

LAPORAN PRAKTIKUM

SAMPLING, FRAME BLOCKING DAN KONVOLUSI

Muhammad Fadhil Syahputra, Ismail Ragi Alfarugi .

Universitas Lambung Mangkurat

Email korespondensi : syahputraf112@gmail.com

PENDAHULUAN

Dalam kehidupan sehari-hari, sinyal analog banyak dijumpai dalam berbagai bentuk, seperti suara manusia, musik, atau getaran mekanik. Sinyal analog merupakan sinyal kontinu yang membawa informasi dalam bentuk perubahan nilai amplitudo terhadap waktu. Untuk keperluan pemrosesan sinyal secara elektronik, sinyal analog sering kali mengalami pengolahan awal melalui proses Analog Signal Processing (ASP), seperti penguatan, penyaringan (filtering), atau modulasi, dengan output yang tetap dalam bentuk sinyal analog.

Seiring perkembangan teknologi digital, pengolahan sinyal kini semakin banyak dilakukan dalam ranah digital. Pengolahan sinyal digital memiliki keunggulan dari sisi fleksibilitas, ketahanan terhadap noise, dan kemudahan integrasi dengan sistem komputer. Namun, sistem pengolahan sinyal digital hanya dapat bekerja dengan sinyal dalam bentuk digital, sehingga sinyal analog perlu terlebih dahulu dikonversi melalui proses Analog-to-Digital Conversion (ADC). Proses ini mencakup tiga tahap utama: sampling, quantizing, dan coding. Sebaliknya, untuk menghasilkan keluaran dalam bentuk analog kembali, sistem digital membutuhkan proses Digital-to-Analog Conversion (DAC).

Dalam praktikum ini, pembahasan difokuskan pada tahap awal dari proses digitalisasi sinyal, yaitu sampling, beserta dua konsep penting lain yang umum digunakan dalam pengolahan sinyal digital, yaitu frame blocking dan konvolusi. Pemahaman terhadap ketiga konsep ini merupakan dasar penting dalam dunia Digital Signal Processing (DSP), terutama dalam aplikasi seperti pengenalan suara, analisis frekuensi, dan pemrosesan audio digital. Adapun konsep quantizing dan coding akan menjadi materi lanjutan yang dapat dipelajari secara mandiri, terutama bagi

yang tertarik mendalami bidang komunikasi digital atau kompresi sinyal suara.

TINJAUAN PUSTAKA

Frekuensi sampling secara visual merujuk pada penentuan seberapa sering sinyal analog diamati atau dicatat dalam proses pengolahan. Visualisasi frekuensi sampling membantu dalam mengidentifikasi kemungkinan distorsi seperti aliasing yang terjadi akibat laju sampling yang tidak memadai. Sinyal analog diolah dalam ASP (Analog Signal Processing). Sinyal input analog ini diperlukan untuk proses digitalisasi di ADC (Analog to Digital Conversion) melalui beberapa proses seperti sampling (pengambilan cuplikan data), quantizing (perubahan sinyal sample dari sifat continuous menjadi nilai bersifat diskrit), dan coding atau pemrograman (Khairunnisa, 2019). Syarat dalam proses sampling adalah Nyquist berupa $f_s > 2 \cdot f_i$. Keterangan f_s sebagai frekuensi sinyal sampling sedangkan f_i adalah frekuensi sinyal informasi sampling (Astomo et al., 2023).

Sinyal suara (audio) diolah dari rangkaian proses untuk menghasilkan informasi terkait durasi atau waktu bunyi dengan bentuk grafik berisikan sinyal suara dan spektrumnya. Proses ini diawali dengan menginput suara oleh pemrogram melalui mikrofon pada laptop, kemudian sinyal suara diproses oleh sistem ADC agar sesuai dengan frekuensi sampling, dikuantisasi, dan dikonversi menjadi kode biner. DAC menjadi tempat untuk memasukan kode biner sehingga ada kesesuaian di hasil karakteristik sinyal input yang dihasilkan oleh sinyal analog. Alat akurat seperti kalkulator matriks interaktif yang sederhana dibangun dari pustaka perangkat lunak matriks LINPACK dan EISPACK untuk analisis dan visualisasi data adalah MATLAB (Matrix Laboratory). Fungsi MATLAB adalah simulasi proses sampling,

memvisualisasikan sinyal asli, dan menganalisis efek aliasing. Hasil efek dari proses sampling memiliki frekuensi yang berbeda dari sinyal sebenarnya disebut dengan aliasing (Shabrina et al., 2024).

