Introduction to Pattern Recognition WS 2016/17

Daniel Stromer, daniel.stromer@fau.de

Programming Task 1: Basics, Sampling and Fourier Decomposition (15 Points)

Deadline: 06. November 2016

LEHRSTUHL FÜR
MUSTER-
ERKENNUNG

1 pq-Formula Solver (2 P)

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\left(\frac{p}{2}^2 - q\right)} \quad (1)$$

1. Create a class *calculation.py*. In that class, define a function $pqsolver(p, q)$ that takes the two arguments p, q returning result x_1, x_2 according to Eq. 1. Also implement a method of your choice that allows only float real values as output.

2. Create a separate main class *main.py* to test your functions.

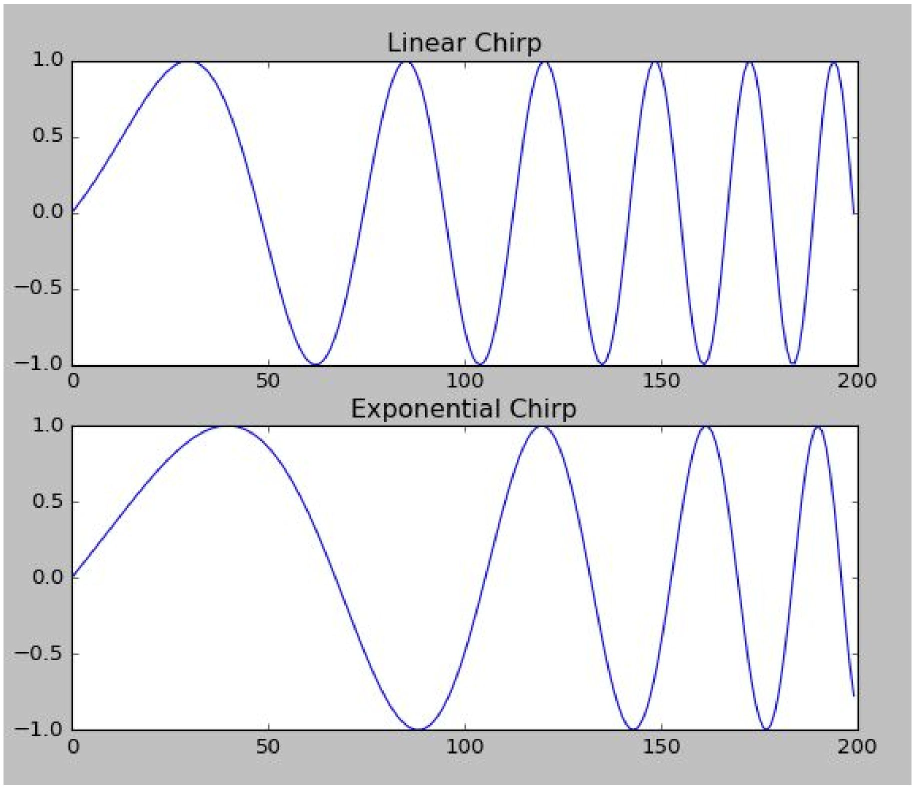
2 Sampling Theorem (5 P)

1. Create a class *chirp.py*. In this class, create a function

   $$createChirpSignal(samplingrate, duration, freqfrom, freqto, linear)$$

   where a chirp-signal is generated and returned. The user should be able choose between a linear and an exponential chirp-signal through setting the variable *linear* to *True* or *False*. The sampling rate (*samplingrate*), the *duration* in seconds and the starting (*freqfrom*) and end frequency (*freqto*) has to be defined. It is not allowed to use the provided function *scipy.signal.chirp*, but of course you can check your result with comparing it to the signal.

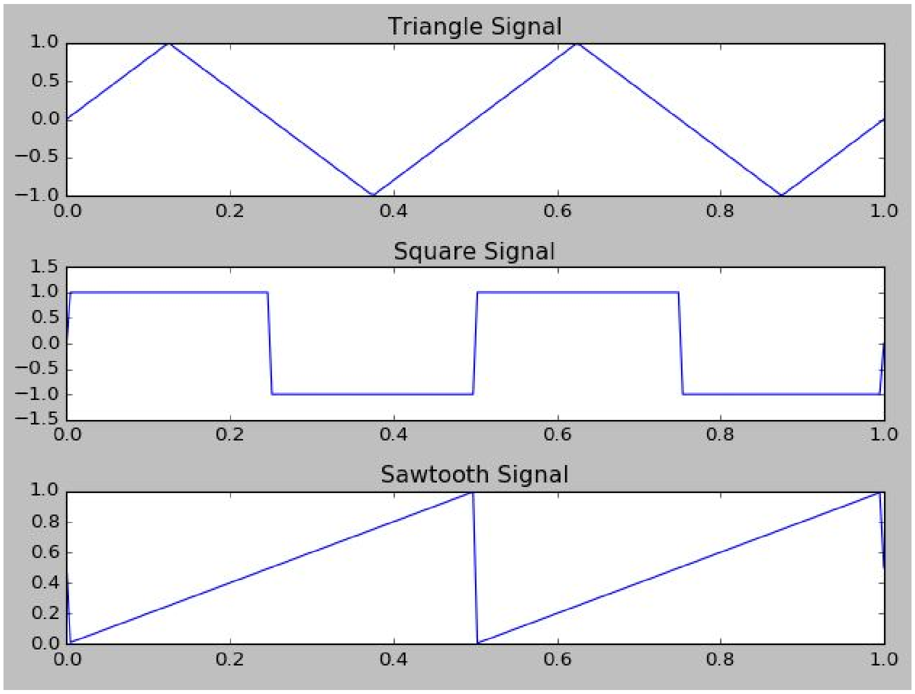
2. In the existing class *main.py*: Create a linear and an exponential chirp signal by calling your chirp function. Use the main function from Task 1 and test your function. (For example: $samplingrate = 200$, $duration = 1,$ frequency from 1 to 10 Hz).



3 Fourier Decomposition (8 P)

In this task, we want to reconstruct a Square, a Triangle and a Sawtooth Signal by implementing their specific Fourier Decomposition functions according to the lecture slides (Chapter 3).

1. Create a class *decomposition.py* and define the three functions shown below returning the specific signal function over the time of one second. *samples* denotes the number of samples, $frequency$ the signals frequency, $kMax$ the last computed k and $amplitude$ the amplitude for the sawtooth signal.
   - $createTriangleSignal(samples, frequency, kMax)$
   - $createSquareSignal(samples, frequency, kMax)$
   - $createSawtoothSignal(samples, frequency, kMax, amplitude)$
2. Test your function in the existing class *main.py*. (Example: $samples = 200$, $frequency = 2$, $kMax = 10000$, $[amplitude = 1]$)


Triangle Signal
Square Signal
Sawtooth Signal