ANALISA SUARA EXHAUST MOTOR 4 CYLINDER BERDASARKAN CC MESIN

****

Disusun oleh :

**JAMES PHILIP (5024211064)**

**DEPARTEMEN TEKNIK KOMPUTER**

**FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS**

**(FTEIC)**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA**

**2023**

**Kata Pengantar**

Salam sejahtera dalam nama Tuhan Yang Maha Esa,

Dengan rasa syukur dan pengharapan yang tinggi, Penulis dengan penuh keikhlasan dan dedikasi menyusun makalah ini dengan tema Pengolahan Sinyal Digital dalam Perbandingan antara 3 Jenis Exhaust Motor Berdasarkan CC pada Kondisi Idle dengan Python. Makalah ini merupakan hasil dari kerja keras dan kerjasama tim Penulis dalam menggali lebih dalam pengetahuan tentang pengolahan sinyal digital serta menerapkannya dalam konteks industri otomotif.

Penulis percaya bahwa dalam era yang serba digital ini, pengolahan sinyal digital menjadi hal yang sangat relevan dan penting. Dalam konteks perbandingan antara jenis exhaust motor berdasarkan kapasitas mesin pada kondisi idle, pengolahan sinyal digital dapat memberikan pemahaman yang lebih komprehensif mengenai performa dan karakteristik masing-masing motor. Melalui penggunaan bahasa pemrograman Python, Penulis dapat mengolah data dengan efisien dan menghasilkan analisis yang akurat.

Makalah ini dirancang untuk memberikan wawasan mendalam kepada pembaca mengenai pentingnya pengolahan sinyal digital dalam dunia otomotif serta penggunaan Python sebagai alat yang efektif dalam proses pengolahan data. Penulis mengharapkan bahwa makalah ini dapat memberikan sumbangan berharga dalam bidang penelitian dan pengembangan di industri otomotif, serta menjadi sumber inspirasi bagi para profesional, akademisi, dan mahasiswa yang tertarik dalam bidang ini.

Penulis ingin mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dan bantuan selama penyusunan makalah ini. Penulis juga berterima kasih kepada Tuhan YME yang telah memberikan kekuatan, hikmat, dan petunjuk dalam setiap langkah Penulis.

Penulis berharap dengan penuh rendah hati bahwa makalah ini dapat menjadi bahan referensi yang berharga dan mendorong terjadinya peningkatan dalam penelitian, pengembangan, dan penerapan pengolahan sinyal digital dalam industri otomotif. Marilah kita bersama-sama menjelajahi dan menerangi dunia ini dengan ilmu pengetahuan yang bermanfaat bagi umat manusia.

Akhir kata, Penulis berdoa agar Tuhan YME senantiasa memberkati dan melindungi kita semua dalam segala langkah dan perjalanan kita. Semoga makalah ini memberikan manfaat dan terus menerangi jalan pengetahuan kita.

Terima kasih.

Salam hormat,

James Philip

**BAB 1**

**PENDAHULUAN**

* 1. **Latar Belakang**

Dalam dunia otomotif, pengembangan teknologi kendaraan terus mengalami perkembangan yang pesat. Salah satu aspek penting dalam pengembangan kendaraan adalah sistem knalpot (exhaust system), yang memiliki peran vital dalam mengatur aliran gas buang dan mengoptimalkan kinerja mesin. Salah satu faktor yang mempengaruhi performa knalpot adalah kapasitas mesin atau disebut juga dengan CC (Cubic Centimeters).

Dalam kondisi idle, yaitu saat mesin berjalan tanpa beban atau dalam keadaan diam, karakteristik suara dan getaran yang dihasilkan oleh knalpot menjadi sangat penting. Kualitas suara dan getaran dapat memberikan informasi mengenai performa mesin, efisiensi pembakaran, dan tingkat kebisingan yang dihasilkan oleh kendaraan. Oleh karena itu, perbandingan antara jenis exhaust motor berdasarkan CC pada kondisi idle menjadi topik yang menarik untuk diteliti. Pengolahan sinyal digital menjadi salah satu pendekatan yang efektif dalam menganalisis karakteristik suara dan getaran knalpot. Dalam pengolahan sinyal digital, data suara dan getaran diubah menjadi bentuk yang dapat dianalisis menggunakan algoritma dan metode tertentu. Dalam konteks ini, penggunaan bahasa pemrograman Python sebagai alat untuk mengolah data menjadi sangat relevan. Python memberikan kemudahan dalam manipulasi data, pengolahan sinyal, visualisasi, dan analisis statistik.