Frame blocking adalah proses untuk memotong atau membagi data dari sinyal suara yang lebih panjang menjadi sinyal suara pendek dalam beberapa frame dengan durasi 20 hingga 30 ms yang berisi N sampel. Fungsi proses ini adalah untuk memudahkan perhitungan karena keadaan sinyal suara umumnya konstan berubah. Tahap overlapping terjadi di proses ini untuk mencegah hilangnya karakteristik suara asli (Lazuardhy & Prasetyo, 2022).

Operasi matematis yang melibatkan gabungan 2 fungsi atau sinyal menjadi satu kesatuan sinyal baru disebut dengan konvolusi. Konvolusi dua sinyal sinus

METODE PENELITIAN

Alat dan Bahan

1. PC atau Laptop
2. Python 3.x
3. Pustaka python

Dalam praktikum ini, kita mempelajari tiga konsep penting dalam pengolahan sinyal digital, yaitu sampling, frame blocking, dan konvolusi sinyal. Pertama, terkait sampling, peningkatan frekuensi sampling dari 1 kHz ke 2 kHz menyebabkan sinyal hasil sampling menjadi lebih mendekati sinyal aslinya. Hal ini terjadi karena frekuensi sampling yang lebih tinggi menghasilkan lebih banyak titik data dalam satu detik, sehingga bentuk gelombang yang dihasilkan setelah interpolasi menjadi lebih halus dan menyerupai sinyal kontinu. Sebaliknya, jika frekuensi sampling terlalu rendah (kurang dari dua kali frekuensi maksimum sinyal, melanggar kriteria Nyquist), akan terjadi aliasing, yaitu fenomena di mana frekuensi tinggi dalam sinyal asli salah diinterpretasikan sebagai frekuensi rendah dalam sinyal hasil sampling. Akibatnya, bentuk sinyal menjadi terdistorsi dan tidak merepresentasikan sinyal aslinya.

Selanjutnya, pada frame blocking, sinyal dibagi menjadi potongan-potongan kecil yang disebut frame, yang biasanya saling tumpang tindih. Tujuannya adalah untuk menangkap dinamika lokal sinyal yang dapat berubah seiring waktu, terutama pada sinyal

dihasilkan dari proses penggabungan antara dua gelombang dengan frekuensi dan fase tertentu. Pola interferensi terbentuk akibat perbedaan atau kesamaan frekuensi dan fase kedua sinyal. Pemrosesan sinyal digital dan sistem komunikasi sering melibatkan proses ini untuk memahami perubahan bentuk sinyal oleh sistem. Nilai root mean square (rms) yang ditentukan digunakan untuk memperoleh nilai daya (P) dan tegangan (V) pada sinyal sinusoida analog. Pengujian awal pada sistem konvolusi membutuhkan masukan berupa sinyal kontinu yang dapat disebut dengan sinyal sinusoida berfrekuensi tertentu (Sadiku & Nelatury, 2021).r Wavelet adalah teknik pemrosesan sinyal digital menggunakan konsep citra yang

ditransformasi terlebih dulu didekomposisi sub-sub citra sesuai dengan tujuan level transformasi (Utami et al., 2022)

non-stasioner seperti sinyal suara. Penggunaan parameter frame length dan frame shift sangat mempengaruhi hasil pembagian ini. Frame length menentukan panjang tiap frame, sedangkan frame shift mengatur seberapa besar perpindahan antara frame satu dengan berikutnya. Frame shift yang lebih kecil menghasilkan frame yang lebih banyak dan saling bertumpang tindih, meningkatkan akurasi analisis tetapi juga meningkatkan beban komputasi. Oleh karena itu, pemilihan parameter yang tepat sangat penting untuk menyeimbangkan kualitas representasi sinyal dan efisiensi pemrosesan.

