# ALaRI: Digital Signal Processing

## Introductory Exercises

Alexander Bertrand, Deepaknath Tandur, and Toon van Waterschoot[1]

May 2013

## 1 Exercise 1: Sum of sines

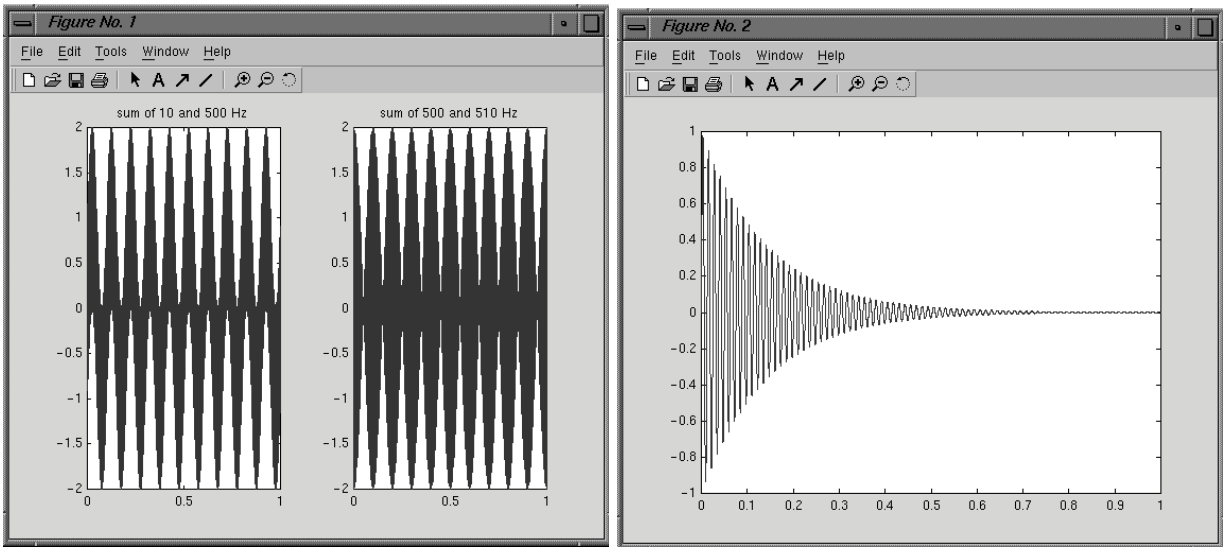


Figure 1: Results of exercise1

- Make a time vector $t$ which starts at 0, runs until 1 sec with a sampling frequency of 10000 Hz.
- Create three sine-waves with frequencies 10, 500 and 510 Hz.
- Create $y$ and $z$ which respectively are the sum of the sine-waves of 10 and 500 Hz. and 500 and 510 Hz.
- Make a plot of $y$ and of $z$. Give both an appropriate title and save them. Zoom in if necessary.

[1]For questions and remarks: toon.vanwaterschoot@esat.kuleuven.be

- Plot the function $f(t)$ defined as:

$$f(t) = sin(500t)e^{-7t}$$

  Save the plot as *f-plot*.

- Save the variables $t$, $y$ and $z$ in a file called *exc1.mat*

## 2 Exercise 2: Discrete system

- A certain discrete system has an input-output function described by

$$y[n] = 0.5\left(y[n-1] + \frac{x[n]}{y[n-1]}\right), n > 0$$

- Find what the output $y$ converges to for $n \to \infty$ if $y[0] = 1$ and $x[n] = \alpha$. Do this by simulating $y$ for different values of *alpha* and plotting $y$. Of course it's impossible to simulate the function for all values of $n$ up to $\infty$; just choose a maximum value of $n$ and see if the function has converged. If it hasn't, increase your maximum value of $n$. You can then read the value the function converges to from the plot with the command `ginput`.

- Determine the settling-time, i.e. the sample-index after which the output signal is always less than 2% from the equilibrium point.

## 3 Exercise 3: Three sequences

- Write a Matlab program (called *sequences.m*) to generate the following sequences.

