PRAKTIKUM
SINYAL DAN SISTEM

VT 045108

Oleh:
Tri Budi Santoso
Haniah Mahmudah
Nur Adi Siswandari

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
2012

KATA PENGANTAR

Alhamdulillah, dengan mengucap puji syukur ke hadirat Alloh SWT, karena atas rahmad dan hidayahNya *Buku Petunjuk Praktikum Sinyal dan Sistem* ini selesai dibuat.

Buku ini disusun untuk memenuhi kebutuhan mahasiswa Politeknik Elektronika Negeri Surabaya dan bisa dipergunakan untuk semua Program Studi yang ada, baik untuk jenjang D3 maupun D4. Buku ini merupakan pengganti dari Modul Sinyal dan Sistem sebelumnya. Dengan metode penyampaian yang sederhana diharapkan siswa dapat memanfaatkan modul ini dalam pembelajaran. Di dalam buku ini juga dilengkapi contoh-contoh yang lebih mengarah ke bentuk yang lebih aplikatif dengan memanfaatkan perangkat lunak Matlab.

Dengan selesainya buku ini dengan tulus ikhlas dan kerendahan hati, kami mengucapkan terima kasih yang mendalam kepada:

1. Para Pimpinan Politeknik Elektronika Negeri Surabaya - Institut Teknologi Sepuluh Nopember yang telah memberikan segala fasilitas, dorongan semangat, dan suasana kerja yang memberikan semangat kami untuk menyelesaikan buku ini.
2. Rekan-rekan di Group *Electromagnetic Compatibility* dan Group *Signal Processing* kampus Politeknik Elektronika Negeri Surabaya yang selalu menyediakan waktunya untuk berdiskusi dalam proses pembuatan buku ini.
3. Keluarga di rumah yang dengan ikhlas meluangkan waktu bagi kami untuk menyelesaikan buku ini.

Mudah-mudahan sumbangan pemikiran yang kecil ini bisa memberikan kontribusi dalam mencerdaskan mahasiswa dilingkungan kampus PENS.

Surabaya, 3 September 2010

Penulis

Daftar Isi

Satuan Acara Pelaksanaan Praktikum

Mata Kuliah : Praktikum Sinyal Sistem
Kode Mata Kuliah :
Jumlah Jam / Minggu : 3 Jam / minggu
Semester : 5 (lima)

Tujuan Instruksional Umum:

Mahasiswa mampu memahami konsep dasar sinyal dan sistem serta mampu menganalisanya dengan berbagai macam metoda, baik untuk sinyal kontinyu maupun diskrit menggunakan perangakat lunak.

Minggu Ke	Topik Pembahasan	Uraian Materi Pembahasan
1	Pendahuluan	• SAP (rencana praktikum) • Kesepakatan kelas • Pra syarat
2	Operasi Dasar Matlab	• Membuat lembar kerja dalam m-file • Perintah operasi sederhana • Fungsi dalam matlab • Pembuatan gambar (plotting)
3	Pembangkitan Sinyal Kontinyu	• Sinyal Sinusoida • Persegi • Pembangkitan Sinyal Kontinyu Fungsi Ramp • Pembangkitan Sinyal dengan file *WAV
4	Pembangkitan Sinyal Diskrit	• Sekuen Impulse • Sekuan Step • Sinusoida Diskrit • Sekuen Konstan • Sekuen Rectangular (persegi)
5	Operasi Dasar pada Sinyal	• Penjumlahan (2 atau lebih sinyal sinus) • Penjumalah sinyal audio dengan noise • Pengurangan noise pada sinyal audio • Perkalian (2 atau lebih sinyal sinus)
6	Penguatan dan Pelemahan Sinyal	• Penguatan pada sinyal sinus • Penguatan pada sinyal audio • Pelemahan sinyal sinus • Pelemahan noise pada sinyal audio
7	Pengolahan Sinyal Analog	• Proses Sampling dengan variasi frekuensi • Proses Aliasing
8	Konvolusi Sinyal Diskrit	• Konvolusi dua sinyal unit step • Konvolusi Dua Sinyal Sekuen konstan • Konvolusi Dua Sinyal Sinus Diskrit
9	Konvolusi Sinyal Kontinyu	• Konvolusi Dua Sinyal Sinus • Konvolusi Sinyal Bernoise dengan Raise Cosine

10	Analisa Sinyal domain frekuensi	• FFT • Analisa Spektrum
11	Analisa Sinyal domain waktu	• IFFT • Implus
12	Proyek 1	• Program Faktorial
13	Proyek 2	• Program Transformasi Z
14	Proyek 3	• Program Transformasi Lapace
15	Pos-Test	Materi dari minggu 2 s/d 14
16	Perbaikan Pos-Test	Materi dari minggu 2 s/d 14

MODUL 1
OPERASI DASAR MATLAB

I. Tujuan Instruksional Khusus

- Mahasiswa mampu mengoperasikan Matlab dan memanfaatkannya sebagai perangkat Simulasi untuk praktikum Sinyal dan Sistem

II. Pengenalan Perangkat Lunak Matlab

Matlab adalah sebuah bahasa dengan (high-performance) kinerja tinggi untuk komputasi masalah teknik. Matlab mengintegrasikan komputasi, visualisasi, dan pemrograman dalam suatu model yang sangat mudah untuk digunakan dimana masalah-masalah dan penyelesaiannya diekspresikan dalam notasi matematika yang familiar. Penggunaan Matlab meliputi bidang–bidang:

- Matematika dan Komputasi
- Pembentukan Algorithm
- Akusisi Data
- Pemodelan, simulasi, dan pembuatan prototipe
- Analisa data, explorasi, dan visualisasi
- Grafik Keilmuan dan bidang Rekayasa

Matlab merupakan singkatan dari *matrix laboratory*. Matlab pada awalnya ditulis untuk memudahkan akses perangkat lunak matrik yang telah dibentuk oleh LINPACK dan EISPACK. Saat ini perangkat Matlab telah menggabung dengan LAPACK dan BLAS library, yang merupakan satu kesatuan dari sebuah seni tersendiri dalam perangkat lunak untuk komputasi matrix.

Dalam lingkungan perguruan tinggi teknik, Matlab merupakan perangkat standar untuk memperkenalkan dan mengembangkan penyajian materi matematika, rekayasa dan kelimuan.

2.1. Kelengkapan pada Sistem Matlab

Sebagai sebuah system, Matlab tersusun dari 5 bagian utama:

1. ***Development Environment***. Merupakan sekumpulan perangkat dan fasilitas yang membantu anda untuk menggunakan fungsi-fungsi dan file-file Matlab. Beberapa perangkat ini merupakan sebuah *graphical user interfaces* (GUI). Termasuk didalamnya adalah *Matlab Desktop* & *Command Window*, *Command History*, sebuah *Editor* & *Debugger*, dan Browsers untuk melihat *Help*, *Workspace*, *Files*, dan *Search Path*.
2. ***Matlab Mathematical Function Library***. Merupakan sekumpulan algoritma komputasi mulai dari fungsi-fungsi dasar seperti: *sum*, *sin*, *cos*, dan *complex arithmetic*, sampai dengan fungsi-fungsi

yang lebih kompek seperti *matrix inverse*, *matrix eigenvalues*, *Bessel functions*, dan *Fast Fourier Transforms*.

