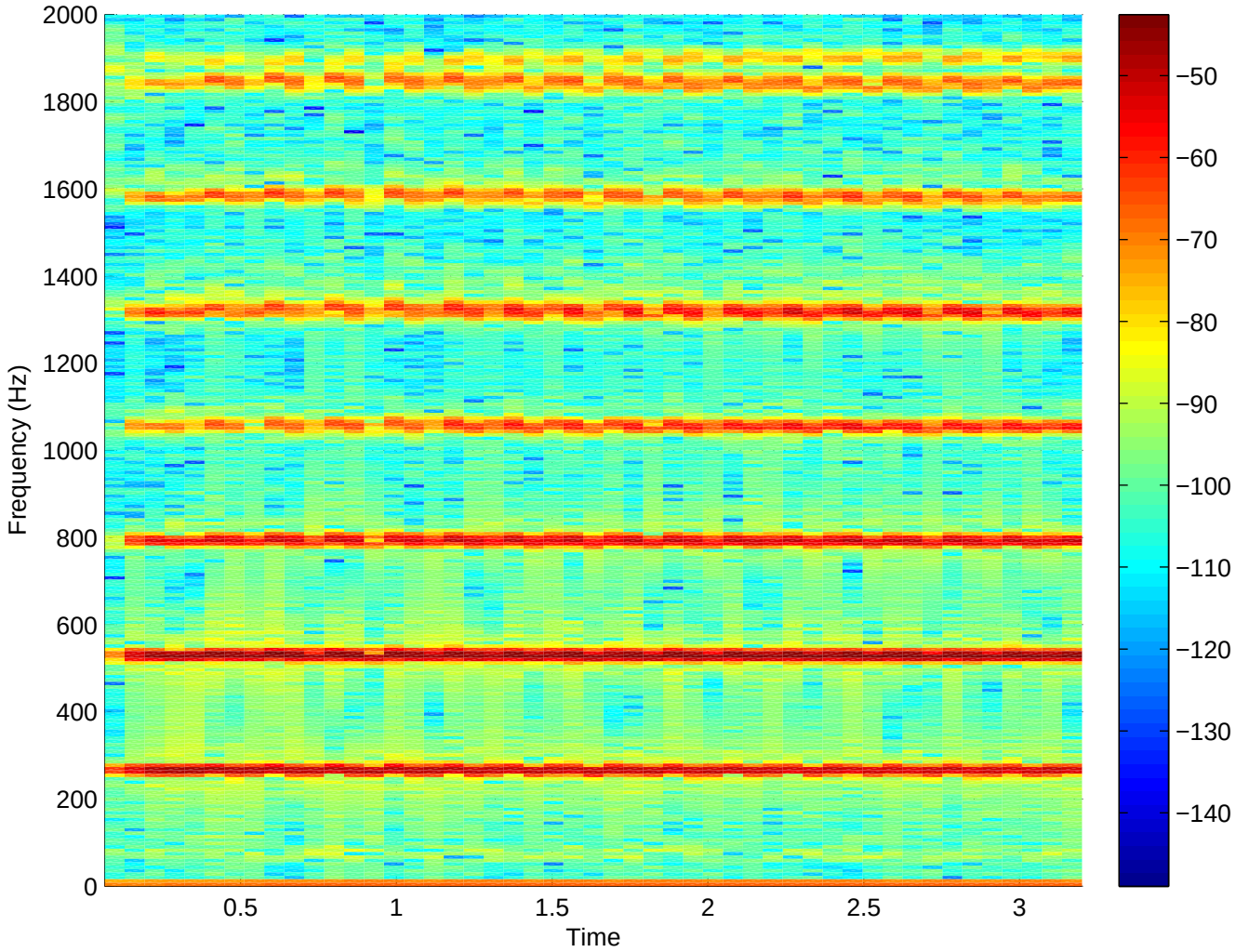


g) Finally we will use Spectrogram with following specifications:

```
nwin = 512; % samples of a window
noverlap = 256; % samples of overlaps
nfft = 512; % samples of fast Fourier
transform
f = figure(1)
spectrogram(x1, nwin, noverlap, nfft, sr2,
'yaxis');
colorbar % not stricly necessary
print -djpeg90 figure-spect % or ...
saveas(f, 'figure-spect.eps', 'epsc);
```

h) Carefully study the options for the spectrogram using help!

