



Designnotat

Tittel: Op-AMP

Forfattere: Eirik Mathias Silnes

Versjon: 1.0

Dato: 15. desember 2023

Innhold

1	Problembeskrivelse	2
2	Prinsipiell løsning	3
2.1	Kretstopologi	3
2.2	Forsterkning	3
2.3	Total harmonisk distorsjon	4
2.4	Åpen Løkke forsterkning	4
3	Realisering	5
3.1	Kretsopkobling	5
3.2	Målinger	6
3.3	Diskusjon av måleresultater	6
3.4	Forbedringer	7
4	Konklusjon	8
5	Takk	8
	Referanser	9
A	Python script for utregning	10

1 Problembeskrivelse

I dette designnotatet skal det designes en operasjonsforsterker på transistor nivå. En ideell operasjonsforsterker har følgende egenskaper og modell som vist i fig Figur 1.

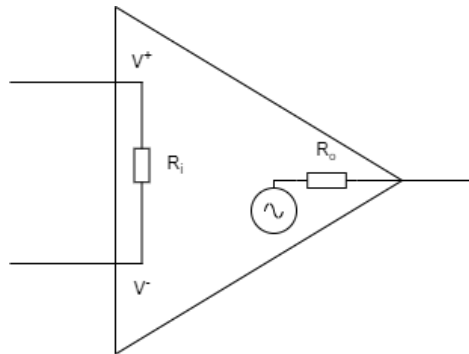
- Inngangsimpedansen til $R_i = \infty$
- Utgangsimpedansen til $R_o = 0$
- Utgangen er gitt som

$$V_{out} = f(V_+ - V_-) = \begin{cases} \min\{V, A(v^+ - v^-)\} & \text{for } v^+ - v^- > 0 \\ \max\{V, A(v^+ - v^-)\} & \text{for } v^+ - v^- < 0 \end{cases} \quad (1)$$

Spesielt i dette designnotatet skal de følgende egenskapene undersøkest nærmere:

- forsterkningen A ved sinuspåtrykk med frkvens $f = 1kHz$ og
- Total harmonisk distorsjon (THD) ved sinuspåtrykk med frekvens $f = 1kHz$

De to punktene skal undersøkest med to forskjellige lastmotstander $R_L = 100k\Omega$ og $R_L = 100\Omega$. Det skal også undersøkest hvor godt kretsløsningen virker som en opamp i en inverterende forsterker med forsterkning $A = -10$ og $R_L = 1k\Omega$. Sammenlign dette med ved både åpen løkke forsterkning og negativ tilbakekobling.