Terakhir, pada proses konvolusi sinyal, kita menggabungkan dua sinyal (biasanya sinyal input dan sinyal sistem) untuk menghasilkan sinyal keluaran yang mencerminkan respons sistem terhadap input tersebut. Konvolusi ini penting dalam berbagai aplikasi seperti filter digital. Hasil konvolusi tergantung pada bentuk kedua sinyal yang terlibat. Sebagai contoh, jika salah satu sinyal adalah filter low-pass, maka hasil konvolusinya akan meredam komponen frekuensi tinggi dari sinyal input. Dengan kata lain, konvolusi dapat digunakan untuk memodifikasi sinyal sesuai karakteristik sistem, seperti menajamkan, menghaluskan, atau menggeser fase sinyal. Pemahaman tentang konvolusi penting untuk merancang sistem pengolahan sinyal yang efektif dan sesuai tujuan aplikasi.

HASIL DAN PEMBAHASAN

Tabel 1. Respons Unit Step

No	Code	Gambar
1	<pre> amp = 1 freq = 1 sampling_freq = 10 t = np.arange(0, 1, 1/sampling_freq) signal = amp * np.sin(2 * np.pi * freq * t) fig, ax = plt.subplots() ax.stem(t, signal) ax.set(xlabel='time (s)', ylabel='amplitude', title='Sine Wave') ax.grid() plt.show() </pre>	
2	<pre> sampling_freqs = [12, 16, 18, 20] everforest_colors = ['#f85552', '#dfa000', '#8da101', '#3a94c5'] fig, axs = plt.subplots(2,2, figsize=(10, 8)) fig.suptitle('Sine Waves with Different Sampling Frequencies', fontsize=16) axs = axs.flatten() for i,(sampling_freq, color) in enumerate(zip(sampling_freqs, everforest_colors)): t = np.arange(0, 1, 1/sampling_freq) signal = amp * np.sin(2 * np.pi * freq * t) markerline, stemlines, baseline = axs[i].stem(t, signal, linefmt=color, markerfmt='o', label=f'sampled points {len(signal)}') markerline.set_markerfacecolor(color) markerline.set_markeredgcolor(color) stemlines.set_color(color) stemlines.set_linewidth(2) stemlines.set_alpha(0.7) baseline.set_color(color) axs[i].set(xlabel='time (s)', ylabel='amplitude', title=f'Sine Wave with Sampling Frequency {sampling_freq} Hz') axs[i].grid() axs[i].legend(loc='best') plt.tight_layout() plt.show() </pre>	