Makalah ini bertujuan untuk melakukan perbandingan antara 3 jenis exhaust motor berdasarkan CC pada kondisi idle menggunakan pendekatan pengolahan sinyal digital dengan menggunakan Python sebagai alat utama. Dalam makalah ini, kami akan menjelaskan langkah-langkah pengolahan sinyal digital yang meliputi pemrosesan data suara dan getaran, analisis karakteristik, serta perbandingan antara jenis exhaust motor. Penelitian ini memiliki potensi untuk memberikan kontribusi yang signifikan dalam pengembangan dan pemahaman mengenai karakteristik exhaust motor berdasarkan CC pada kondisi idle. Hasil dari penelitian ini dapat menjadi acuan bagi produsen kendaraan dalam merancang knalpot yang efisien dan menghasilkan suara serta getaran yang optimal. Selain itu, penelitian ini juga dapat memberikan wawasan dan inspirasi bagi peneliti, akademisi, dan mahasiswa yang tertarik dalam bidang pengolahan sinyal digital dan industri otomotif.

Dengan demikian, melalui makalah ini kami berharap dapat memberikan kontribusi yang bermanfaat bagi perkembangan ilmu pengetahuan dan teknologi, serta memberikan informasi yang berguna bagi industri otomotif dalam mengembangkan sistem knalpot yang lebih baik.

* 1. **Dasar Teori**

Pengolahan sinyal digital memiliki peran penting dalam membandingkan karakteristik exhaust motor 4 silinder berdasarkan cc mesin seperti 250cc, 600cc, dan 1000cc. Pengolahan sinyal digital adalah proses manipulasi dan analisis sinyal suara menggunakan algoritma komputasi.

Dalam konteks exhaust motor, pengolahan sinyal digital memungkinkan kita untuk menggali informasi yang terkandung dalam suara knalpot dan membandingkan karakteristik antara motor-motor dengan cc yang berbeda. Beberapa konsep dasar yang terkait dengan penerapan pengolahan sinyal digital untuk tujuan ini meliputi:

Akuisisi Sinyal: Sinyal suara dari knalpot motor dapat direkam menggunakan mikrofon dan diubah menjadi format digital. Ini melibatkan konversi sinyal analog menjadi sinyal digital dengan mengambil sampel suara pada interval waktu tertentu. Proses ini disebut akuisisi sinyal.

Preprocessing: Sebelum analisis lebih lanjut, sinyal suara knalpot perlu melalui tahap preprocessing. Ini dapat mencakup langkah-langkah seperti normalisasi amplitudo untuk menghindari distorsi, penghilangan noise yang tidak diinginkan, dan pengurangan gangguan frekuensi rendah atau tinggi yang tidak relevan.

Transformasi Fourier: Transformasi Fourier digunakan untuk mengubah sinyal suara dari domain waktu menjadi domain frekuensi. Ini memungkinkan kita untuk melihat kontribusi berbagai komponen frekuensi dalam sinyal suara. Dengan transformasi Fourier, kita dapat menganalisis spektrum frekuensi dan mengidentifikasi karakteristik yang berbeda antara motor-motor dengan cc yang berbeda.

Ekstraksi Fitur: Pengolahan sinyal digital juga melibatkan ekstraksi fitur dari sinyal suara. Fitur-fitur ini dapat mencakup amplitudo maksimum, durasi suara, tingkat kebisingan, dan fitur-fitur statistik lainnya. Ekstraksi fitur memberikan representasi numerik yang dapat digunakan untuk membandingkan dan mengklasifikasikan karakteristik exhaust motor dengan cc yang berbeda

Analisis dan Visualisasi: Hasil pengolahan sinyal digital dapat dianalisis dan divisualisasikan untuk pemahaman yang lebih baik. Grafik spektrum frekuensi, grafik amplitudo, dan visualisasi lainnya dapat membantu dalam membandingkan karakteristik suara antara motor-motor dengan cc yang berbeda

Dengan menggunakan teknik pengolahan sinyal digital, kita dapat menggali informasi yang terkandung dalam suara knalpot motor dan membandingkan karakteristik exhaust motor 4 silinder berdasarkan cc mesin seperti 250cc, 600cc, dan 1000cc. Pendekatan ini memberikan pemahaman yang lebih dalam tentang perbedaan dalam spektrum frekuensi, amplitudo, durasi, dan fitur audio lainnya antara motor-motor dengan cc yang berbeda.