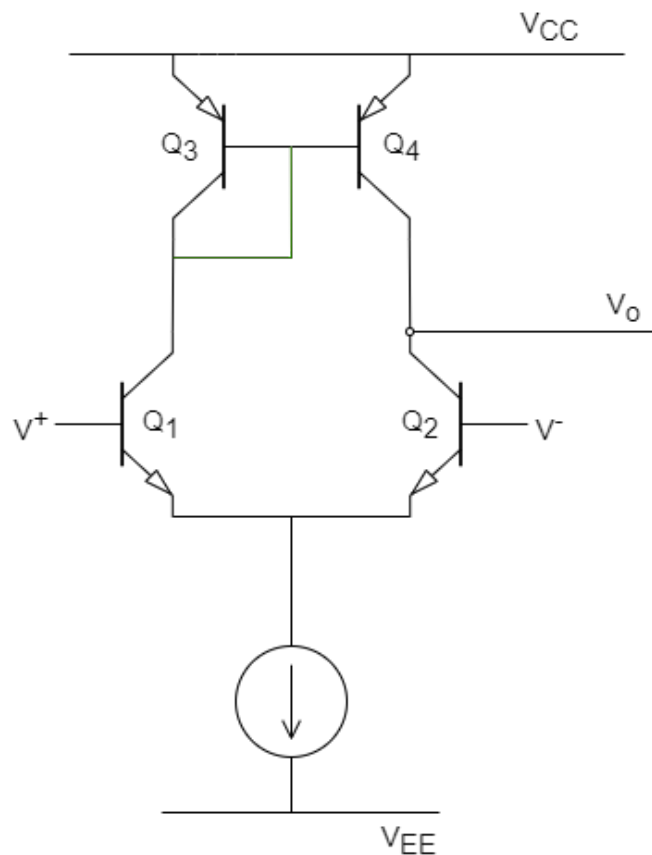


Figur 1: Ideell opamp modell

2 Prinsipiell løsning

2.1 Kretstopologi

En differensialforsterker er en nøkkelkomponent i designet av operasjonsforsterkere (op-amp)[2, s. 105]. I en differensialforsterker benyttes ofte to inngangstransistorer i en konfigurasjon som tillater differensiell signalbehandling. Bipolare transistorer, som for eksempel NPN- og PNP-transistorer, er valgt for deres egenskaper som forsterkningsenheter og deres evne til å drive signaler med høy presisjon. Et typisk eksempel på en differensialforsterker er vist i figur 2.



Figur 2: Differensialforsterker som en operasjonsforsterker

2.2 Forsterkning

For å finne forsterkningen til en op-amp, kan man gå utifra Ligning 1 og omformulere den til Ligning 2. Her er V^+ og V^- spenningen inn på den positive og negative inngangen til opampen (Differnansen mellom de vil bli omtalt som V_{in}). V_o er spenningen ut av opampen

over lasten og da blir A forsterkningen til opampen.

$$A = \frac{V_o}{(V^+ - V^-)} = \frac{V_o}{V_{in}} \quad (2)$$

2.3 Total harmonisk distorsjon

Total harmonisk distorsjon er hvor mye av det originale signalet dominerer over harmoniske multiplikative versjoner av seg selv, og er en vanlig måleenhet for støymåling i et ikke linjeært system. Man benytter seg av V_{RMS} på både den originale frekvensen og på alle harmoniske frekvenser. For å finne total harmonisk distorsjon, kan vi bruke Ligning 3. Her er V_1 er spenningen ut av opampen ved en sinusformet inngangsspenning og V_n RMS verdien til den n 'te harmoniske frekvensen.

$$THD(\%) = \frac{\sqrt{V_2^2 + V_3^2 + V_4^2 + \dots + V_n^2}}{V_1} \cdot 100 \quad (3)$$

velg en av de to under, sjekk formelen

2.4 Åpen Løkke forsterkning

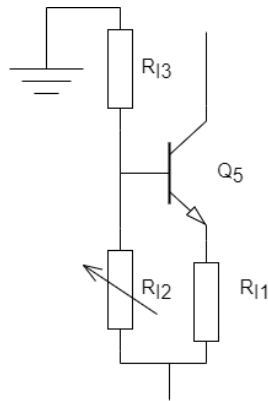
For å beregne åpen løkke forsterkning, kan følgende ligning Ligning 4 benyttes. Her representerer V_{out} utgangsspenningen fra opampen. V_{in} er inngangsspenningen til opampen, og A er forsterkningen til opampen. Åpen løkke betyr at det ikke er noen tilbakekobling i kretsen [?].

$$V_{out} = AV_{in} \quad (4)$$

3 Realisering

3.1 Kretsoptkobling

Oppkoblingen ble gjort som vist i den prinsipielle løsningen i Figur 2, hvor strømkliden ble realisert som i Figur 3. Hvor Q_5 og R_{I1} fungerer som en strømkilde som er styrt av spenningen som er mellom R_{I2} og R_{I3} , hvor R_{I2} ble realisert som et potensiometer for å lettere kunne justere strømstyrken underveis. Modellnummer og komponent verdiene som ble brukt i oppkoblingen vises i Tabell 1.