  1. The unit sample sequence $\delta[n]$
  2. The unit step sequence $\mu[n]$
  3. The ramp sequence $n\mu[n]$

- The input parameter specified by the user is the desired length. Be careful: half the samples should lie before 0 and the other half after 0. Write the program only for odd lengths. (You can check if a number if odd or even with the function `rem`. Read the help page for an explanation on this function).

- Use this program to generate the first 101 samples of the three functions.

- Plot these samples with the command `stem`. Type `help stem` to see what this function does.

- Now plot the same data, but leave out all odd sample-times. Note that this can be done easily with only one (plot)-command, so you don't need commands like `rem` and such.

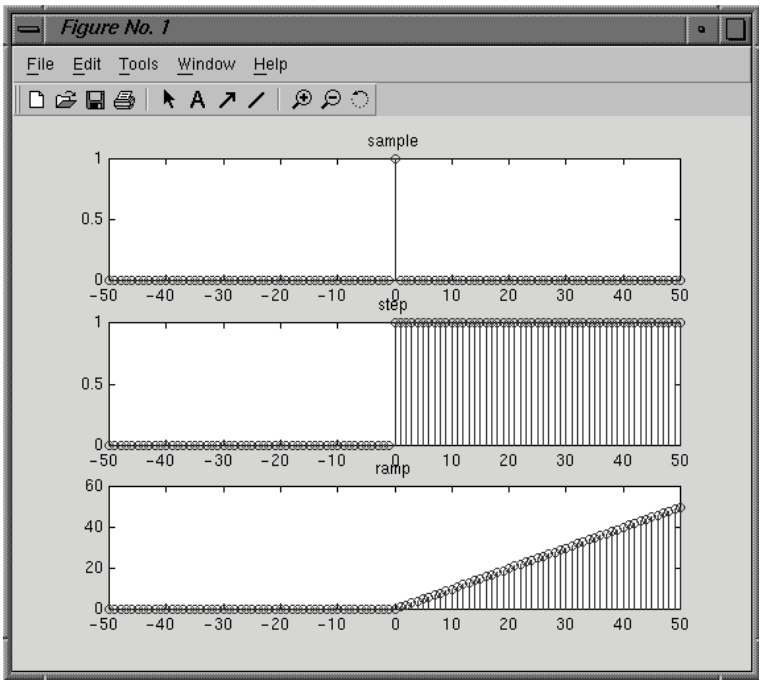


Figure 2: Results of exercise3

# 4 Exercise 4: Square and sawtooth

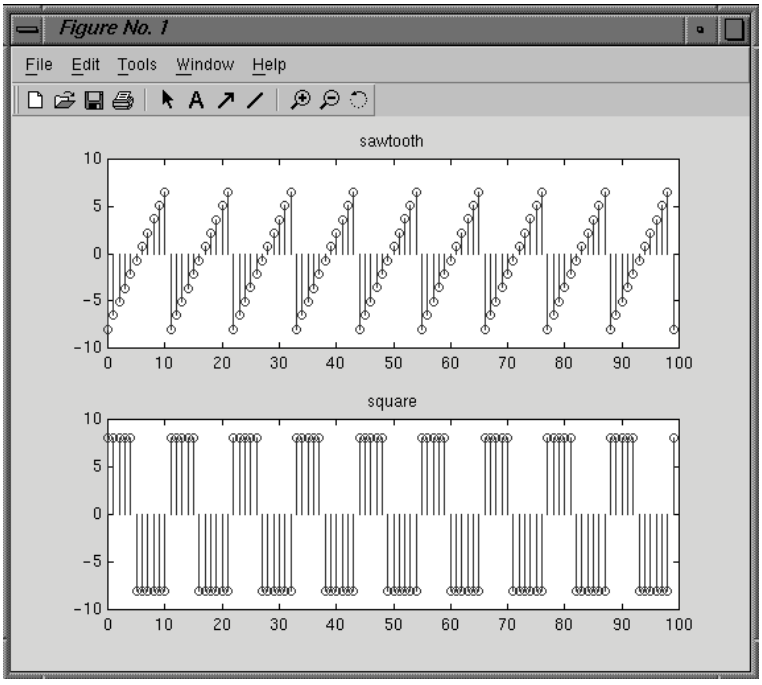


Figure 3: Results of exercise4

- The square wave and the sawtooth are two periodic sequences.
- Write two Matlab programs to generate the two sequences displayed and plot them using the function stem.
- The input parameters specified by the user are the desired length $L$ of the sequence, the peak value $A$ and the period $N$. For the square wave sequence there's one extra parameter: the duty-cycle which is the percent of the period for which the signal is positive.
- Use this program to generate the first 100 samples of the two sequences with a peak value of 8, a period of 11 and a duty-cycle of 40%.

