

Pengolahan Sinyal Digital

“Signal Processing”



Nama: Fenny Anggraini

NIM: 09011182025019

Kelas : SK6A Indralaya

Dosen pengampuh : Abdurahman, S.Kom.,M.Han

Jurusan Sistem Komputer
Fakultas Ilmu Komputer
Universitas Sriwijaya
2022

Signal Processing

I. Judul Praktikum

“ Signal Processing with python “

II. Tujuan Praktikum

- Memahami definisi dari signal processing
- Melakukan percobaan dengan menggunakan code pada jupyter notebook
- Melakukan analisa terhadap code percobaan signal processing

III. Alat dan Bahan

- PC atau Lapto
- Jupyter notebook
- Anaconda
- Python
- Library Numpy
- Library Matplotlib
- Library Wave


IV. Teori Dasar

Pengertian Signal Processing

Digital Signal Processor atau yang disingkat DSP merupakan sebuah rangkain yang terintegrasi layaknya mikroprosesor. Akan tetapi, arsitekturnya mempunyai spesifikasi untuk bisa menjalankan pemrosesan data dekrit berkecepatan tinggi misalnya *fast fourier transform* dan proses *filtrring*.


Kelebihan inilah yang menjadikan Digital Signal Processor lebih baik jika dibandingkan dengan mikroprosesor maupun mikrokontroler di dalam memproses sinyal. Di dalam memproses data, Digital Signal Processor awalnya akan bekerja dengan cara mengubah sinyal analog menjadi sinyal elektronik oleh sebuah tranduser (microphone). Setelah itu, Digital Signal Processor akan melakukan proses pengambilan sinyal masukan berupaa sinyal kontinyu. Nantinya sinyal kontinyu akan diubah menjadi sinyal diskrit. Proses tersebut dijalankan oleh sebuah Analog to Digital Converter atau ADC.

V. Prosedur Percobaan

 jupyter










Fenny Anggraini_09011182025019_PSD

Last Checkpoint: 29 menit yang lalu (unsaved changes)

 Logout

FileEditViewInsertCellKernelWidgetsHelp

TrustedPython 3 (ipykernel) C



Code

Signal Processing - Pengolahan Signal Digital

Nama : Fenny Anggraini
NIM : 09011182025019
Kelas : SK6A Indralaya

```
In [19]: import numpy as np
import matplotlib.pyplot as plt

import scipy
from scipy import signal
```

```
In [20]: t = np.arange(0, 11)
x = (0.85) ** t
```

```
In [21]: print(t)
print(x)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10]
[1.  0.85  0.7225  0.614125  0.52200625  0.44370531
 0.37714952 0.32057709 0.27249053 0.23161695 0.1968744 ]
```

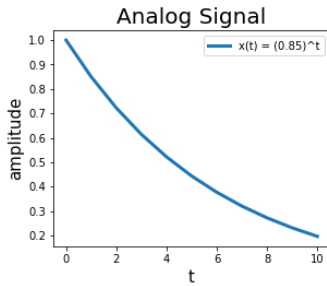
1. Plotting Analog Signal

```
In [22]: plt.figure(figsize = (10,8))

plt.subplot(2, 2, 1)
plt.title('Analog Signal', fontsize=20)

plt.plot(t, x, linewidth=3, label='x(t) = (0.85)^t')
plt.xlabel('t', fontsize=15)
plt.ylabel('amplitude', fontsize=15)
plt.legend(loc='upper right')
```

```
Out[22]: <matplotlib.legend.Legend at 0x20d89a72100>
```

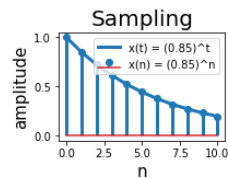


2. Sampling and Plotting of Sampled signal

```
In [23]: plt.subplot(2, 2, 2)
plt.title('Sampling', fontsize=20)
plt.plot(t, x, linewidth=3, label='x(t) = (0.85)^t')
n = t

markerline, stemlines, baseline = plt.stem(n, x, label='x(n) = (0.85)^n')
plt.setp(stemlines, 'linewidth', 3)
plt.xlabel('n', fontsize = 15)
plt.ylabel('amplitude', fontsize = 15)
plt.legend(loc='upper right')
```