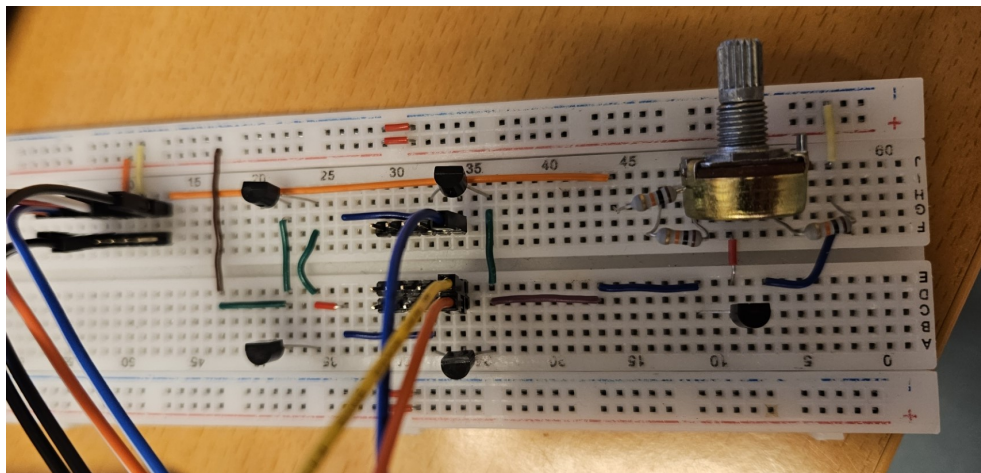


Tabell 1: Komponenter og verdier brukt i designet.

Komponent	Verdi/Produktnummer
$Q_1 \& Q_2 \& Q_5$	BC547A (NPN)
$Q_3 \& Q_4$	BC557B (PNP)
R_{I1}	3k Ω
R_{I2}	0 Ω – 10k Ω
R_{I3}	10k Ω

Figur 3: Realisert strømkilde.

Den oppkoblede kretsen vises i Figur 4.



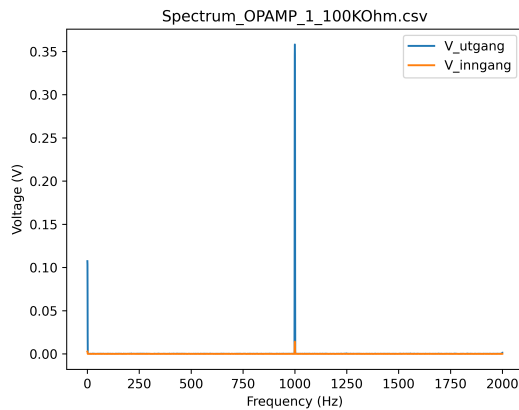
Figur 4: Realisert operasjonsforsterker

3.2 Målinger

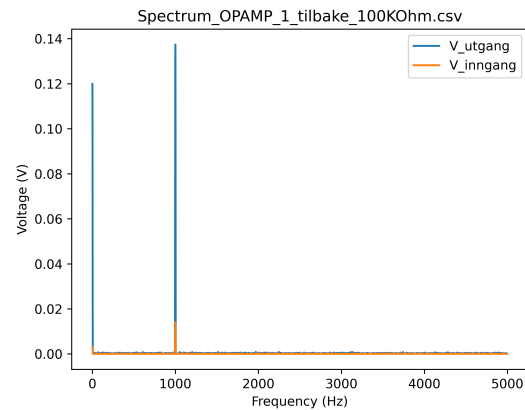
Det ble gjort målinger med oscilloskop og spektrumsanalysator, deretter ble målingene prosessert i python for å finne THD og forsterkning. Målingene ble gjort med en inngangsamplitude på 0.04V og en frekvens på 1kHz. Resultatene av målingene vises i tabell Tabell 2. Ut av tabellen kan vi se at forsterkningen varierer mye når vi ikke har tilbakekobling på utgangen og når vi setter på en lav lastmotstand så synker forsterkningen kraftig. Spektrumsanalysen av utgagnssignalet med last på $R_L = 100k\Omega$ både med og uten tilbakekobling vises i figur Figur 6 og i figur Figur 5.