# 5 Exercise 5: Median filter

- The median filter is a non-linear filter that is often used in image-processing to smooth images while at the same time preserving the edges. For us-

age in image-processing one needs the 2D-version of the filter. We will implement the 1D-version.

- The median filter has only one parameter: the length. We will only worry about filters with odd lengths (i.e. length mod 2 is 1).
- To get the result of the filter we slide a window with the given length over the vector we want to filter. We order the values of this window in ascending order and take the middle one. The original value of the element in the middle of the window is replaced by this "median" value. Then we slide the window one place further and repeat the process. The first and last $(length - 1)/2$ values are not changed. Figure 4 will make things clear.
- For the sorting the function `sort` can be useful.
- Your task is to write a function "median"
  `function out = median(length,in)`. The result (stored in out) should be the median filtered version of the vector "in" with a median filter of length "length". Obviously, you may not use the pre-defined command `median`.
- Check the filter on the vector $v$ you find in `medianvector.mat` in `ftp://ftp.esat.kuleuven.be/pub/sista/vanwaterschoot/alari/exercise_session0_MSc/`. The original and filtered vector with a median filter of length 5 look like figure 5.

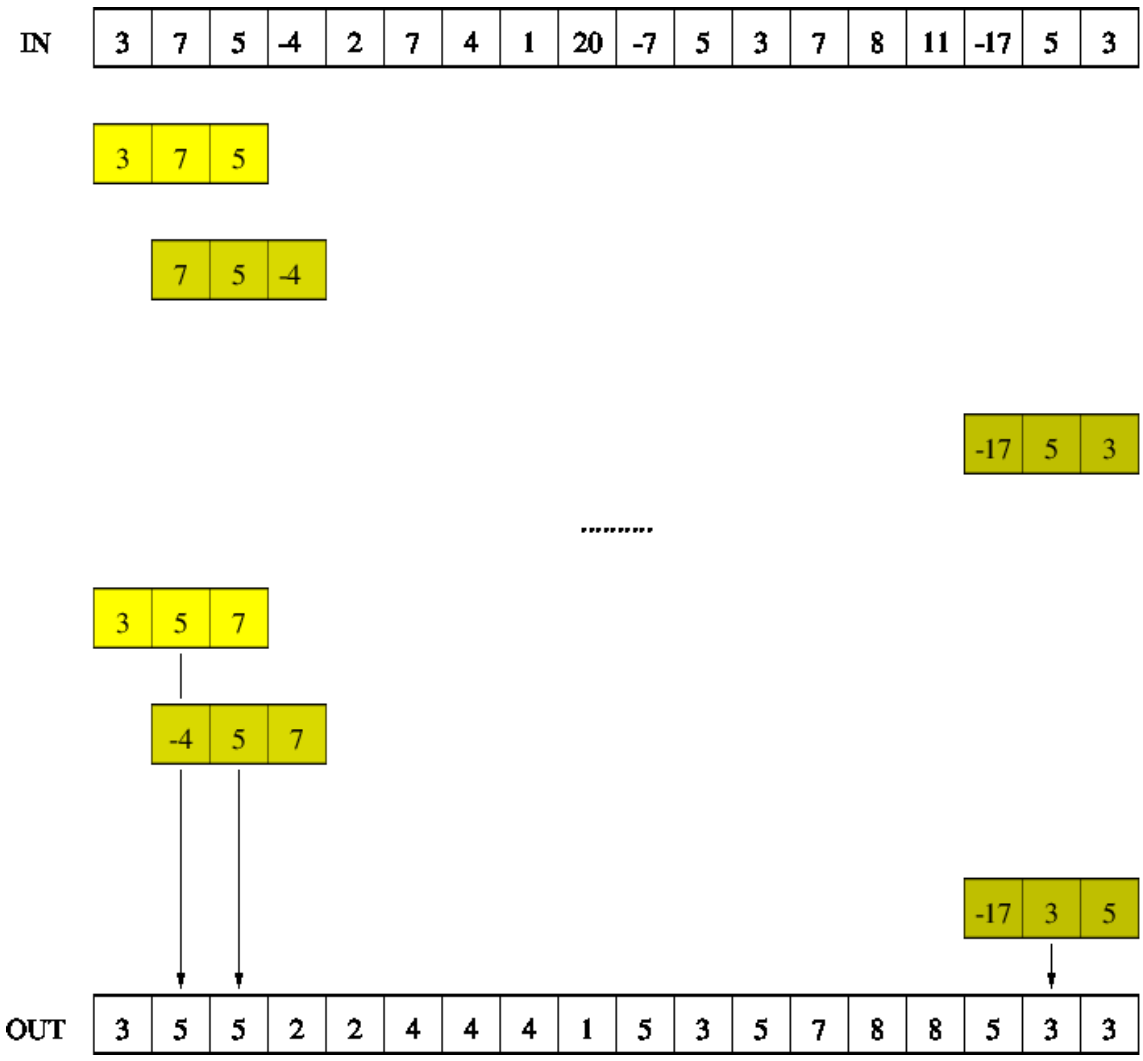


Figure 4: the median filter

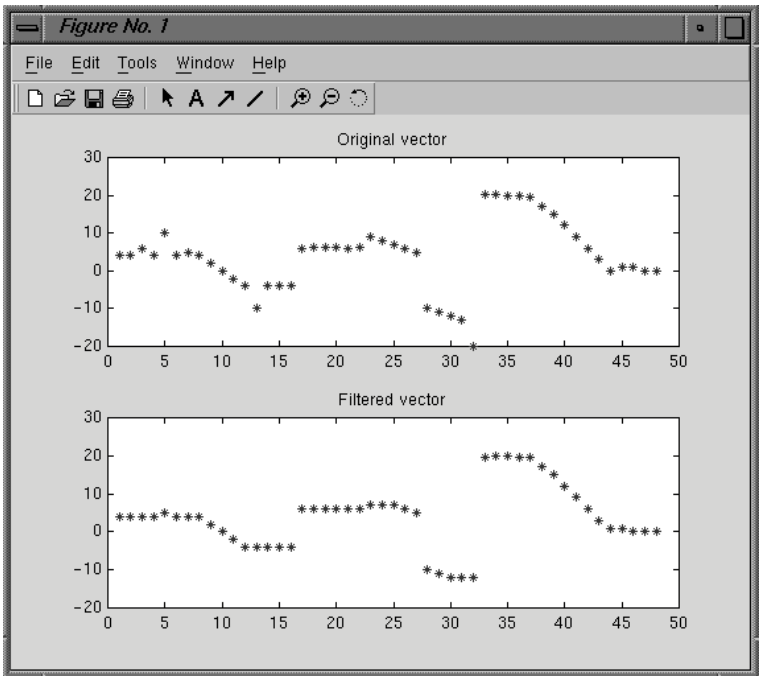


Figure 5: Results of exercise 6

# 6 Exercise 6: The musical scale

- Play the complete musical scale on your computer loudspeakers using Matlab, or compose your own song. All tones are given below (starting with 'do'=C):

| | |
|---|---|
| C | 262 Hz |
| D | 294 Hz |
| E | 330 Hz |
| F | 349 Hz |
| G | 392 Hz |
| A | 440 Hz |
| B | 494 Hz |
| C | 523 Hz |

- Generate a colored image of the spectrogram of this signal. The vertical axis should indicate the frequencies in Hertz, the horizontal axis should indicate the time in seconds. Your colleagues should be able to interpret the spectrogram and write down the 'music' you generated.

# 7 Exercise 7: half-wave rectify

- Generate a white-noise signal that produces approximately the same number of negative and positive values.
- Half-wave rectify this signal, *without* using a control structure (=for, while, if,...). (If you don't know what half-rectification is, google is your friend...)