3. ***Matlab Language***. Merupakan suatu *high-level matrix/array language* dengan *control flow statements*, *functions*, *data structures*, *input/output*, dan fitur-fitur *object-oriented programming*. Hal ini memungkinkan bagi kita untuk melakukan kedua hal, baik '*pemrograman dalam lingkup sederhana*' untuk mendapatkan hasil yang cepat, dan '*pemrograman dalam lingkup yang lebih besar*' untuk memperoleh hasil-hasil dan aplikasi yang komplek.
4. ***Graphics***. Matlab memiliki fasilitas untuk menampilkan vektor dan matrik sebagai suatu grafik. Di dalamnya melibatkan *high-level functions* (fungsi-fungsi level tinggi) untuk visualisasi data dua dikensi dan data tiga dimensi, *image processing*, *animation*, dan *presentation graphics*. Ini juga melibatkan fungsi level rendah yang memungkinkan bagi anda untuk membiasakan diri untuk memunculkan grafik mulai dari benutk yang sederhana sampai dengan tingkatan graphical user interfaces pada aplikasi MATLAB anda.
5. ***Matlab Application Program Interface*** (API). Merupakan suatu library yang memungkinkan program yang telah anda tulis dalam bahasa C dan Fortranmampu berinterakasi dengan Matlab. Hal ini melibatkan fasilitas untuk pemanggilan routines dari Matlab (*dynamic linking*), pemanggilan Matlab sebagai sebuah *computational engine*, dan untuk membaca dan menuliskan *MAT-files*.

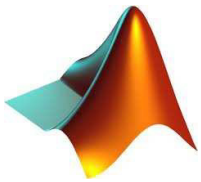
III. Perangkat Yang Diperlukan

- PC/Laptop yang dilengkapi dengan perangkat multimedia (sound card, microphone, speaker aktif, atau headset)
- Sistem Operasi Windows dan Perangkat Lunak Matlab yang dilengkapi dengan tool box DSP

IV. Langkah Percobaan

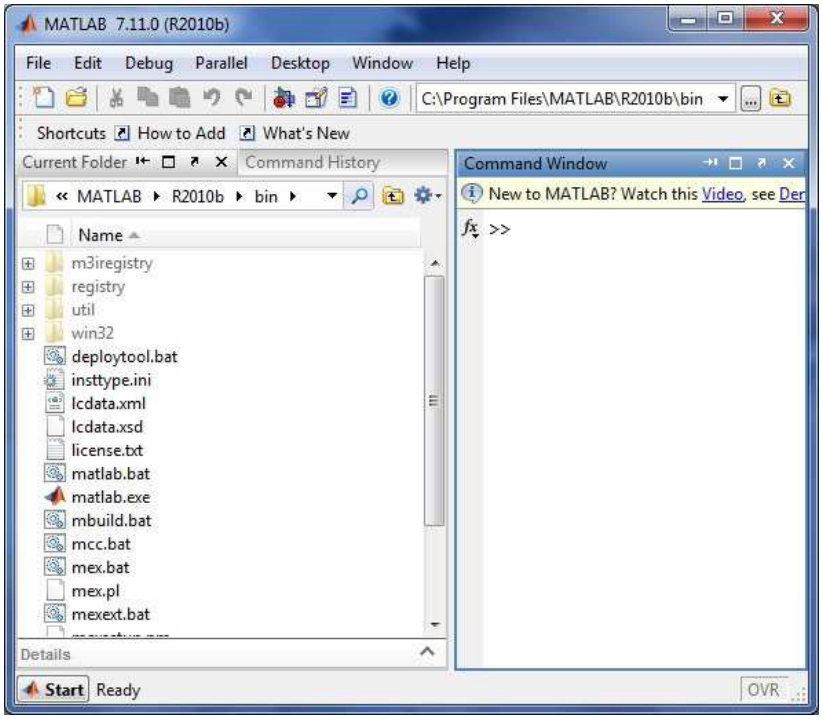
4.1. Memulai Matlab

Anda dapat memulai menjalankan Matlab dengan cara double-clicking pada shortcut icon Matlab.



Gambar 1.1. Icon Matlab pada dekstop PC/Laptop

Selanjutnya anda akan mendapatkan tampilan seperti pada Gambar berikut ini.



Gambar 1.2. Tampilan pertama Matlab

Secara umum tampilan awal Matlab akan menyajikan:

- *Command Window* yang merupakan tempat atau mengetikkan perintah yang dapat dieksekusi secara langsung.
- *Command History* yang berisi berbagai perintah uang telah dieksekusi oleh Command Window. Ini merupakan fitur untuk melakukan tracking ketika proses developing atau debugging programs atau untuk mengkonfirmasi bahwa perintah-perintah telah dieksekusi sepanjang suatu penghitungan multi-step dari command line.
- *Current Folder*, yang menyajikan informasi folder tempat bekarja saat ini dan isi yang ada di folder tersebut. Window ini bermanfaat untuk menemukan lokasi file-file dan script-script sehingga dapat diedit, dipindahkan, diganti nama, dihapus, dsb.

Untuk mengakhiri sebuah sesi Matlab, anda bisa melakukan dengan dua cara, pertama pilih ***File ➔ Exit MATLAB*** dalam window utama Matlab yang sedang aktif, atau cara kedua lebih mudah yaitu cukup ketikkan '***exit***' dalam Command Window.

4.2. Memulai Perintah Sederhana

Penjumlahan dan Perkalian

Langkah kita yang pertama adalah dengan menentukan variable scalar dengan cara melakukan pengetikan seperti berikut

```
>> x = 2 <Enter>
 x =
     2
>> y = 3
 y =
     3
 >> z = x + y
 z =
     5
```

Untuk operasi perkalian anda bisa melakukan seperti berikut

```
>> z = x * y
 z =
     6
```

Saya percaya anda tidak mengalami kesulitan.

Operasi Vektor dan Matrik

Sebuah vektor bisa saja didefinisikan sebagai matrik yang memiliki ukuran 1xN, dengan kata lain sebuah vektor adalah matrik yang hanya memiliki baris sebanyak 1, dan kolom N. Misalnya vektor x merupakan matrik yang berukuran 1x3 dengan nilai-nilai 1, 2 dan 3.

Maka kita bisa mendefinisikan vector x sbb.

```
>> x = [1 2 3]
x =
    1     2     3
```

Sedangkan untuk vektor y yang memiliki jumlah elemen sama, tetapi dengan nilai berbeda bisa dituliskan sebagai

```
>> y = [4 5 6]
y =
    4     5     6
```

Jika anda ingin mengetahui elemen ke 1 dari vektor y, and anda bisa menuliskannya sebagai berikut.

```
>> y(1)
 ans =
     4
```

Sekarang kita jumlahkan keduanya:

```
>> x+y
 ans =
 5    7    9
```

Coba anda rubah vektor y dengan perintah

```
>> y'
 ans =
     4
     5
     6
```

Artinya anda melakukan transpose pada vektor y, kalau belum paham coba anda buka buku matematika anda tentang operasi matrik. Pasti anda menemukan jawabannya....

Sekarang hitung inner product

```
>> x*y'
 ans =
       32
```

Hasil ini diperoleh dari perhitungan seperti ini 1*4 + 2*5 + 3*6 = 32. Dimana **y'** adalah transpose pada y dan merupakan suatu vector kolom.