3	<pre> freq = [i*100 for i in range(2, 10)] sampling_freq = 1000 fig, axs = plt.subplots(8,1) fig.set_size_inches(10, 24) fig.suptitle('Sine Waves with Different Frequencies', fontsize=16, y=1) axs = axs.flatten() for i, f in enumerate(freq): t = np.arange(0, 1, 1/sampling_freq) signal = 1*np.sin(2 * np.pi * f * t) color = next(color_cycle) markerline, stemlines, baseline = axs[i].stem(t, signal, linefmt=color, markerfmt='o', label=f'sampled points {len(signal)}') markerline.set_markerfacecolor(color) markerline.set_markeredgecolor(color) stemlines.set_color(color) stemlines.set_linewidth(2) stemlines.set_alpha(0.7) baseline.set_color(color) axs[i].set(xlabel='time (s)', ylabel='amplitude', title=f'Sine Wave with Frequency {f} Hz') axs[i].grid() axs[i].legend(loc='best') axs[i].set_xlim(0, 0.05) axs[i].set_ylim(-1.5, 1.5) plt.tight_layout() plt.show() </pre>	
4	<pre> plt.figure("Lagu Gundul-Gundul Pacul") Fs = 8000 t = np.arange(0, 0.25, 1/Fs) c = np.sin(2 * np.pi * 262 * t) d = np.sin(2 * np.pi * 294 * t) e = np.sin(2 * np.pi * 330 * t) f_ = np.sin(2 * np.pi * 249 * t) g = np.sin(2 * np.pi * 392 * t) a = np.sin(2 * np.pi * 440 * t) b = np.sin(2 * np.pi * 494 * t) cl = np.sin(2 * np.pi * 523 * t) nol = np.zeros_like(t) </pre>	

```
nada1 = np.concatenate([c,e,c,e,f_,g,g,nol,b,c1,b,c1,b,g,nol,nol])
nada2 = np.concatenate([c,e,c,e,f_,g,g,nol,b,c1,b,c1,b,g,nol])
nada3 = np.concatenate([c,nol,e,nol,g,nol,f_,f_,g,f_,e,c,f_,e,c,nol])
nada4 = np.concatenate([c,nol,e,nol,g,nol,f_,f_,g,f_,e,c,f_,e,c])

lagu = np.concatenate([nada1, nada2, nada3, nada4])

plt.plot(lagu)
plt.title("Gundul-Gundul Pacul")
plt.tight_layout()
plt.show()

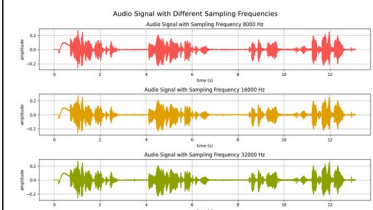
Audio(lagu, rate=Fs)
```

```
sampling_freqs = [8000, 16000, 32000]
everforest_colors = ['#f85552', '#dfa000', '#8da101']
fig, axs = plt.subplots(3,1)
fig.set_size_inches(12, 7)
fig.suptitle('Audio Signal with Different Sampling Frequencies', fontsize=16)
y_resampled = []

y_resampled = librosa.resample(y, orig_sr=sr, target_sr=sampling_freqs[0])
t = np.arange(0, len(y_resampled)) / sampling_freqs[0]
y_resampled.append({'y': y_resampled, 'sr': sampling_freqs[0], })

axs[0].plot(t, y_resampled, color=everforest_colors[0])
axs[0].set(xlabel='time (s)', ylabel='amplitude', title=f'Audio Signal with Sampling Frequency {sampling_freqs[0]} Hz')
axs[0].grid()

y_resampled = librosa.resample(y, orig_sr=sr, target_sr=sampling_freqs[1])
```



```

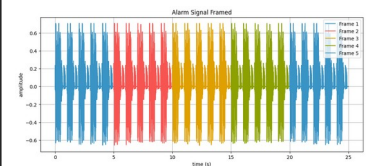
t = np.arange(0, len(y_resampled)) /
sampling_freqs[1]
y_resampled.append({
'y': y_resampled,
'sr': sampling_freqs[1],
})
axs[1].plot(t, y_resampled,
color=everforest_colors[1])
axs[1].set(xlabel='time (s)',
ylabel='amplitude',
title=f'Audio Signal with Sampling
Frequency {sampling_freqs[1]} Hz')
axs[1].grid()
y_resampled = librosa.resample(y,
orig_sr=sr, target_sr=sampling_freqs[2])
t = np.arange(0, len(y_resampled)) /
sampling_freqs[2]
y_resampled.append({
'y': y_resampled,
'sr': sampling_freqs[2],
})
axs[2].plot(t, y_resampled,
color=everforest_colors[2])
axs[2].set(xlabel='time (s)',
ylabel='amplitude',
title=f'Audio Signal with Sampling
Frequency {sampling_freqs[2]} Hz')
axs[2].grid()
plt.tight_layout()
plt.show()

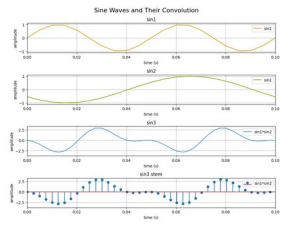
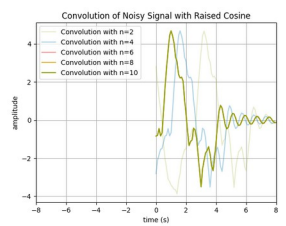
```

```

fig, ax = plt.subplots()
fig.set_size_inches(12, 5)
for i, (frame, t_frame) in
enumerate(zip(framed_alarms, framed_ts)):
color = next(color_cycle)
ax.plot(t_frame, frame, color=color,
label=f'Frame {i+1}')
ax.set(xlabel='time (s)',
ylabel='amplitude',
title='Alarm Signal Framed')
ax.legend()
ax.grid()