* 1. **Rumusan Masalah**

Berdasarkan latar belakang yang telah dibuat , maka dapat diidentifikasi beberapa permasalahan yang didapat untuk tugas dari mataa kuliah Pengolahan Sinyal Digital yaitu pembuatan suatu aplikasi atau program dengan Python untuk memfilter suara-suara yang diberikan (Exhaust Sound) dari 3 data motor yang ada.

* 1. **Tujuan**

Tujuan utama dari pembuatan program ini adalah untuk memenuhi tugas dari mata kuliah Pengolahan Sinyal Digital. Selain itu ada juga tujuan lain dari pembuatan dan pengaplikasian kode yang dibuat adalah sebagai berikut :

1. Menganalisis karakteristik suara dan getaran knalpot pada kendaraan berdasarkan CC (Cubic Centimeters) pada kondisi idle.
2. Membandingkan tiga jenis exhaust motor berdasarkan kapasitas mesin (CC) pada kondisi idle.
3. Menggunakan pendekatan pengolahan sinyal digital dan bahasa pemrograman Python untuk memproses data suara dan getaran knalpot.
4. Mendapatkan hasil filter dari pemfilteran suara idle 3 motor yang menjadi datanya.

**BAB 2**

**PEMBAHASAN**

**2.1 Metodologi**

Dalam pembuatan dan persiapan program untuk menfilter suatu suara dapat diperlukan beberapa step atau langkah-langkah yang bisa diikuti. Metodologi yang dilakukan terdiri dari persiapan data, import libraries, analisa data, filter dan pemrosesan, analisa perbandingan, dan visualisasi grafis dari output yang telah difilter. Dari beberapa step diatas maka akan diperoleh suara yang berbeda atau yang sudah defiler. Berikut merupakan contoh penjelasan metodologi dari pemfilteran suara motor tersebut :

1. Persiapan Data
   * Kumpulkan data suara knalpot dari masing-masing motor dengan menggunakan mikrofon atau alat perekam suara.
   * Simpan data suara dalam format WAV untuk kompatibilitas dengan modul pengolahan sinyal digital di Python.
   * Pastikan data yang dikumpulkan mencakup variasi kondisi penggunaan motor yang relevan, seperti berbagai kecepatan putaran mesin atau beban akselerasi.
2. Impor Libraries
   * Impor library **scipy** untuk membantu dalam pengolahan sinyal digital.
   * Impor library **matplotlib** untuk visualisasi data.
3. Baca dan Analisis Data
   * Gunakan library **scipy** untuk membaca file WAV dan mengambil data suara. Misalkan kita memiliki tiga file WAV: **motor250cc.wav**, **motor600cc.wav**, dan **motor1000cc.wav**.
   * Menggunakan fungsi **scipy.io.wavfile.read()**, baca file WAV dan simpan ke dalam variabel dengan nama yang sesuai (misalnya **samplerate\_250, data\_250 = wavfile.read('motor250cc.wav')**).
   * Analisis sinyal suara menggunakan library **scipy** dengan mengaplikasikan operasi-transformasi Fourier untuk mendapatkan spektrum frekuensi dari sinyal suara. Fungsi **scipy.fft()** dapat digunakan untuk ini.
4. Filter dan Pemrosesan
   * Terapkan filter sesuai dengan tujuan perbandingan yang ingin dilakukan. Misalnya, Anda dapat menerapkan filter low-pass untuk membatasi frekuensi suara yang ingin dianalisis.
5. Ekstraksi Fitur
   * Identifikasi dan ekstraksi fitur-fitur penting dari sinyal suara yang telah diproses. Fitur-fitur ini dapat mencakup amplitudo maksimum, durasi suara, atau fitur-fitur statistik lainnya.
6. Analisis dan Perbandingan
   * Bandingkan fitur-fitur yang diekstraksi antara motor-motor dengan cc yang berbeda menggunakan metode analisis statistik atau metode pemrosesan sinyal lainnya.
   * Misalnya, kita dapat menggunakan uji hipotesis atau analisis perbandingan statistik untuk membandingkan rata-rata amplitudo atau durasi suara antara motor 250cc, 600cc, dan 1000cc.
7. Visualisasi
   * Gunakan library **matplotlib** untuk membuat visualisasi grafis yang memperlihatkan perbedaan karakteristik suara antara motor-motor dengan cc yang berbeda.
   * Buat grafik spektrum frekuensi, grafik amplitudo, atau visualisasi lainnya untuk memperlihatkan perbedaan secara jelas dan informatif.