Jika anda ingin melakukan operasi perkalian sebagai dua vektor baris, coba lakukan dengan perintah perkalianelement-demi-element:

```
>>  x.*y
 ans =
       4    10    18
```

Suatu saat anda harus merubah vektor menjadi sebuah matrik dengan ukuran tiga kali satu (3x1). Untuk itu anda harus memodifikasi penulisan vektor menjadi matrik dengan memasukkan tanda semicolon (;) diantara angka-angka tersebut.

```
>> x=[1;2;3]
x =
     1
     2
     3
```

Matlab memberikan kemudahan anda untuk melakukan cara cepat dalam menyusun vektor/matrik tertentu, misalnya

```
>> x = ones(1, 10)
x =
    1    1    1    1    1    1    1    1    1    1
```

Atau

```
>> x = zeros(3, 1)
x =
    0
    0
    0
```

Bilangan Acak

Anda bisa melakukan pembangkitan bilangan acak dengan mudah, misalnya anda akan membangkitkan sebuah vektor yang tersusun dari 10 bilangan acak terdistribusi uniform.

```
>> rand(1,10)
ans =
  0.8147    0.9058    0.1270    0.9134    0.6324    0.0975    0.2785    0.5469    0.9575    0.9649
```

Atau anda ingin membangkitkan bilangan acak terdistribusi Gaussian (normal)

```
>> randn(1,10)
ans =
  -1.3499    3.0349    0.7254   -0.0631    0.7147   -0.2050   -0.1241    1.4897    1.4090
   1.4172
```

Untuk bilangan binary anda bisa melakukannya seperti berikut

```
>> randint(1,10)
ans =
    1    0    1    1    1    1    1    0    1    0
```

Coba anda lakukan pembangkitan sebuah vektor denan cara seperti berikut

```
>> randint(1,10,4)
```

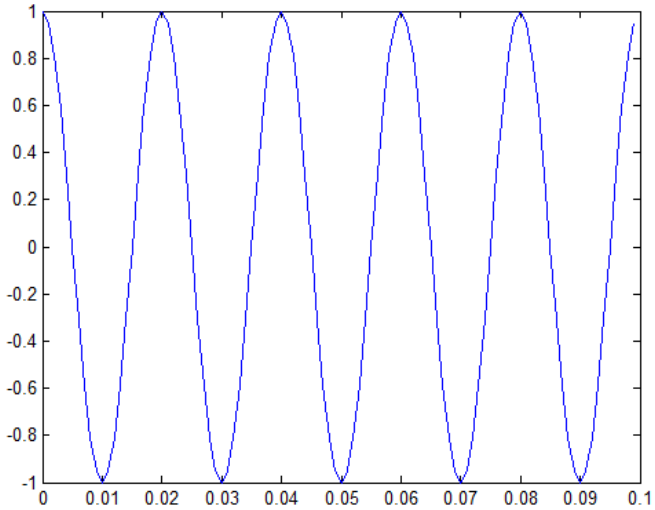
Jelaskan apa yang anda lihat pada layar monitor. Kemudian anda coba bangkitkan sebuah matrik berukuran 10x 10 yang tersusun dari bilangan acark berdistribusi Gaussian.

Membuat Grafik

Salah satu kelebihan dari Matlab adalah kemudahan dalam mengolah grafik. Sehingga anda tidak perlu kesulitan untuk melihat suatu respon system, misalnya pada kasus melihat bentuk sinyal dalam domain waktu anda cukup mengikuti langkah berikut.

```
>>  time = [0:0.001:0.099];
>> x = cos(0.1*pi*(0:99));
>>  plot(time,x)
```

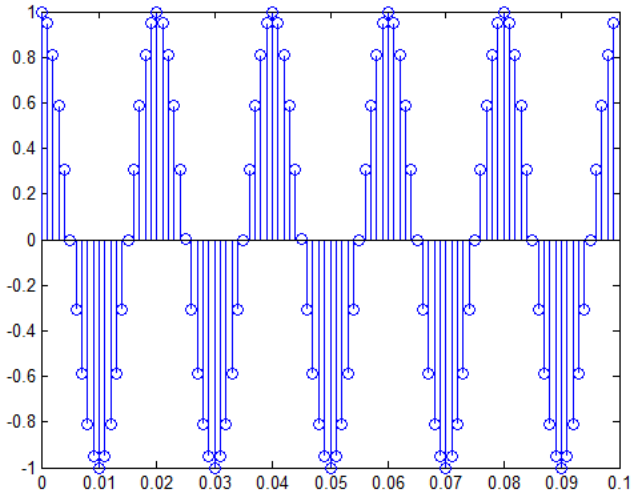
Dan akan tampil sebuah grafik sinuoida seperti berikut. Hal ini merepresentasikan sebuah sinyal dalam domain waktu konitnyu.



Gambar 1.3. Grafik sinyal waktu kontinyu

Untuk sederetan nilai fungsi waktu diskrit adalah dengan menggunakan perintah "stem". Dari contoh deretan perintah coba anda rubah beberapa bagian dengan perintah berikut.

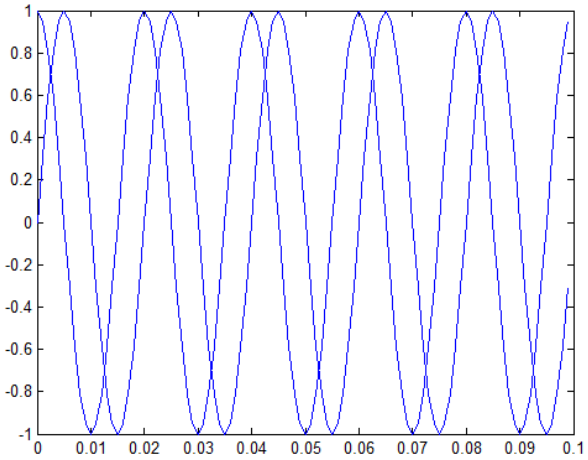
```
>>  stem(time,x)
```



Gambar 1.4. Grafik sinyal waktu diskrit

Anda bisa melakukan penggabungan lebih dari satu grafik pada sebuah tampilan. Dengan perintah *hold on* dan *hold off*.

