Class Test 02

1. (a) Consider a time-domain signal that consists of two sinusoidal components with frequencies of 50 Hz and 120 Hz. Random noise with an amplitude of 2.5 is added to this clean signal, resulting in a noisy version. Using Python, perform the following tasks to analyze and filter the noise from the signal:

   Define and Generate the Signals:

   - Set the sampling interval `dt` to 0.001 seconds, and generate a time array t from 0 to 1 second with this interval.
   - Create the clean signal as a sum of two sine waves: one at 50 Hz and the other at 120 Hz.
   - Add random noise to this clean signal to produce the noisy signal. Syntax: `2.5*np.random.randn(len(t))`

   Plot the Signals in the Time Domain:

   - Plot both the noisy and clean signals on the same graph, using distinct colors for each.
   - Label the axes appropriately, set the x-axis limits from the start to the end of the time array, and include a legend.

   Perform FFT and Compute the Power Spectral Density (PSD):

   - Calculate the Fast Fourier Transform (FFT) of the noisy signal.
   - Compute the Power Spectral Density (PSD) from the FFT by squaring the magnitude of the Fourier coefficients and dividing by the length of the signal.
   - Define a frequency array using the sampling interval dt and length of the signal.

   Plot the Power Spectral Density (PSD) in the Frequency Domain:

   - Plot the PSD for the positive frequency components only.
   - Set the x-axis limits appropriately, label the axes, and add a legend to the plot.

   Filter the Noise by Applying a Threshold to the PSD:

   - Set a threshold value to filter out frequencies that correspond to low PSD values.
   - Zero out the FFT coefficients corresponding to frequencies with PSD values below this threshold.
   - Compute the inverse FFT of the filtered coefficients to reconstruct a "cleaned" version of the signal.

   Plot the Filtered Signal in the Time Domain:

- Plot the filtered signal and compare it with the original noisy signal.
- Ensure the plot is well-labeled and includes a legend.

1. (b) The “metal.wav”, is the audio file of a iron metal disc rotation on the cement floor. Using python complete the following tasks.

   Task 01: Convert the time domain data into frequency domain data using FFT.

   Task 02: Find the dominating frequency of the audio.

   Task 03: Plot the spectrogram of the audio.

   Task 4: Using the noise reduction algorithm which has been given in question 1(a) reduce the noise.

   Task 05: Plot the spectrogram for the noise free audio signal.

2. Consider an ideal refrigeration cycle that uses R134a as the working fluid. The temperature of the refrigerant in the evaporator is -10 ℃ and the temperature in the condenser is 55 ℃. Determine the heat extracted at the evaporator and the coefficient of performance (COP) of the refrigeration plant.

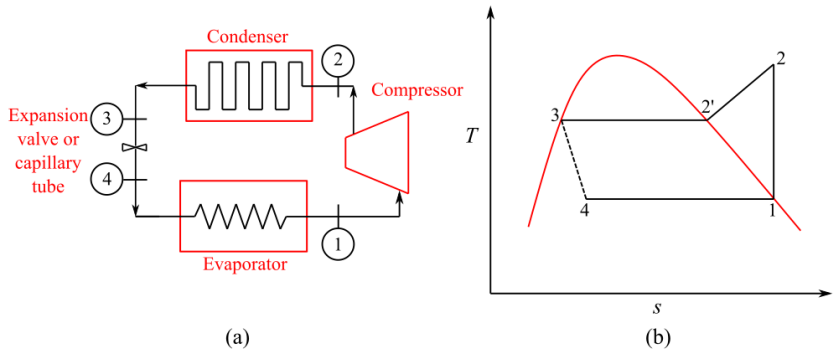


Figure: Ideal Refrigeration Cycle (Vapor Compression)

Compressor analysis: (1 – 2) is an isoentropic process, $s_1 = s_2$. The power input to the compressor can be found by the enthalpy difeerence between state 2 and 1.

Condenser analysis: (2 - 3) is an isobaric heat rejection process. The heat rejected by condenser can be formulated by the enthalpy difference between point 3 and 2.

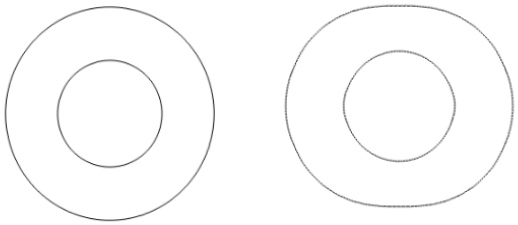
Expansion valve analysis: (3 - 4) is an isoenthalpic process, $h_3 = h_4$.

Evaporator analysis: (4 - 1) is an isobaric heat addition process. The heat addition to the evaporetor is equal to the difference in enthalpy at state 1 and 4.

$$COP = \frac{h_1 - h_4}{h_2 - h_1}$$

Hint: `prop_R134a = obj.get("mp.C2H2F4")`

3. The file "compound_droplet" stack of images (.png) of compound droplet. The compound droplet consists of an outer shell and a concentric core. Both the core and the shell deforms due to the presence of a uniform electric field. The images in the folder are named as "axyz" where x, y, z are numeric digits and $time = \frac{xyz-1}{10}\ seconds$. Find variation of the deformation of the core and the shell with time. Print the final and initial deformation.



Before deformation and after deformation

$$Deformation = \frac{Major\ \ length - Minor\ \ length}{Mejor\ length + Minor\ length}$$

Hint:

The code `` `image_files = sorted(os.listdir(folder))` `` retrieves a list of all filenames in the specified `` `folder` `` and sorts them in lexicographical order. This ensures that the images are processed in the correct sequence based on their names.

```
Syntax: import os
folder = 'compound_droplet'
image_files = sorted(os.listdir(folder))
```