**2.2 Code**

Berikut merupakan code per sample variasi yaitu 250cc, 600cc, dan 1000cc :

1. **250cc (ZX25R)**

from scipy.io import wavfile

import IPython

import matplotlib.pyplot as plt

import sounddevice as sd

samplerate, data = wavfile.read('./zx25r.wav')

IPython.display.Audio('./zx25r.wav')

mono\_data = data[:, 0]

plt.plot(mono\_data)

plt.show()

from scipy import io, signal, fft

import numpy as np

N3 = len(data3)

T3 = 1. / samplerate3

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

N = len(mono\_data)

T = 1.0 / samplerate

fft\_data = fft.fft(mono\_data)

xf = fft.fftfreq(N, T)[:N//2]

amplitude = 2.0/N \* np.abs(fft\_data[0:N//2])

plt.figure(figsize=(12, 6))

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.xlim(0, 1000)

plt.ylim(0, 2000)

plt.show()

from scipy import io, signal, fft

import numpy as np

N3 = len(data3)

T3 = 1. / samplerate3

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

N = len(mono\_data)

T = 1.0 / samplerate

fft\_data = fft.fft(mono\_data)

xf = fft.fftfreq(N, T)[:N//2]

amplitude = 2.0/N \* np.abs(fft\_data[0:N//2])

plt.figure(figsize=(12, 6))

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.xlim(0, 1000)

plt.ylim(0, 2000)

plt.show()

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

N = len(mono\_data)

T = 1.0 / samplerate

fft\_data = fft.fft(mono\_data)

xf = fft.fftfreq(N, T)[:N//2]

amplitude = np.abs(fft\_data[N//2:])/N \* np.abs(fft\_data[0:N//2])

plt.figure(figsize=(12, 6))

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

b, a = signal.butter(1, [4000, 7000], btype='bandpass', analog=True)

w, h = signal.freqs(b, a)

plt.figure(figsize=(12, 6))

plt.semilogx(w, 20 \* np.log10(abs(h)))

plt.axvline(0.67, color='green')

plt.axvline(40, color='green')

plt.axhline(-3, color='green')

plt.grid(which='both', axis='both')

plt.ylim((-30, 0))

plt.xlabel('Frequency (Hz)')

plt.ylabel('Magnitude (dB)')

plt.show()

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

sos = signal.butter(1, 18000, 'hp', fs=samplerate, output='sos')

w, h = signal.sosfreqz(sos)

plt.figure(figsize=(12, 8))

plt.subplot(2, 1, 1)

db = 20\*np.log10(np.maximum(np.abs(h), 1e-5))

plt.plot(w/np.pi, db)

plt.ylim(-75, 5)

plt.grid(True)

plt.yticks([0, -20, -40, -60])

plt.ylabel('Gain [dB]')

plt.title('Frequency Response')

plt.subplot(2, 1, 2)

plt.plot(w/np.pi, np.angle(h))

plt.grid(True)

plt.yticks([-np.pi, -0.5\*np.pi, 0, 0.5\*np.pi, np.pi],

           [r'$-\pi$', r'$-\pi/2$', '0', r'$\pi/2$', r'$\pi$'])

plt.ylabel('Phase [rad]')

plt.xlabel('Normalized frequency (1.0 = Nyquist)')

plt.show()