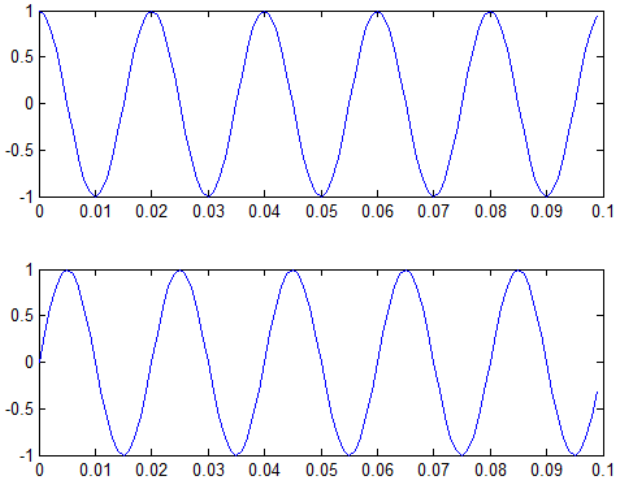
```
>>  time = [0:0.001:0.099];
>> x = cos(0.1*pi*(0:99));
>> y = sin(0.1*pi*(0:99));
>> plot(time,x)
>> hold on
>> plot(time,y)
>> hold off
```



Gambar 1.5. Grafik dua sinyal bersamaan

Jika anda ing in menampilkan dua buah gambar pada frame berbeda-beda, anda bisa memanfaatkan perintah subplot. Coba modifikasi beberapa perintah seperti berikut.

```
>> subplot(211); plot(time,x)
 >>subplot(212); plot(time,y)
```



Gambar 1.6. Tampilan dua grafik pada dua frame

Coba anda melakukan modifikasi perintah diatas dengan cara seperti berikut.

```
>> figure(1);
>> plot(time,x)
>> figure(2);
>> plot(time,y)
```

Silahkan anda berkreasi dengan berbagai model tampilan grafik, jika anda kurang puas anda bisa mengetikkan perintah help pada Matlab Command Window. Anda gali sebanyak-banyaknya tentang menampilkan grafik dengan Matlab. Selamat mencoba...

Membuka File

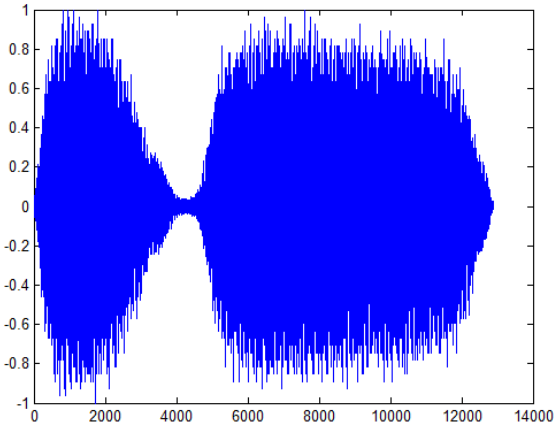
Matlab memiliki kemudahan di dalam membuka file-file tertentu yang sudah didukung oleh library-nya. Dalam hal ini anda bisa membuka file text, suara atau gambar.

```
>> clear all
>> load train
>> whos
```

Name	Size	Bytes	Class	Attributes
Fs	1x1	8	double	
y	12880x1	103040	double	

Dalam hal ini ditunjukkan bahwa hasil pemanggilan file ‘train’ dipanggil (load) secara default dengan frakuensi sampling sebesar Fs (44100 Hz), disimpan sementara pada matrik y (default) dengan ukuran 12880 x1, jumlah Byte sebesar 103040 dan merupakan tipe data double, Atribute tidak ada penjelasan. Untuk mengetahui bagaimana bunyi file tersebut, anda bisa memberikan perintah berikut.

```
>> sound(y,Fs)
>> plot(y)
```



Gambar 1.7. Grafik dari file train

Coba anda lakukan untuk file ‘handel’, ‘gong’, dan ’chirp’.

Salah satu fungsi lain untuk membaca file suara adalah '*wavread*', sedangkan untuk memainkan anda bisa memanfaatkan '*wavplay*', dan untuk menyimpan anda bisa memanfaatkan '*wavwrite*'. Tentu saja anda harus banyak memanfaatkan fasilitas help untuk lebih mengenali fungsi-fungsi ini.

Anda bisa memanfaatkan perintah load untuk memanggil sebuah file gambar yang ada di dalam direktori standar Matlab dengan perintah load. Untuk membuka file gambar anda juga bisa memanfaatkan fungsi '*imread*', sedangkan untuk menyimpannya anda bisa memanfaatkan fungsi '*imwrite*'. Format gambar yang bisa diolah dengan matlab cukup banyak, seperti tiff, jpeg, bmp dan png.

Anda masih penasaran? Coba rubah direktori tempat sesuai dengan direktori dimana anda menyimpan file gambar yang anda punya. Dalam contoh berikut ini kita pilih file actress tanah air yang cukup populer, 'Agnes Monica'. Kalau anda belum punya filenya, anda bisa download dari berbagai sumber yang ada di internet. Kemudian coba anda ketikkan perintah berikut ini. Jika anda kesulitan mencari file gambar Agness Monica, ambil foto teman sebelah anda dan beri nama '***agnes.jpg***'.

```
>> y=imread('agnes.jpg');
>> imshow(y)
```

Anda bisa mengklonversi dari format RGB menjadi format Gray seperti berikut

```
>> yg=rgb2gray(y);
>> imshow(yg)
```

Untuk mengetahui karakter file sebelum dan sesudah proses konversi anda bisa melakukan perintah berikut.

```
>> whos
```

Seperti sebelumnya, anda bisa melihat perbandingan jumlah array penyusunnya, dll. Dalam hal ini format rgb tersusun dari komponen pixcell x,y, r (red), g (green) dan b (blue). Sedangkan format gray tersusun dari komponen x,y, dan gray level. Selengkapnya bisa dilihat seperti berikut.

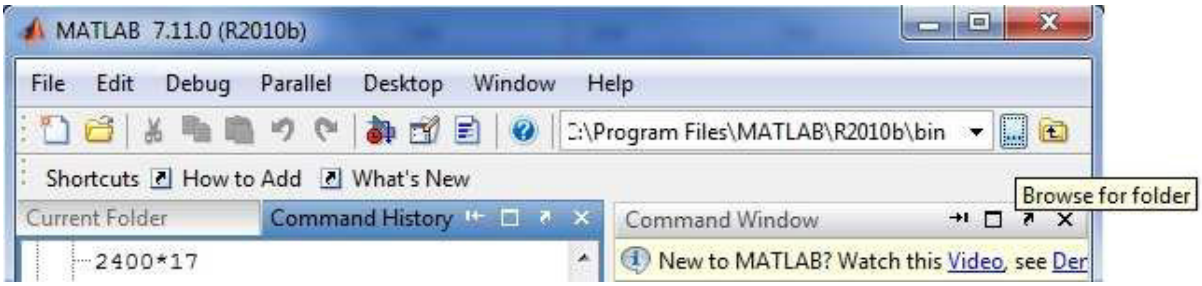
```
Name         Size              Bytes  Class     Attributes
X          200x320            512000  double
caption      2x1                  4  char
map         81x3                1944  double
y         214x235x3            150870  uint8
yg         214x235             50290  uint8
```

(a) Format RGB (b) Format Gray

Gambar 1.8. Tampilan file image

4.3. Menentukan Direktori Tempat Bekerja

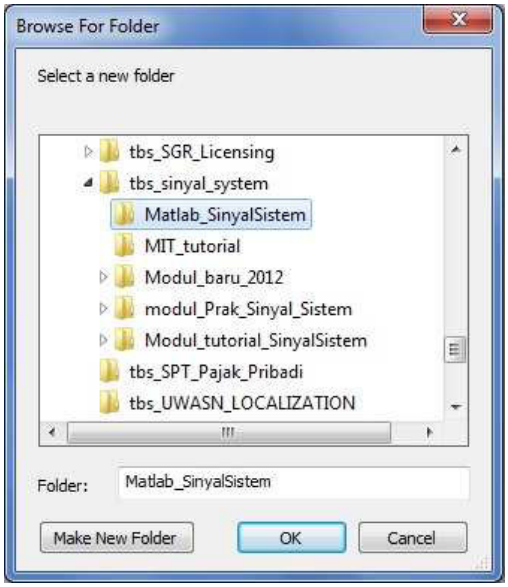
Anda dapat bekerja dengan Matlab secara default pada directory Work ada di dalam Folder Matlab. Tetapi akan lebih bagus dan rapi jika anda membuat satu directory khusus dengan nama yang sudah anda persiapkan, "Matlab_SinyalSistem" atau nama yang lain yang mudah untuk diingat. Hal ini akan lebih baik bagi anda untuk membiasakan bekerja secara rapi dan tidak mencampur program yang anda buat dengan program orang lain. Untuk itu arahkan pointer mouse anda pada kotak bertanda… yang ada disebelah kanan tanda panah kebawah (yang menunjukkan folder yang sedang aktif).