MATLAB project Spectrogram – I



MATLAB project Spectrogram – II

a) Construct a signal `f` of two sinusoids on interval $t \in [0, 2)$ sampled at 1 kHz

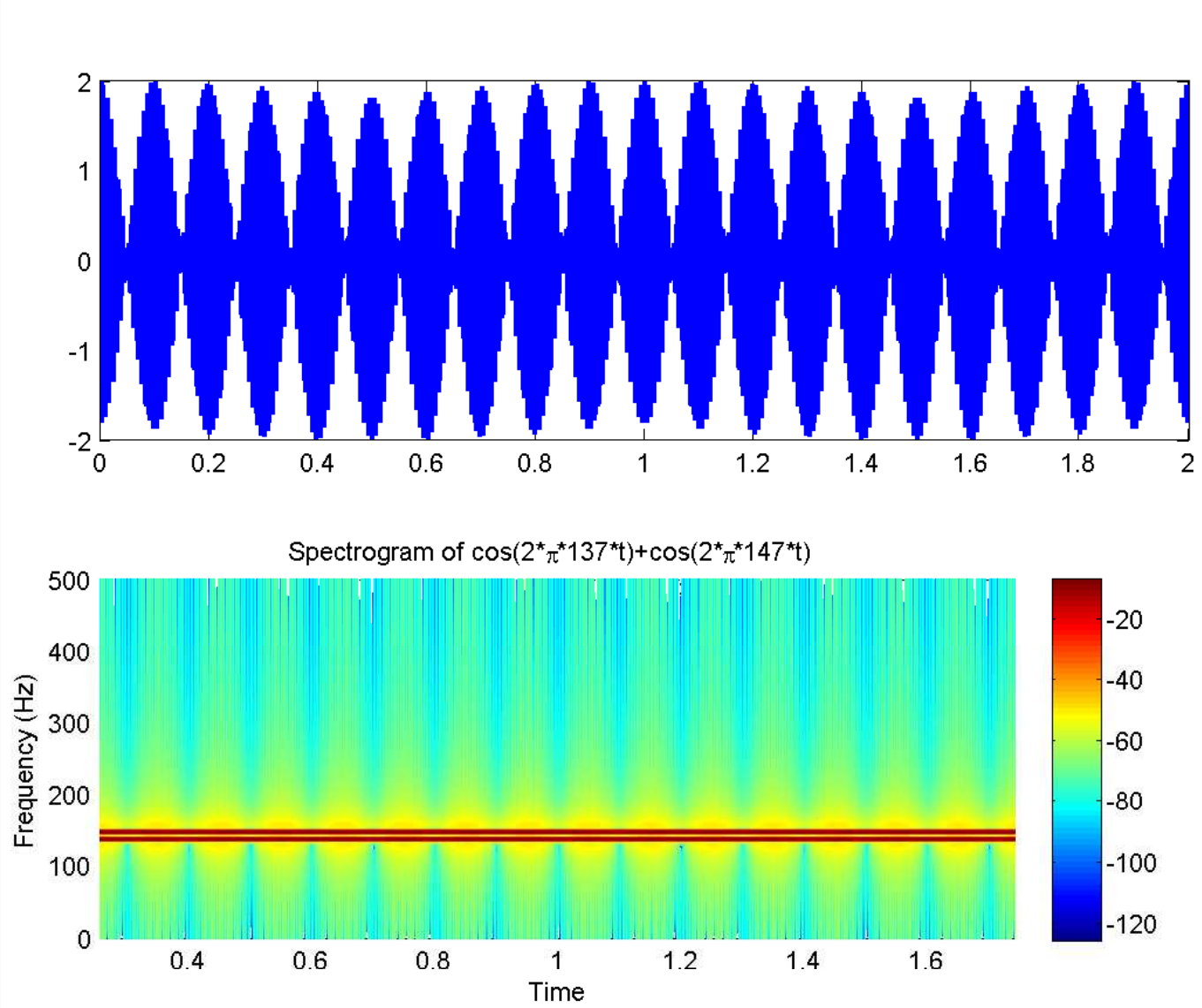
```
t=[0:1999]/1000;
f=cos(2*pi*137*t)+cos(2*pi*147*t);
```

b) Compute the DFT of `f` and display its magnitude in time and frequency domain.

c) Construct a spectrogram using

```
spectrogram(f,512,510,512,1000,'yaxis');
```

MATLAB project – Spectrogram II



MATLAB project Spectrogram II