```



7	<pre> points = 200 f1 = 20 f2 = 10 theta1= 0 theta2= 10 t = np.linspace(0, 1, points) sin1 = np.sin(2*np.pi*f1*t + theta1) sin2 = np.sin(2*np.pi*f2*t + theta2) sin3 = np.convolve(sin1, sin2) t3 = np.linspace(0, 1, len(sin3)) fig, axs = plt.subplots(4,1) fig.set_size_inches(10, 8) fig.suptitle('Sine Waves and Their Convolution', fontsize=16) axs = axs.ravel() axs[0].plot(t, sin1, label='sin1', color=next(color_cycle)) axs[0].set_title('sin1') axs[0].legend() axs[0].grid() axs[1].plot(t, sin2, label='sin2', color=next(color_cycle)) axs[1].set_title('sin2') axs[1].legend() axs[1].grid() axs[2].plot(t3, sin3, label='sin1*sin2', color=next(color_cycle)) axs[2].set_title('sin3') axs[2].legend() axs[2].grid() axs[3].stem(t3, sin3, label='sin1*sin2') axs[3].set_title('sin3 stem') axs[3].legend() axs[3].grid() for ax in axs: ax.set_xlabel('time (s)') ax.set_ylabel('amplitude') ax.set_xlim(0, 0.1) plt.tight_layout() plt.show() </pre>	
8	<pre> ns = [(i+1)*2 for i in range(0, 5)] fig, ax = plt.subplots() for n in ns: n_range= np.arange(-0.9, 8.1*n, 0.5) y_sinc = np.sinc(4 * n_range / 8) conv = np.convolve(x_noisy, y_sinc, mode='same') </pre>	

```

t = np.arange(0, len(conv)) / 10
color = next(color_cycle)
ax.plot(t, conv, color=color,
label=f'Convolution with n={n}',
alpha=n*0.1)
ax.set(xlabel='time (s)',
ylabel='amplitude',
title='Convolution of Noisy Signal with
Raised Cosine')
ax.legend(loc='best')
ax.grid()
ax.set_xlim(-8, 8)
plt.show()

```

PEMBAHASAN

Kode pertama menunjukkan proses sampling sinyal sinusoidal dengan amplitudo 1 dan frekuensi 1 Hz menggunakan frekuensi sampling sebesar 10 Hz. Dengan menggunakan fungsi stem, hasil visualisasi memperlihatkan titik-titik diskrit dari sinyal tersebut dalam rentang waktu 0 hingga 1 detik. Ini memberikan gambaran dasar bagaimana sinyal analog diubah menjadi sinyal diskrit melalui proses sampling.

Selanjutnya, ditampilkan perbandingan efek dari beberapa frekuensi sampling yang berbeda, yakni 12 Hz, 16 Hz, 18 Hz, dan 20 Hz. Setiap subplot menunjukkan sinyal yang sama namun disampling dengan frekuensi berbeda. Hal ini bertujuan untuk memperlihatkan bagaimana kualitas representasi sinyal semakin meningkat seiring bertambahnya frekuensi sampling, di mana sinyal hasil sampling semakin menyerupai bentuk gelombang aslinya.