Gambar 1.9. Proses awal membuat folder tempat bekerja

Selanjutnya akan muncul sebuah window ***Browse For Folder***, anda click pada tombol ***Make New Folder***, dan ketikkan nama folder untuk membuat direktori tempat kerja anda. Dalam hal ini anda bisa memberi nama '*Matlab_SinyalSistem*', dan dilanjutkan dengan menekan tombok ***Ok***.

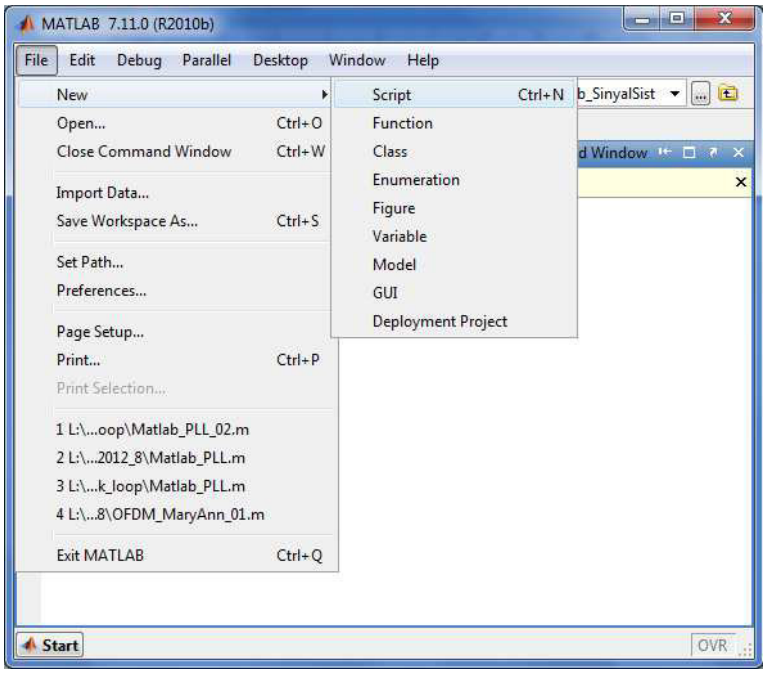
Setelah langkah ini pada tampilan Matlab Current Folder akan mengalami perubahan, bisa saja tampil kosong tanpa ada satu file apapun. Hal ini terjadi karena anda belum mebuat sebuah programatau mengkopikan file ke folder tersebut.



Gambar 1.10. Membuat nama folder baru tempat bekerja

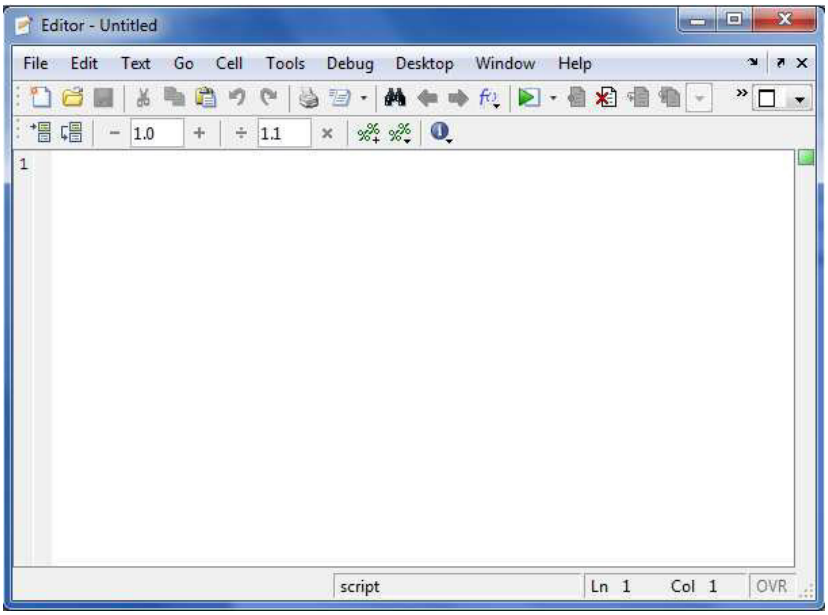
4.4. Membuat Program Baru

Anda dapat menyusun sebuah program di Matlab dengan memanfaatkan Matlab Editor, sehingga setelah anda selesai dengan sebuah proses perhitungan anda bisa menyimpan program tersebut, memanggil di waktu yang lain atau melakukan perubahan sesuai keinginan anda. Ini dapat dilakukan jika anda membuat program Matlab pada Matlab editor, caranya adalah dengan menekan Click pada **File→New→ Script Ctrl+N**. Pada versi Matlab berbeda, bisa saja caranya sedikit berbeda.



Gambar 1.11. Membuat program baru dengan Matlab Editor

Selanjutnya anda akan mendapatkan sebuah tampilan Matlab Editor yang masih kosong seperti ini.



Gambar 1.12. Tampilan awal Matlab Editor

Untuk membuat program anda bisa mengetikkan script berikut ini.

```
%File Name: buka_gambar.m
clear all
y=imread('agnes.jpg');
figure(1);
imshow(y)
yg=rgb2gray(y);
figure(2);
imshow(yg)
```

Anda simpan dengan cara click tanda panah hijau ke arah kanan, dan beri nama '*buka_gambar.m*'. Selanjutnya secara otomatis Matlab akan melakukan eksekusi program anda. Anda bisa juga hanya menyimpan dengan cara Click gambar floppy disk pada toolbar Matlab Editor, atau bisa juga dengan Click pada File→Save Ctrl+S→ tuliskan nama file, misalnya '*buka_gambar.m*' Tentu saja penyimpanan anda lakukan pada folder yang sudah ditetapkan dimana file image '*agnes.jpg*' berada.

4.5. Membuat Fungsi Matlab

Anda bisa membangun fungsi sendiri dengan Matlab Editor. Setelah anda mebuka Matlab Editor, anda ketikkan script berikut ini.

```
function y = x2(t)
y = t.^2;
```

Anda simpan dengan nama ‘x2.m’, kalau anda lupa tidak perlu kawatir, sebab Matlab akan secara otomatis menyimpan sesuai dengan nama variable di belakang perintah ***function***, yaitu x2. Untuk memanfaatkan fungsi tersebut, anda bisa memanggilnya melalui perintah di Matlab Command Window,

```
>>t=0:1:10;
>> y_2=x2(t)
y_2 =
    0    1    4    9   16   25   36   49   64   81  100
```

Atau bisa juga melalui sebuah program yang anda buat pada Matlab Editor.