Out[23]: <matplotlib.legend.Legend at 0x20d89afbee0>



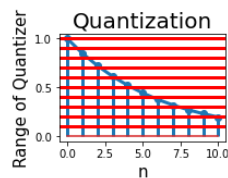
3. Quantization

```
In [24]: plt.subplot(2, 2, 3)
plt.title('Quantization', fontsize = 20)

plt.plot(t, x, linewidth =3)
markerline, stemlines, baseline=plt.stem(n,x)
plt.setp(stemlines, 'linewidth', 3)
plt.xlabel('n', fontsize = 15)
plt.ylabel('Range of Quantizer', fontsize=15)

plt.axhline(y = 0.1, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.2, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.3, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.4, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.5, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.6, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.7, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.8, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.9, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 1.0, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
```

Out[24]: <matplotlib.lines.Line2D at 0x20d89bbdfa0>

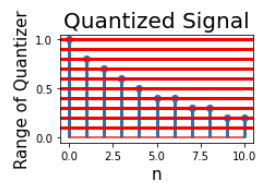


4. Quantized Signal

```
In [25]: plt.subplot(2, 2, 4)
plt.title('Quantized Signal', fontsize = 20)
xq = np.around(x,1)
markerline, stemlines, baseline = plt.stem(n,xq)
plt.setp(stemlines, 'linewidth', 3)
plt.xlabel('n', fontsize = 15)
plt.ylabel('Range of Quantizer', fontsize=15)

plt.axhline(y = 0.1, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.2, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.3, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.4, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.5, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.6, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.7, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.8, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 0.9, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)
plt.axhline(y = 1.0, xmin = 0, xmax = 10, color = 'r', linewidth = 3.0)

plt.tight_layout()
```



5. Signal Impulse

```
In [26]: impulse = signal.unit_impulse(10, 'mid')
shifted_impulse = signal.unit_impulse(7, 2)

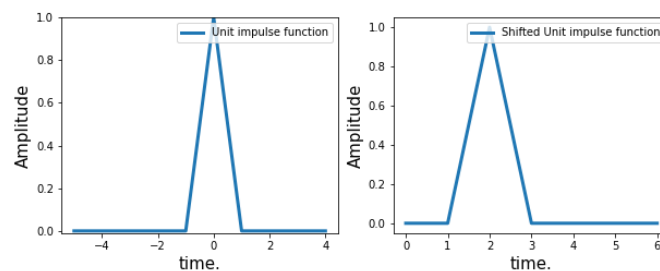
plt.figure(figsize=(10, 8))

plt.subplot(2, 2, 1)
plt.plot(np.arange(-5, 5), impulse, linewidth=3, label='Unit impulse function')
plt.ylim(-0.01,1)
plt.xlabel('time.', fontsize=15)
plt.ylabel('Amplitude', fontsize=15)
plt.legend(fontsize=10, loc='upper right')

plt.subplot(2, 2, 2)
plt.plot(shifted_impulse, linewidth=3, label='Shifted Unit impulse function')

plt.xlabel('time.', fontsize=15)
plt.ylabel('Amplitude', fontsize=15)
plt.legend(fontsize=10, loc='upper right')
```

Out[26]: <matplotlib.legend.Legend at 0x20d89bec550>



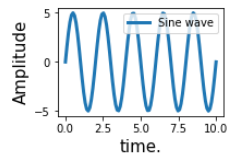
6. Sinus Signal

```
In [27]: t = np.linspace(0, 10, 100)
         amp = 5 # Amplitude
         f = 50
         x = amp * np.sin(2 * np.pi * f * t)

         plt.subplot(2, 2, 3)
         plt.plot(t, x, linewidth=3, label='Sine wave')

         plt.xlabel('time.', fontsize=15)
         plt.ylabel('Amplitude', fontsize=15)
         plt.legend(fontsize=10, loc='upper right')
```

Out[27]: <matplotlib.legend.Legend at 0x20d89d40370>



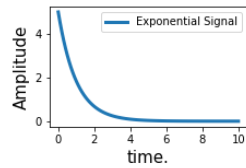
7. Exponen Signal

```
In [28]: x_ = amp * np.exp(-t)

         plt.subplot(2, 2, 4)
         plt.plot(t, x_, linewidth=3, label='Exponential Signal')

         plt.xlabel('time.', fontsize=15)
         plt.ylabel('Amplitude', fontsize=15)
         plt.legend(fontsize=10, loc='upper right')

         plt.tight_layout()
```