Kemudian, eksperimen dilanjutkan dengan mengubah frekuensi sinyal sinus dari 200 Hz hingga 900 Hz, dengan tetap mempertahankan frekuensi sampling sebesar 1000 Hz. Dengan menggunakan beberapa subplot vertikal, kode ini menunjukkan bagaimana sinyal dengan frekuensi lebih tinggi akan menghasilkan gelombang dengan lebih banyak siklus dalam satu detik. Pengaturan xlim dan ylim membantu agar hasil visualisasi tetap terbaca meskipun frekuensi meningkat.

Bagian selanjutnya merupakan pembuatan sinyal audio sederhana dari lagu tradisional "Gundul-Gundul Pacul" menggunakan kombinasi nada dasar. Masing-masing nada dihasilkan dengan fungsi sinus sesuai frekuensi nadanya dan waktu durasi tertentu, kemudian digabungkan menjadi satu array untuk

membentuk lagu secara keseluruhan. Hasilnya divisualisasikan dengan plot, dan juga dapat diputar sebagai audio menggunakan Audio.

Setelah itu, dilakukan proses resampling terhadap sinyal audio dengan tiga frekuensi sampling berbeda: 8000 Hz, 16000 Hz, dan 32000 Hz. Visualisasi dilakukan pada tiga subplot untuk membandingkan hasil sinyal audio setelah proses resampling. Ini menunjukkan bahwa semakin tinggi frekuensi sampling, semakin halus dan akurat bentuk gelombang suara yang dihasilkan.

Pada bagian analisis sinyal alarm, dilakukan pemrosesan sinyal dengan pemotongan berdasarkan frame, kemudian masing-masing frame divisualisasikan secara terpisah. Tujuan dari ini adalah untuk menunjukkan bagaimana sinyal dapat dianalisis secara lokal dalam segmen waktu yang lebih kecil (frame blocking), yang berguna dalam pengolahan sinyal seperti deteksi fitur atau pengenalan pola.

Bagian terakhir menunjukkan proses konvolusi dua sinyal sinus. Dua sinyal sinus dengan frekuensi berbeda (20 Hz dan 10 Hz) dan fase awal yang berbeda dikalikan secara konvolusional. Hasil dari konvolusi ini divisualisasikan menggunakan plot dan stem. Proses konvolusi ini menunjukkan bagaimana dua sinyal berinteraksi satu sama lain dan merupakan dasar dari berbagai aplikasi seperti filtering dalam domain waktu.

Secara keseluruhan, rangkaian kode ini mencakup demonstrasi komprehensif mengenai proses sampling, perubahan frekuensi sampling, pembentukan sinyal audio, resampling, pemrosesan sinyal berbasis frame, hingga operasi konvolusi, yang merupakan aspek penting dalam pemrosesan sinyal digital.

DAFTAR PUSTAKA

Astomo, R. B. W., Kurniawati, I., Mahmudah, A., & Sya'ronni. (2023). Rancang Bangun Instrumen Pengukur Tinggi Gelombang Permukaan Laut Menggunakan Sensor IMU GY955. *JASEE Journal of Application and Science on Electrical Engineering*, (2023), 4(1), 26-36.

Khairunnisa. (2019). *Pengolahan Sinyal*. Poliban Press, Yogyakarta.

Lazzuardhy, D. A., & Prasetyo, B. H. (2022). Sistem Pengenalan Intensitas Emosi Sedih melalui Ucapan menggunakan Ekstraksi Bark-Frequency Cepstral Coefficient dan K-Nearest Neighbor berbasis Raspberry Pi 4. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 6(11), 5560-5568.

Sadiku, M. S., & Nelatury, S. (2021). *Elements of electromagnetics*. Oxford University Press, Oxford.

Shabrina, H., Ahmayani, I., Putri, K. A. E., & Setyowati, E. (2024). Analisis Efek Aliasing Pada Sinyal Audio Dengan Variasi Frekuensi Sampling Pada Lagu Berjudul Terhebat. *Jurnal ELECTRA : Electrical Engineering Articles*, 5(1), 1-11.