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

sos = signal.butter(1, 18000, 'hp', fs=samplerate, output='sos')

w, h = signal.sosfreqz(sos)

plt.figure(figsize=(12, 8))

plt.subplot(2, 1, 1)

db = 20\*np.log10(np.maximum(np.abs(h), 1e-5))

plt.plot(w/np.pi, db)

plt.ylim(-75, 5)

plt.grid(True)

plt.yticks([0, -20, -40, -60])

plt.ylabel('Gain [dB]')

plt.title('Frequency Response')

plt.subplot(2, 1, 2)

plt.plot(w/np.pi, np.angle(h))

plt.grid(True)

plt.yticks([-np.pi, -0.5\*np.pi, 0, 0.5\*np.pi, np.pi],

           [r'$-\pi$', r'$-\pi/2$', '0', r'$\pi/2$', r'$\pi$'])

plt.ylabel('Phase [rad]')

plt.xlabel('Normalized frequency (1.0 = Nyquist)')

plt.show()

samplerate, data = wavfile.read('./zx25r.wav')

filtered = signal.sosfilt(sos, mono\_data)

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

sos = signal.butter(2, [15000, 17000], 'bp', fs=samplerate, output='sos')

filtered = signal.sosfilt(sos, mono\_data)

N = len(filtered)

T = 1.0 / samplerate

fft\_data = np.fft.fft(filtered)

xf = np.fft.fftfreq(N, T)[:N//2]

amplitude = np.abs(fft\_data[N//2:])/N \* np.abs(fft\_data[0:N//2])

plt.figure(figsize=(12, 6))

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

samplerate, data = wavfile.read('./zx25r.wav')

mono\_data = data[:, 0]

sos = signal.butter(1, [5000, 10000], 'bp', fs=samplerate, output='sos')

filtered = signal.sosfilt(sos, mono\_data)

volume\_factor = 5.0

increased\_volume = filtered \* volume\_factor

wavfile.write('output.wav', samplerate, increased\_volume.astype(np.int16))

IPython.display.Audio('./output.wav')

1. **600 cc (ZX6R)**

from scipy.io import wavfile

import IPython

import matplotlib.pyplot as plt

import sounddevice as sd

samplerate2, data2 = wavfile.read('./zx6r.wav')

IPython.display.Audio('./zx6r.wav')

mono\_data2 = data2[:, 0]

plt.plot(mono\_data2)

plt.show()

from scipy import io, signal, fft

import numpy as np

N2 = len(data2)

T2 = 1. / samplerate2

samplerate2, data2 = wavfile.read('./zx6r.wav')

# Mengambil saluran pertama untuk data mono

mono\_data2 = data2[:, 0]

# Panjang sinyal dan interval waktu

N2 = len(mono\_data2)

T2 = 1.0 / samplerate2

# Melakukan Transformasi Fourier Cepat (FFT)

fft\_data2 = fft.fft(mono\_data2)

# Menghasilkan array frekuensi positif

xf2 = fft.fftfreq(N2, T2)[:N2//2]

# Menghitung amplitudo spektrum frekuensi

amplitude2 = 2.0/N2 \* np.abs(fft\_data2[0:N2//2])

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot spektrum frekuensi

plt.plot(xf2, amplitude2)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

# Menentukan batas sumbu x dan y untuk zoom

plt.xlim(0, 1000)  # Batas sumbu x dari 0 hingga 1000 Hz

plt.ylim(0, 2000)  # Batas sumbu y

plt.show()

samplerate, data = wavfile.read('./zx6r.wav')

mono\_data = data[:, 0]

N = len(mono\_data)

T = 1.0 / samplerate

fft\_data = fft.fft(mono\_data)

xf = fft.fftfreq(N, T)[:N//2]

amplitude = np.abs(fft\_data[N//2:])/N \* np.abs(fft\_data[0:N//2])

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot spektrum frekuensi

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

samplerate2, data2 = wavfile.read('./zx6r.wav')

# Mengambil saluran pertama untuk data mono

mono\_data2 = data2[:, 0]

# Desain filter Butterworth analog

b, a = signal.butter(1, [4000, 7000], btype='bandpass', analog=True)

# Menghitung respons frekuensi filter

w, h = signal.freqs(b, a)