8. Discrete Signals

```
In [29]: # Sine wave
n = np.linspace(0, 10, 100)
amp = 5 # Amplitude
f = 50
x = amp * np.sin(2 * np.pi * f * n)

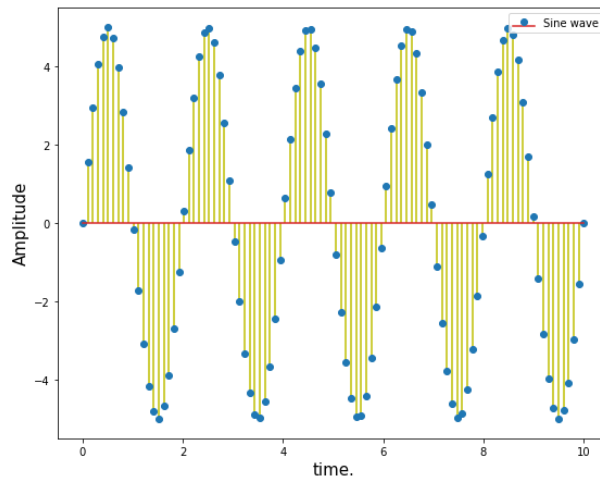
# Exponential Signal
x_ = amp * np.exp(-n)
```

```
In [30]: plt.figure(figsize=(20, 16))

plt.subplot(2, 2, 1)
plt.stem(n, x, 'yo', label='Sine wave')

plt.xlabel('time.', fontsize=15)
plt.ylabel('Amplitude', fontsize=15)
plt.legend(fontsize=10, loc='upper right')
```

Out[30]: <matplotlib.legend.Legend at 0x20d89dfb520>

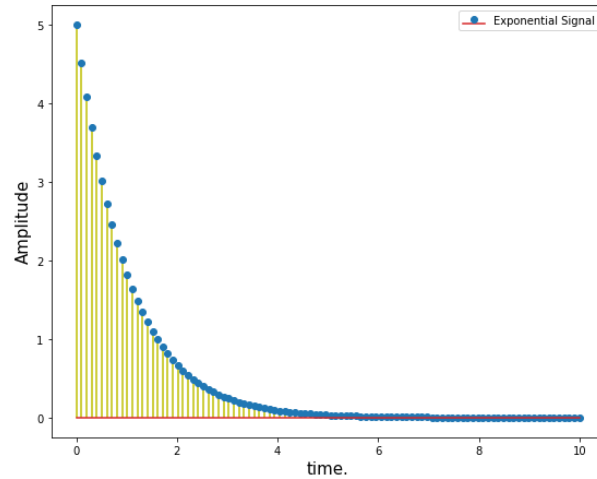


```
In [31]: plt.figure(figsize=(20, 16))

plt.subplot(2, 2, 2)
plt.stem(n, x_, 'yo', label='Exponential Signal')

plt.xlabel('time.', fontsize=15)
plt.ylabel('Amplitude', fontsize=15)
plt.legend(fontsize=10, loc='upper right')
```

Out[31]: <matplotlib.legend.Legend at 0x20d89e9c670>



VI. Kesimpulan

Dari percobaan yang dilakukan di dapatkan kesimpulan mengenai signal processing ;

- Pemrosesan Sinyal adalah bidang ilmu yang melibatkan manipulasi sinyal dari domain waktu ke frekuensi dan sebaliknya, menghaluskan sinyal, memisahkan noise dari sinyal yaitu menyaring, mengekstrak informasi dari sinyal.
- Sinyal yang ada di alam adalah sinyal kontinyu. Sinyal waktu kontinu (atau analog) ada untuk interval kontinu (t_1, t_2) dapat berkisar dari $-\infty$ to $+\infty$
- Karena komputer membutuhkan sinyal digital untuk diproses, oleh karena itu, untuk menggunakan sinyal analog di komputer harus didigitalkan dengan konverter analog-ke-digital. Dengan demikian, ada kebutuhan untuk antarmuka antara sinyal analog dan prosesor sinyal digital.

VII. Daftar Pustaka

[1] Sayah, Fares. 2022. *"Signal Processing with Python"*.

Diakses pada 13 Mei 2022, dari

<https://www.kaggle.com/code/faressayah/signal-processing-with-python/notebook>

Link Github :

https://github.com/fennyanggraini17/Signal-Processing-Fenny_Anggraini-Pengolahan-Signal-Digital.git