Konfigurasjon	Amplitude inn	Amplitude ut	Forsterkning	THD
$R_L = 0$	0.04V	0.90V	20	1.38%
$R_L = 100k\Omega$	0.04V	0.99V	22	0.78%
$R_L = 100\Omega$	0.04V	0.72V	16	10.67%
Tilbakekobling $R_L = 0$	0.04V	0.42V	9.7	1.33%
Tilbakekobling $R_L = 100k\Omega$	0.04V	0.42V	9.4	0.41%
Tilbakekobling $R_L = 100\Omega$	0.04V	0.21V	4.7	0.54%

Tabell 2: THD and Amplitude Data



Figur 5: Spectrum med $R_L = 100k\Omega$ åpen løkke

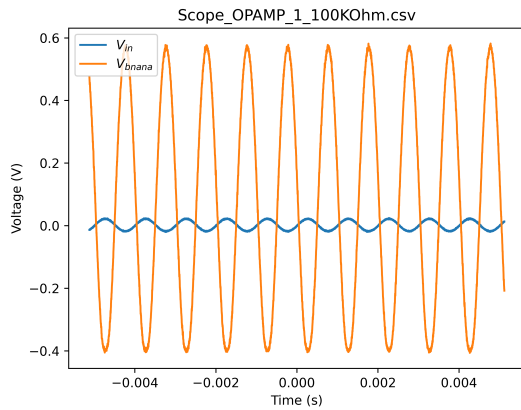


Figur 6: Spectrum med $R_L = 100k\Omega$ tilbakekobling

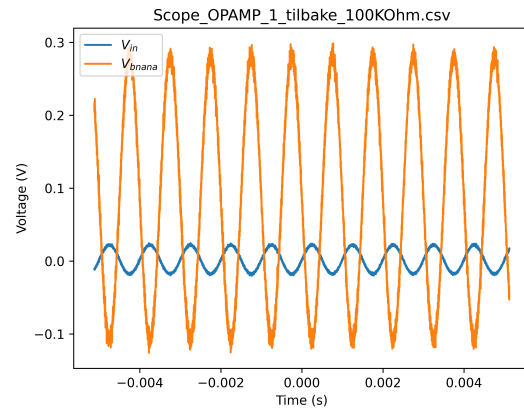
Som vi ser i figur Figur 6 så er det mye distorsjon ved 0Hz. Dette er fordi vi har en DC offset på utgangen. Dette kan vi fjerne ved å sette inn en kondensator i tilbakekoblingen.

3.3 Diskusjon av måleresultater

Det som står mest ut blant målingene er de store frekvenskomponentene rett ved 0Hz i spektrumsanalysen. Dette er fordi vi har en DC offset på utgangen som forskyver signalene i forhold til hverandre noe som man enkelt kan se ved å se på oscilloskopsdataen som vist i Figur 7 og Figur 8. Ettersom dette har en veldig stor påvirkning på THD uten å egentlig bidra med noe støy så kan vi beregne THD uten å ta hensyn til DC offseten som vist i Tabell 2. Dette gjør