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot respons frekuensi

plt.semilogx(w, 20 \* np.log10(abs(h)))

# Menandai batas-batas filter

plt.axvline(0.67, color='green')

plt.axvline(40, color='green')

plt.axhline(-3, color='green')

# Menambahkan grid

plt.grid(which='both', axis='both')

# Menentukan batas sumbu y

plt.ylim((-30, 0))

plt.xlabel('Frequency (Hz)')

plt.ylabel('Magnitude (dB)')

plt.show()

samplerate2, data2 = wavfile.read('./zx6r.wav')

# Mengambil saluran pertama untuk data mono

mono\_data2 = data2[:, 0]

# Desain filter Butterworth high-pass

sos = signal.butter(1, 18000, 'hp', fs=samplerate2, output='sos')

# Menghitung respons frekuensi filter

w, h = signal.sosfreqz(sos)

# Menentukan ukuran gambar

plt.figure(figsize=(12, 8))

# Memplot respons frekuensi (Gain)

plt.subplot(2, 1, 1)

db = 20\*np.log10(np.maximum(np.abs(h), 1e-5))

plt.plot(w/np.pi, db)

plt.ylim(-75, 5)

plt.grid(True)

plt.yticks([0, -20, -40, -60])

plt.ylabel('Gain [dB]')

plt.title('Frequency Response')

# Memplot respons frekuensi (Phase)

plt.subplot(2, 1, 2)

plt.plot(w/np.pi, np.angle(h))

plt.grid(True)

plt.yticks([-np.pi, -0.5\*np.pi, 0, 0.5\*np.pi, np.pi],

           [r'$-\pi$', r'$-\pi/2$', '0', r'$\pi/2$', r'$\pi$'])

plt.ylabel('Phase [rad]')

plt.xlabel('Normalized frequency (1.0 = Nyquist)')

plt.show()

samplerate2, data2 = wavfile.read('./zx6r.wav')

filtered = signal.sosfilt(sos, mono\_data2)

samplerate2, data2 = wavfile.read('./zx6r.wav')

# Mengambil saluran pertama untuk data mono

mono\_dat2 = data2[:, 0]

# Desain filter Butterworth band-pass

sos = signal.butter(2, [15000, 17000], 'bp', fs=samplerate2, output='sos')

# Mengaplikasikan filter SOS ke data mono

filtered = signal.sosfilt(sos, mono\_data)

# Panjang sinyal dan interval waktu

N = len(filtered)

T = 1.0 / samplerate2

# Melakukan transformasi Fourier cepat (FFT)

fft\_data = np.fft.fft(filtered)

# Menghasilkan array frekuensi positif

xf = np.fft.fftfreq(N, T)[:N//2]

# Menghitung amplitudo spektrum frekuensi

amplitude = np.abs(fft\_data[N//2:])/N \* np.abs(fft\_data[0:N//2])

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot spektrum frekuensi

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

# Membaca file audio

samplerate2, data2 = wavfile.read('./zx6r.wav')

# Mengambil saluran pertama untuk data mono

mono\_data2 = data2[:, 0]

# Desain filter Butterworth band-pass

sos = signal.butter(1, [3000, 10000], 'bp', fs=samplerate2, output='sos')

# Mengaplikasikan filter SOS ke data mono

filtered = signal.sosfilt(sos, mono\_data2)

# Peningkatan volume

volume\_factor = 6

increased\_volume = filtered \* volume\_factor

# Menyimpan data mono yang sudah difilter dan diproses ke file output.wav

wavfile.write('output.wav', samplerate, increased\_volume.astype(np.int16))

IPython.display.Audio('./output.wav')

1. **1000cc (ZX10R)**

from scipy.io import wavfile

import IPython

import matplotlib.pyplot as plt

import sounddevice as sd

samplerate3, data3 = wavfile.read('./ZX10R.wav')

IPython.display.Audio('./ZX10R.wav')

mono\_data3 = data3[:, 0]

plt.plot(mono\_data3)

plt.show()

from scipy import io, signal, fft

import numpy as np

N3 = len(data3)

T3 = 1. / samplerate3

samplerate3, data3 = wavfile.read('./ZX10R.wav')