V. TUGAS

1. Dari contoh-contoh program yang sudah anda jalankan, coba berikan penjelasan arti setiap perintah terhadap output yang dihasilkannya.
2. Coba anda cari bagaimana cara menampilkan grafik untuk tampilan tiga dimensi dan grafik polar.
3. Bagaimana cara menampilkan lebih dari satu persamaan dalam satu grafik? Misalnya anda memiliki dua fungsi sinus yang berbeda fase. Fungsi pertama anda tampilkan, lalu anda lanjutkan menampilkan fungsi kedua, dengan catatan tamplan pada fungsi pertama tidak boleh hilang.
4. Bagaimana cara menampilkan lebih dari satu grafik dalam satu tampilan? Misalnya anda gunakan fungsi pada soal ke-3, satu fungsi ditampilkan diatas dan fungsi lainya di bagian bawah.
5. Bagimana cara menampilkan dua fungsi diman masing-masing fungsi disajikan dalam grafik berbeda. Misalnya anda gunakan contoh kasus pada soal ke-3, fungsi pertama anda tampilkan pada figure(1), sementara fungsi kedua anda tampilkan pada figure(2).

MODUL 2
PEMBANGKITAN SINYAL KONTINYU

I. Tujuan Instruksional Khusus:

- Setelah melakukan praktikum ini, diharapkan mahasiswa dapat membangkitkan beberapa jenis sinyal waktu kontinyu dasar yang banyak digunakan dalam analisa Sinyal dan Sistem.

II. Dasar Teori Sinyal

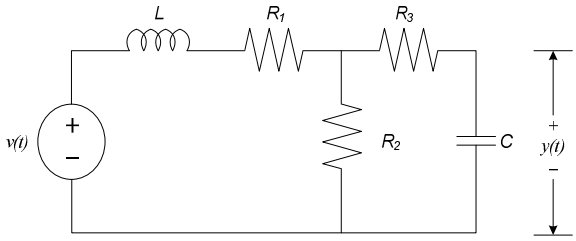
2.1. Konsep Dasar Tentang Sinyal

Sinyal merupakan sesuatu yang secara kuantitatif bisa terdeteksi dan digunakan untuk memberikan informasi yang berkaitan dengan fenomena fisik. Contoh sinyal yang kita temui dalam kehidupan sehari hari, suara manusia, cahaya, temperatur, kelembaban, gelombang radio, sinyal listrik, dsb. Sinyal listrik secara khusus akan menjadi pembicaraan di dalam praktikum ini, secara normal diskpresikan di dalam bentuk gelombang tegangan atau arus. Dalam aplikasi bidang rekayasa, banyak sekali dijumpai bentuk sinyal-sinyal lingkungan yang dikonversi ke sinyal listrik untuk tujuan memudahkan dalam pengolahannya.

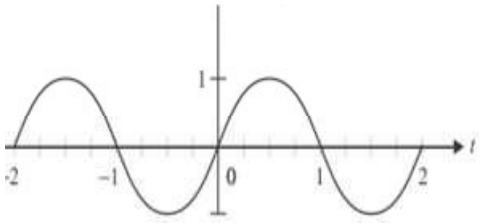
Secara matematik sinyal biasanya dimodelkan sebagai suatu fungsi yang tersusun lebih dari satu variabel bebas. Contoh variabel bebas yang bisa digunakan untuk merepresentasikan sinyal adalah waktu, frekuensi atau koordinat spasial. Sebelum memperkenalkan notasi yang digunakan untuk merepresentasikan sinyal, berikut ini kita mencoba untuk memberikan gambaran sederhana berkaitan dengan pembangkitan sinyal dengan menggunakan sebuah sistem.

Perhatikan Gambar 2.1, yang mengilustrasikan bagaimana sebuah sistem di bidang rekayasa (engineering) dan bentuk sinyal yang dibangkitkannya. Gambar 2.1a merupakan contoh sederhana sistem rangkaian elektronika yang tersusun dari sebuah capasitor C, induktor L dan resistor R. Sebuah tegangan v(t) diberikan sebagai input dan mengalis melalui rangkaian RLC, dan memberikan bentuk output berupa sinyal sinusoida sebagai fungsi waktu seperti pada Gambar 2.1b. Notasi v(t) dan y(t) merupakan variabel tak bebas, sedangkan notasi t merupakan contoh variabel bebas. Pada Gambar 2.1c merupakan sebuah ilustrasi proses perekaman menggunakan digital audio recorder. Sedangkan Gambar 2.1d adalah contoh sinyal ouput hasil perekaman yang disajikan di dalam bentuk grafik tegangan sebagai fungsi waktu.

Salah satu cara mengklasifikasi sinyal adalah dengan mendefinisikan nilai-nilainya pada variabel bebas t (waktu). Jika sinyal memiliki nilai pada keselutuhan waktu t maka didefinisikan sebagai sinyal waktu kontinyu atau consinous-time (CT) signal. Disisi lain jika sinyal hanya memiliki nilai pada waktu-waktu tertentu (diskrete), maka bisa didefinisikan sebagai sinyal waktu diskrit atau discrete-time (DT) signal.

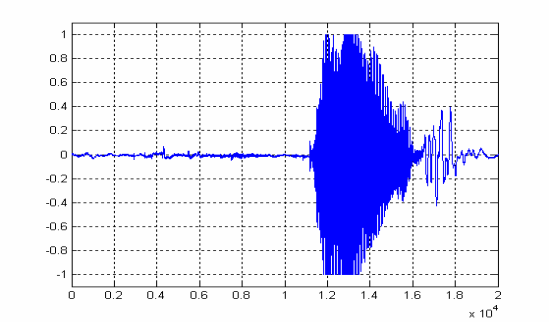


(a) Rangkaian RLC



(b) Sinyal output rangkaian RLC

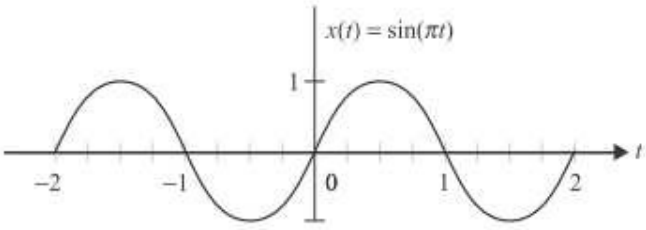
(c) Perekaman suara



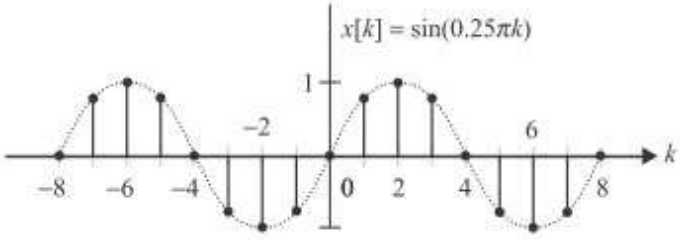
(b) Sinyal output perekaman

Gambar 2.1. Contoh gambaran sistem dan sinyal ouput yang dihasilkan

Contoh bentuk sinyal waktu kontinyu bisa dilihat seperti pada Gambar 2.2a, yang dalam hal ini memiliki bentuk sinusoida dan bisa dinyatakan dalam bentuk fungsi matematik $x(t) = \sin(\pi t)$. Sedangkan contoh sinyal waktu diskrit bisa dibentuk dengan menggunakan bentuk dasar sinyal yang sama, tetapi nilai-nilainya muncul pada setiap interval waktu $T = 0.25dt$, dengan bentuk representasi matematik sebagai berikut, $x[k] = \sin(0.25\pi k)$. Dan gambaran sinyal waktu diskrit pada sekuen dengan rentang waktu $-8 \leq k \leq 8$ bisa dilihat seperti pada Gambar 2.2b.