Figur 7: Ocilloskopsdata

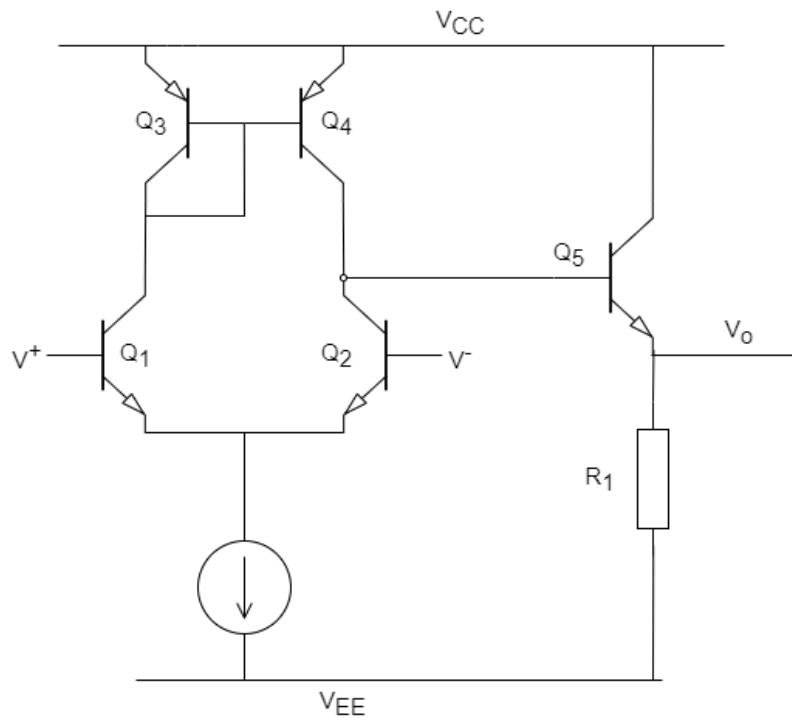


Figur 8: Ocilloskopsdata med tilbakekobling

vi ved å bruke Ligning 3 på dataen fra spektrumsanalysen ved hjelp av python koden under. Hadde man tatt med signalene på 0Hz så hadde THD blitt i snitt 200% i motsetning til det som er vist i Tabell 2.

3.4 Forbedringer

Denne kretsen kunne blitt bedre hadde man koblet opp en emitter følger ved utgangen, slik at kretsen hadde blitt seende ut som i Figur 9. Denne endringen ville også sansyneligvis redusert THD ved å fjerne DC offseten på utgangen i tillegg til å sentrere utgangssignalet rundt 0V.



Figur 9: Forbedret differensialforsterker

4 Konklusjon

Kretsen som var koblet opp var en differensialforsterker som ble brukt som en operasjonsforsterker. Total Harmonisk Distorsjon var litt høyere enn forventet på grunn av DC oppførselen til kretsen. Forsterkninkgen varierer mye med lastmotstand, og det er derfor viktig å ha en emitterfølger ved utgangen. Forsterkninkgen blir mer stabli med tilbakekobling frem til man trekker for mye strøm fra kretsen.

5 Takk

Referanser

- [1] L. Lundheim, *Designprosjekt 6*, Institutt for elektronisk systemdesign NTNU 2023.
- [2] P. Horowitz, W. Hill, *The Art of Electronics*, Cambridge University Press, 3. utgave, 2016.
- [3] ukjent, *Inverting Amplifiers*, Eletronics-Tutorials.ws, https://www.electronics-tutorials.ws/opamp/opamp_2.html, 03/10-23.

A Python script for utregning

#find all files in current directory that start with 'spectrum' and is a .csv file

```
import os
import glob
import numpy as np
import matplotlib.pyplot as plt
import argparse
```

```
parser = argparse.ArgumentParser(description='A script with command-line arguments.')
parser.add_argument('-d', '--dir', type=str, help='The directory to save the plot to. Defaults to current directory')
parser.add_argument('-f', '--file', type=str, help='The file to plot. Defaults to all files in directory')
parser.add_argument('-ph', '--Phase', action='store_true', help='Plot phase as well')
parser.add_argument('-ch_1', '--channel_1', type=str, help='Name of Channel 1, defaults to channel_1')
parser.add_argument('-ch_2', '--channel_2', type=str, help='Name of Channel 2, defaults to channel_2')
parser.add_argument('-thd', '--THD', action='store_true', help='Plot THD as well')
```

```
def find_spectrum_files():
    spectrum_files = []
    for file in glob.glob("spectrum*.csv"):
        spectrum_files.append(file)
    return spectrum_files
```

#get the data from the spectrum files and generate a plot for each file

```
def get_spectrum_data(spectrum_files):
    data_dict = {}
    for file in spectrum_files:
        data = np.loadtxt(file, delimiter=',', skiprows=1)
        #round first column to 0 decimals
        data[:,0] = np.round(data[:,0], 0)
        data_dict[file] = data
    return data_dict
```