# Mengambil saluran pertama untuk data mono

mono\_data3 = data3[:, 0]

# Panjang sinyal dan interval waktu

N3 = len(mono\_data3)

T3 = 1.0 / samplerate3

# Melakukan Transformasi Fourier Cepat (FFT)

fft\_data3 = fft.fft(mono\_data3)

# Menghasilkan array frekuensi positif

xf3 = fft.fftfreq(N3, T3)[:N3//2]

# Menghitung amplitudo spektrum frekuensi

amplitude3 = 2.0/N3 \* np.abs(fft\_data3[0:N3//2])

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot spektrum frekuensi

plt.plot(xf3, amplitude3)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

# Menentukan batas sumbu x dan y untuk zoom

plt.xlim(0, 1000)  # Batas sumbu x dari 0 hingga 1000 Hz

plt.ylim(0, 2000)  # Batas sumbu y

plt.show()

samplerate, data = wavfile.read('./ZX10R.wav')

mono\_data = data[:, 0]

N = len(mono\_data)

T = 1.0 / samplerate

fft\_data = fft.fft(mono\_data)

xf = fft.fftfreq(N, T)[:N//2]

amplitude = np.abs(fft\_data[N//2:])/N \* np.abs(fft\_data[0:N//2])

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot spektrum frekuensi

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

samplerate3, data3 = wavfile.read('./zx10r.wav')

# Mengambil saluran pertama untuk data mono

mono\_data3 = data3[:, 0]

# Desain filter Butterworth analog

b, a = signal.butter(1, [4000, 7000], btype='bandpass', analog=True)

# Menghitung respons frekuensi filter

w, h = signal.freqs(b, a)

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot respons frekuensi

plt.semilogx(w, 20 \* np.log10(abs(h)))

# Menandai batas-batas filter

plt.axvline(0.67, color='green')

plt.axvline(40, color='green')

plt.axhline(-3, color='green')

# Menambahkan grid

plt.grid(which='both', axis='both')

# Menentukan batas sumbu y

plt.ylim((-30, 0))

plt.xlabel('Frequency (Hz)')

plt.ylabel('Magnitude (dB)')

plt.show()

samplerate3, data3 = wavfile.read('./zx10r.wav')

# Mengambil saluran pertama untuk data mono

mono\_data3= data3[:, 0]

# Desain filter Butterworth high-pass

sos = signal.butter(1, 18000, 'hp', fs=samplerate3, output='sos')

# Menghitung respons frekuensi filter

w, h = signal.sosfreqz(sos)

# Menentukan ukuran gambar

plt.figure(figsize=(12, 8))

# Memplot respons frekuensi (Gain)

plt.subplot(2, 1, 1)

db = 20\*np.log10(np.maximum(np.abs(h), 1e-5))

plt.plot(w/np.pi, db)

plt.ylim(-75, 5)

plt.grid(True)

plt.yticks([0, -20, -40, -60])

plt.ylabel('Gain [dB]')

plt.title('Frequency Response')

# Memplot respons frekuensi (Phase)

plt.subplot(2, 1, 2)

plt.plot(w/np.pi, np.angle(h))

plt.grid(True)

plt.yticks([-np.pi, -0.5\*np.pi, 0, 0.5\*np.pi, np.pi],

           [r'$-\pi$', r'$-\pi/2$', '0', r'$\pi/2$', r'$\pi$'])

plt.ylabel('Phase [rad]')

plt.xlabel('Normalized frequency (1.0 = Nyquist)')

plt.show()

samplerate3, data3 = wavfile.read('./zx10r.wav')

filtered = signal.sosfilt(sos, mono\_data3)

samplerate3, data3 = wavfile.read('./zx10r.wav')

# Mengambil saluran pertama untuk data mono

mono\_dat2 = data3[:, 0]

# Desain filter Butterworth band-pass

sos = signal.butter(2, [15000, 17000], 'bp', fs=samplerate3, output='sos')

# Mengaplikasikan filter SOS ke data mono

filtered = signal.sosfilt(sos, mono\_data)

# Panjang sinyal dan interval waktu

N = len(filtered)

T = 1.0 / samplerate3

# Melakukan transformasi Fourier cepat (FFT)

fft\_data = np.fft.fft(filtered)

