DSP
COMPUTER SCIENCE TRIPOS Part II

Monday 10 November 2025 12:00 to Monday 17 November 2025 12:00

Module DSP – Digital Signal Processing – Assignment 3

This assignment involves programming. The recommended programming language is Julia *and useful library functions may be found in the Julia packages `DSP.jl`, `WAV.jl`, `FFTW.jl`, `Plots.jl`, and `HDF5.jl`. [Implementations in other suitable languages, using equivalent library functions (such as* MATLAB*'s* Signal Processing Toolbox*, or the Python packages `matplotlib` and `scipy.signal`), are also acceptable.]*

Prepare the solutions and answers to all parts (except for audio files) as a single PDF file and include all source code written, along with any required outputs produced by the programs. A `Pluto.jl` notebook provides a convenient way to combine answer text, Julia source code and graphical outputs into a single PDF.

Submit your work via

https://www.vle.cam.ac.uk/course/view.php?id=256758

no later than 12:00 on Monday 17 November 2025.

Students may be required to sign an undertaking that work submitted will be entirely their own; no collaboration is permitted.

(*a*) Write a program to FM demodulate a single radio station from IQ data:

- The file `iq-fm-96M-240k.dat` (on the course web page) contains 20 seconds of a BBC Radio Cambridgeshire FM broadcast, IQ sampled at the transmitter's centre frequency of 96.0 MHz, at a sample rate of 240 kHz, after having been filtered to 192 kHz bandwidth.

- Load the IQ samples into a vector of complex floating-point numbers.

  In Julia, you can use e.g.

  ```
  function load_iq(fn)
      len = div(filesize(fn), sizeof(ComplexF32))
      z = Vector{ComplexF32}(undef,len)
      read!(fn, z)
  end
  ```

  In MATLAB you can use e.g.

  ```
  f = fopen('iq-fm-96M-240k.dat', 'r', 'ieee-le');
  c = fread(f, [2,inf], '*float32');
  fclose(f);
  z = c(1,:) + 1j*c(2,:);
  ```

- FM demodulate the complex baseband radio signal `z`

- apply a 16 kHz low-pass filter

- reduce the sample rate from 240 kHz down to 48 kHz (keep only every 5th sample using the `:` operator)

- normalize amplitude to use the full scale, then output as WAV (`wavwrite`, `audiowrite`) and listen (`wavplay`, `audioplay`)

Submit the resulting 48 kHz 16-bit mono WAV file with your answer.

(*b*) FM demodulation of multiple radio stations from IQ data:

- The file `iq-fm-97M-3.6M.dat` contains 3.5 seconds of Cambridgeshire radio spectrum, IQ sampled at a centre frequency of 97.0 MHz, with 2.88 MHz bandwidth and a sample rate of 3.6 MHz. Load this file as in Part (*a*).

- Shift the frequency spectrum of this IQ signal such that the carrier of BBC Radio Cambridge ends up at 0 Hz.

- Apply a 200 kHz low-pass filter to select this channel.

- Display the spectrogram of the first 10,000 samples of the signal after each of the preceding three steps. How does the displayed frequency relate to the original radio frequency? (Recall where the negative frequencies are in the FFT output and label your spectrogram with the correct frequencies. You may find the Julia function `fftshift` useful.)

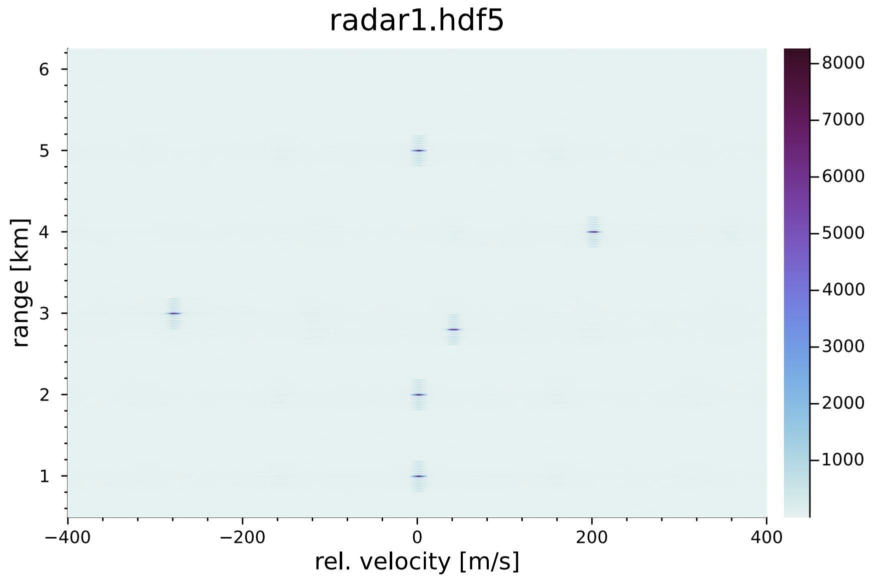
- FM demodulate, low-pass filter, and subsample the signal to 48 kHz, and output it as a 16-bit WAV file, as in Part (*a*).