#save the data from the spectrum as a bodeplot

```
def save_spectrum_plot(data_dict, args):
    saveDir = args.dir

    for key in data_dict:
        mag1 = np.abs(data_dict[key][:,1])
        mag2 = np.abs(data_dict[key][:,3])
        if np.max(mag1) > np.max(mag2):
            strongest = data_dict[key][:,1]
            weakest = data_dict[key][:,3]
```

```

else:
    strongest = data_dict[key][:,3]
    weakest = data_dict[key][:,1]

if args.Phase:
    #generate two subplots one for voltage and one for phase
    fig, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
    #plot the voltage
    ax1.plot(data_dict[key][:,0], strongest)
    ax1.plot(data_dict[key][:,0], weakest)
    ax1.set_ylabel('Voltage (V)')
    ax1.set_title(key)
    #plot the phase
    ax2.plot(data_dict[key][:,0], data_dict[key][:,2])
    ax2.plot(data_dict[key][:,0], data_dict[key][:,5])
    ax2.set_ylabel('Phase (deg)')
    ax2.set_xlabel('Frequency (Hz)')
    #save the plot
else:
    plt.plot(data_dict[key][:,0], strongest)
    plt.plot(data_dict[key][:,0], weakest)
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Voltage (V)')
    plt.title(key)
    plt.legend([args.channel_1, args.channel_2])
    #plt.show()

print('Saving plot to:')
print(os.path.join(saveDir, key[:-4] + '.png'))
savePath = os.getcwd() + os.path.join(saveDir, key[:-4] + '.png')
#print(savePath)
plt.savefig(savePath, dpi = 600)
plt.clf()

def calculate_THD(data_dict, freq):
    #calculate Total Harmonic Distortion
    for key in data_dict:
        #find a given frequency
        #find to div4 and mult4 because of limited bandwidth
        #frequency = 1000
        index_1000 = np.where(data_dict[key][:,0] == freq)
        freq_val1 = data_dict[key][index_1000[0],1]
        freq_val2 = data_dict[key][index_1000[0],3]

```

```

index_div2 = np.where(data_dict[key][:,0] == freq/2)
freq_val1_div2 = data_dict[key][index_div2[0],1]
freq_val2_div2 = data_dict[key][index_div2[0],3]

index_div4 = np.where(data_dict[key][:,0] == freq/4)
freq_val1_div4 = data_dict[key][index_div4[0],1]
freq_val2_div4 = data_dict[key][index_div4[0],3]

index_mult2 = np.where(data_dict[key][:,0] == freq*2)
freq_val1_mult2 = data_dict[key][index_mult2[0],1]
freq_val2_mult2 = data_dict[key][index_mult2[0],3]

index_mult3 = np.where(data_dict[key][:,0] == freq*3)
freq_val1_mult3 = data_dict[key][index_mult3[0],1]
freq_val2_mult3 = data_dict[key][index_mult3[0],3]

index_mult4 = np.where(data_dict[key][:,0] == freq*4)
freq_val1_mult4 = data_dict[key][index_mult4[0],1]
freq_val2_mult4 = data_dict[key][index_mult4[0],3]

zero = data_dict[key][0,3]
#sum of Val 2
sum_val2 = np.sum(freq_val2_div2)**2 + np.sum(freq_val2_div4)**2 + np.sum(freq_val2_mult2)**2 + np.sum(freq_val2_mult3)**2 + np.sum(freq_val2_mult4)**2

thd = np.sqrt(sum_val2)/freq_val2
thd2 = np.sqrt(sum_val2+zero)/freq_val2
print(f'THD for {key} is {round(thd[0]*100, 2)}% or {round(thd2[0]*100, 2)}%')

#print(f'Index of {freq} is {index_1000} and has value {freq_val1} and {freq_val2}')

#main function
def main():
    args = parser.parse_args()

    #Command-line arguments
    if args.file:
        spectrum_files = [args.file]

    #Get the spectrum files
    if not args.file:
        spectrum_files = find_spectrum_files()

    ##Collect data from the spectrum files

```

```
data_dict = get_spectrum_data(spectrum_files)

if args.THD:
    calculate_THD(data_dict, 1000)
    return

save_spectrum_plot(data_dict, args)

if __name__ == '__main__':
    main()
```