(a) Sinyal sinus waktu kontinyu



(b) Sinyal sinus waktu diskrit

Gambar 2.2. Sinyal waktu kontinyu dan sinyal waktu diskrit

2.2. Sinyal Waktu Kontinyu

Suatu sinyal *x(t)* dikatakan sebagai sinyal waktu-kontinyu atau sinyal analog ketika dia memiliki nilai real pada keseluruhan rentang waktu *t* yang ditempatinya. x(t) disebut sinyal waktu kontinyu, jika t merupakan variabel kontinyu. Sinyal waktu kontinyu dapat didefinisikan dengan persamaan matematis sebagai berikut:

$$f(t) \in (-\infty, \infty) \tag{2-1}$$

Dimana *f*(t) adalah variabel tidak bebas yang menyatakan fungsi sinyal waktu kontinyu sebagai fungsi waktu. Sedangkan t merupakan variabel bebas, yang bernilai antara – tak hingga (-∞) sampai + tak hingga (+∞).

Hampir semua sinyal di lingkungan kita ini mseupakan sinyal waktu kontinyu. Berikut ini adalah yang sudah umum:

- Gelombang tegangan dan arus yang terdapat pada suatu rangkaian listrik
- Sinyal audio seperti sinyal wicara atau musik
- Sinyal bioelectric seperti electrocardiogram (ECG) atau electro encephalogram (EEG)
- Gaya-gaya pada torsi dalam suatu sistem mekanik
- Laju aliran pada fluida atau gas dalam suatu proses kimia

Sinyal waktu kontinyu memiliki bentuk-bentuk dasar yang tersusun dari fungsi dasar sinyal seperti fungsi step, fungsi ramp, sinyal periodik, sinyal eksponensial dan sinyal impulse.

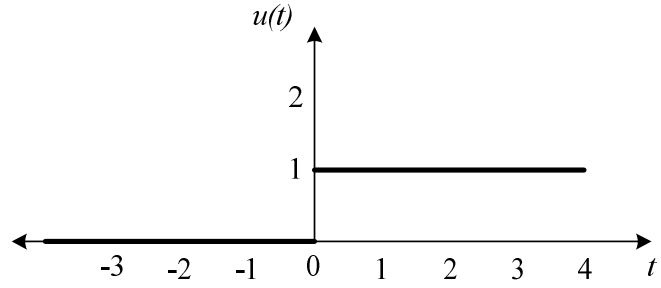
Fungsi Step

Dua contoh sederhana pada sinyal kontinyu yang memiliki fungsi *step* dapat diberikan seperti pada Gambar 2.3a. Sebuah fungsi *step* dapat diwakili dengan suatu bentuk matematis sebagai:

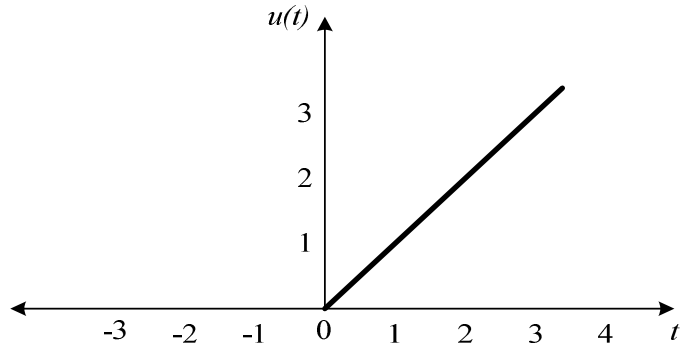
$$u(t) = \begin{cases} 1, & t \geq \\ 0, & t < 0 \end{cases} \tag{2-2}$$

Dimana t merupakan variabel bebas bernilai dari -∞ sampai +∞, dan u(t) merupakan variabel tak bebas yang memiliki nilai 1 untuk t ≥ 0, dan bernilai 0 untuk t < 0. Pada contoh tersebut fungsi step memiliki nilai khusus, yaitu 1 sehingga bisa disebut sebagai *unit step*. Pada kondisi real, nilai output u(t) untuk t ≥ 0 tidak selalu sama dengan 1, sehingga bukan merupakan unit step.

Untuk suatu sinyal waktu-kontinyu *x(t)*, hasil kali *x(t)u(t)* sebanding dengan *x(t)* untuk $t > 0$ dan sebanding dengan nol untuk $t < 0$. Perkalian pada sinyal *x(t)* dengan sinyal *u(t)* mengeliminasi suatu nilai *non-zero*(bukan nol) pada *x(t)* untuk nilai $t < 0$.



(a) . Fungsi step dengan u(t) = 1, untuk t ≥ 0



(b) . Fungsi ramp, dengan r(t) = t, untuk t ≥ 0

Gambar 2.3. Fungsi step dan fungsi ramp

Fungsi Ramp

Fungsi ramp (tanjak) untuk sinyal waktu kontinyu didefinisikan sebagai berikut

$$r(t) = tu(t) = \begin{cases} t, & t \geq 0 \\ 0, & t < 0 \end{cases} \qquad (2\text{-}3)$$

Dimana nilai t bisa bervariasi dan menentukan kemiringan atau *slope* pada *r(t).* Untuk contoh diatas nilai r adalah 1, sehingga pada kasus ini *r(t)* merupakan "*unit slope*", yang mana merupakan alasan bagi *r(t)* untuk dapat disebut sebagai *unit-ramp function*. Jika ada variable *K* sedemikian hingga membentuk *Kr(t)*, maka *slope* yang dimilikinya adalah *K* untuk $t > 0$. Suatu fungsi *ramp* diberikan pada Gambar 2.3b.

Sinyal Periodik

Ditetapkan *T* sebagai suatu nilai real positif. Suatu sinyal waktu kontinyu *x(t)* dikatakan periodik terhadap waktu dengan periode *T* jika :

$$x(t + T) = x(t) \text{ untuk semua nilai } t, -\infty < t < \infty \qquad (2\text{-}4)$$

Dalam hal ini jika *x(t)* merupakan periodik pada periode *T*, ini juga periodik dengan *qT*, dimana *q* merupakan nilai integer positif. Periode fundamental merupakan nilai positif terkecil *T* untuk persamaan (2-5).