- Estimate from the spectrogram of the initial signal the centre frequencies of two other FM radio stations within the recorded band, then frequency-shift and demodulate these too.

Include in your submission a single WAV file with the audio from all three demodulated stations concatenated.

(*c*) The region is in turmoil and you suddenly find yourself in a warzone. Your settlement urgently needs a radar to warn the population of incoming rockets, but trade restrictions mean you need to improvise. Your colleague has already successfully built a radar transmitter, by modifying second-hand microwave ovens such that their ≈ 2.45 GHz output waveform can be BPSK-modulated at 10 Mbit/s. She connected those to directional WiFi antennas and reprogrammed the devices to send out "Barker-13" code pulses, repeated at a rate of a few tens of kHz. She also connected a software-defined radio receiver to another antenna, positioned such that it records both the outgoing pulses and the returned echos.

Your job is to implement software that converts the signals recorded by this receiver into a display that shows both the distance ("range") and relative velocity of radar-reflecting objects, such as



Two test recordings `radar1.hdf5` (low noise) and `radar2.hdf5` (high noise) are available on the course web pages. Each contains a contiguous complex-valued time-series vector `iq`, which represents the recorded radio signal after it was IQ downconverted from a centre frequency of `fc` $= 2.45$ GHz, then low-pass filtered with a cut-off frequency of `B`$/2 = 20$ MHz, and finally sampled at a rate of `fs` $= 50$ MHz.

Since the outgoing radar pulses are many orders of magnitude stronger than the incoming reflections, the receiver alternates between two modes: a transmit mode, where it records the outgoing pulse transmitted by the microwave oven (through a signal attenuator), and a receive mode, where it records the incoming echos (through an antenna amplifier). You can easily recognize these two modes when plotting the signal because the signal looks far more noisy in receive mode. The recorded vector `iq` starts with `nt` samples recording the first outgoing pulse, followed by `nr` samples of incoming echoes, followed by `nt` samples recording the second outgoing pulse, and so on, for a total of (`nt` + `nr`) $\times$ `np` samples coverning `np` = 256 pulses. (These parameters can be read from each HDF5 file.)

- As a warm-up exercise plot the cross-correlation between the first pulse and its echoes, i.e. between the first `nt` samples and the next `nr` samples, showing only the part where they fully overlap. Label the time axis according to the range in kilometers. At what ranges can you see echoes? [*Hint:* If the correlation result from a single pulse is too noisy, try averaging it over many pulses. Does coherent vs. non-coherent averaging make a difference?]

- Then prepare a matrix `xc` where each of the `np` columns contains the cross-correlation between the corresponding pulse and the following echos. The peaks of stationary reflectors will have a constant phase angle, as the distance between radar and object does not change. The peaks of moving objects, on the other hand, will rotate between pulses, like a complex phasor, completing one rotation each time the round-trip distance of the radar pulse changes by one wavelength. Apply DFT-based spectral estimation along each row of `xc`, to turn each rotating phasor into a rotational-frequency peak, to horizontally separate for each range the echos of objects with different relative velocity (Doppler shift), and plot the resulting matrix as a `heatmap`. Use the wavelength of the carrier signal and the time-period between pulses in order to convert the rotational frequency into a radial velocity of the object relative to the radar antenna, and label the horizontal axis accordingly.

- Check that your processing of `radar1.hdf5` results in a range-doppler-map as in the `heatmap` shown above, with six reflections at ranges of 1.0, 2.0, 2.8, 3.0, 4.0 and 5.0 km, where the one at 3 km might be a rocket approaching with 280 m/s, the one at 4 km might be a plane flying away at 200 m/s, and the one at 2.8 km might be the tip of a wind-turbine blade moving away at 40 m/s.

- Produce an equivalent range-doppler-map for `radar2.hdf5` and report on the number of reflections and their distance and relative velocity.

END OF PAPER