# Menghasilkan array frekuensi positif

xf = np.fft.fftfreq(N, T)[:N//2]

# Menghitung amplitudo spektrum frekuensi

amplitude = np.abs(fft\_data[N//2:])/N \* np.abs(fft\_data[0:N//2])

# Menentukan ukuran gambar

plt.figure(figsize=(12, 6))

# Memplot spektrum frekuensi

plt.plot(xf, amplitude)

plt.xlabel('Frequency (Hz)')

plt.ylabel('Amplitude')

plt.show()

# Membaca file audio

samplerate3, data3 = wavfile.read('./zx10r.wav')

# Mengambil saluran pertama untuk data mono

mono\_data3 = data3[:, 0]

# Desain filter Butterworth band-pass

sos = signal.butter(1, [3000, 10000], 'bp', fs=samplerate3, output='sos')

# Mengaplikasikan filter SOS ke data mono

filtered = signal.sosfilt(sos, mono\_data3)

# Peningkatan volume

volume\_factor = 3.0

increased\_volume = filtered \* volume\_factor

# Menyimpan data mono yang sudah difilter dan diproses ke file output.wav

wavfile.write('output.wav', samplerate, increased\_volume.astype(np.int16))

IPython.display.Audio('./output.wav')

**Penjelasan Seluruh Kode :**

Kode di atas adalah contoh implementasi pengolahan sinyal audio menggunakan pustaka-pustaka Python. Dalam pengolahan sinyal audio, terdapat beberapa tahapan yang umum dilakukan, seperti pembacaan file audio, visualisasi sinyal, analisis spektrum frekuensi, dan penerapan filter.

Pertama, kode membaca file audio menggunakan pustaka scipy.io dan memainkannya menggunakan IPython.display.Audio(). Langkah ini penting untuk memastikan file audio dapat diakses dan didengar dengan benar sebelum dilakukan pemrosesan lebih lanjut.

Selanjutnya, sinyal audio mono diekstraksi dari sinyal stereo dan ditampilkan dalam bentuk plot menggunakan matplotlib.pyplot.plot(). Visualisasi ini membantu memahami bentuk gelombang sinyal audio secara visual.

Kemudian, dilakukan analisis frekuensi menggunakan transformasi Fourier menggunakan pustaka numpy.fft. Dengan transformasi Fourier, sinyal audio diubah dari domain waktu menjadi domain frekuensi. Hasilnya, dapat dilihat spektrum frekuensi sinyal audio melalui plot yang menunjukkan amplitudo versus frekuensi. Informasi ini berguna untuk menganalisis komponen frekuensi yang ada dalam sinyal audio.

Selanjutnya, dilakukan penerapan filter Butterworth menggunakan pustaka scipy.signal. Dalam contoh kode tersebut, terdapat penggunaan filter band-pass dan high-pass untuk menunjukkan bagaimana filter dapat digunakan untuk memodifikasi sinyal audio dengan membatasi rentang frekuensi tertentu. Filter ini dapat digunakan untuk menghilangkan atau mempertajam komponen frekuensi tertentu dalam sinyal audio.

Pada akhirnya, sinyal audio hasil filtering juga dapat ditingkatkan volumenya dan disimpan sebagai file audio baru menggunakan wavfile.write(). Hal ini memungkinkan pengguna untuk menghasilkan sinyal audio yang telah diproses sesuai dengan kebutuhan mereka.

**BAB 3**

**KESIMPULAN**

**3.1 Kesimpulan**

Kesimpulan yang didapat dari pembuatan program untuk memfilter suara untuk 3 jenis variasi motor 4 cylinder berdasarkan cc mesin ini adalah dapat diketahui bahwa frekuensi dari setiap exhaust mesin memiliki karakteristik yang berbeda beda. Setelah difiilter frekuensi yang didapat akan berbeda dengan frekuensi awal dari setiap exhaust mesin. Program tersebut membantu dalam memfilter suara motor dengan kapasitas mesin yang berbeda menggunakan teknik pemrosesan sinyal audio dan filter Butterworth.

**LAMPIRAN**

Link GitHub : <https://github.com/JamesVSeVERYBODY/Filter>