Suatu contoh sinyal periodik memiliki persamaan seperti berikut

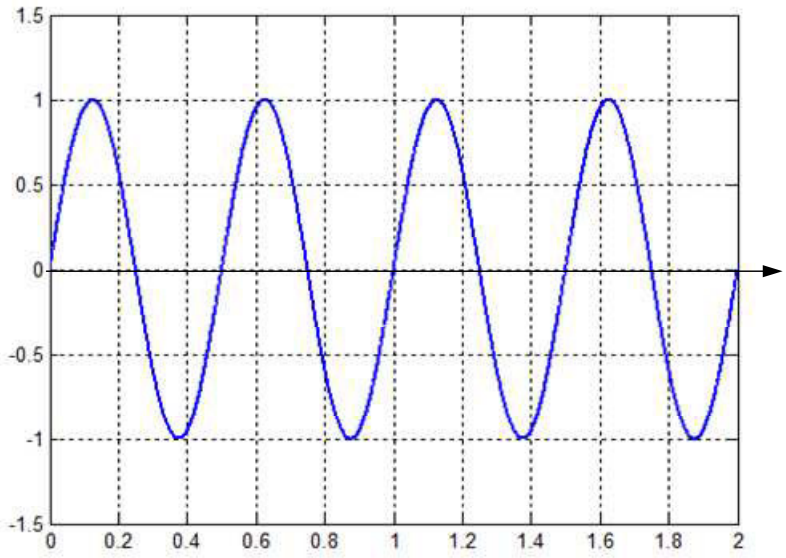
$$x(t)=A\cos(\omega t+\theta) \tag{2-5}$$

Dimana *A* adalah amplitudo, ω adalah frekuensi dalam radian per detik (rad/detik), dan θ adalah fase dalam radian. Frekuensi *f* dalam hertz (Hz) atau siklus per detik adalah sebesar $f = \omega/2\pi$.

Untuk melihat bahwa fungsi sinusoida yang diberikan dalam persamaan (5) adalah fungsi periodik, untuk nilai pada variable waktu *t*, maka:

$$A\cos\left[\omega\left(t+\frac{2\pi}{\omega}\right)+\theta\right]=A\cos(\omega t+2\pi+\theta)=A\cos(\omega t+\theta) \tag{2-6}$$

Sedemikian hingga fungsi sinusoida merupakan fungsi periodik dengan periode $2\pi/\omega$, nilai ini selanjutnya dikenal sebagai periode fundamentalnya. Sebuah sinyal dengan fungsi sinusoida $x(t) = A\cos(wt+\theta)$ diberikan pada Gambar 2.4 untuk nilai $\theta = 0$, dan $f = 2$ Hz.



Gambar 2.4 Sinyal periodik sinusoida

Sinyal periodik bisa berbentuk sinyal rectangular, sinyal gigi gergaji, sinyal segituga, dsb. Bahkan pada suatu kondisi sinyal acak juga bisa dinyatakan sebagai sinyal periodik, jika kita mengetahui bentuk perulangan dan periode terjadinya perulangan pola acak tersebut. Sinyal acak semacam ini selanjutnya disebut sebagai sinyal semi acak atau sinyal pseudo random.

Sinyal Eksponensial

Sebuah sinyal waktu kontinyu yang tersusun dari sebuah fungsi eksponensial dan tersusun dari frekuensi komplek s = σ + jωθ, bisa dinyatakan sebagai berikut:

$$x(t) = e^{st} = e^{(\sigma + j\omega_0)t} = e^{\sigma t}\left(\cos \omega_0 t + j \sin \omega_0 t\right) \qquad (2\text{-}7)$$

Sehingga sinyal waktu kontinyu dengan fungsi eksponensial bisa dibedakan dengan memlilah komponen real dan komponen iamjinernya seperti berikut:

- komponen real $\mathrm{Re}\{e^{st}\} = e^{\sigma t} \cos \omega_0 t$
- komponen imajiner $\mathrm{Im}\{e^{st}\} = e^{\sigma t} \sin \omega_0 t$

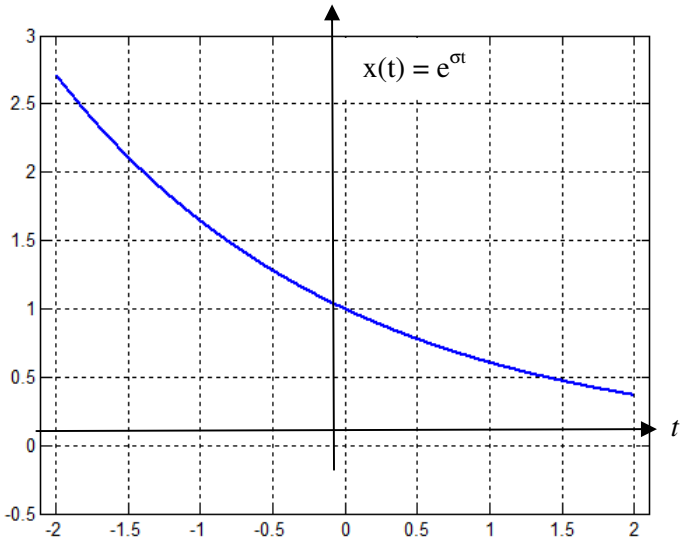
Tergantung dari kemunculan komponen real atau imajiner, dalam hal ini ada dua kondisi khusus yang banyak dijumpai pada sinyal eksponensial, yaitu

Kasus 1: Komponen imajiner adalah nol ($\omega_0 = 0$)

Tanpa adanya komponen imajiner, menyebabkan bentuk sinyal eksponensial menjadi seperti berikut

$$x(t) = e^{\sigma t} \qquad (2\text{-}8)$$

Dimana x(t) merepresentasikan sebuah nilai real pada fungsi eksponensial. Gambar 2.5 memberi ilustrasi nilai real pada fungsi eksponensial pada suatu nilai σ. Ketika nilai σ negatif (σ < 0), maka fungsi eksponensial menujukkan adanya peluruhan nilai (decays) sesuai dengan kenaikan waktu t.



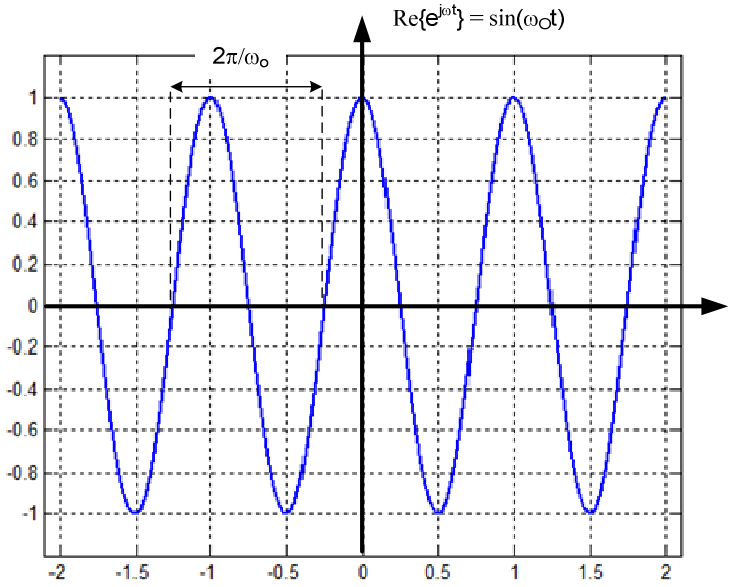
Gambar 2.5 Fungsi eksponensial dengan komponen frekuensi imajiner nol

Kasus 2: Komponen real adalah nol ($\sigma = 0$)

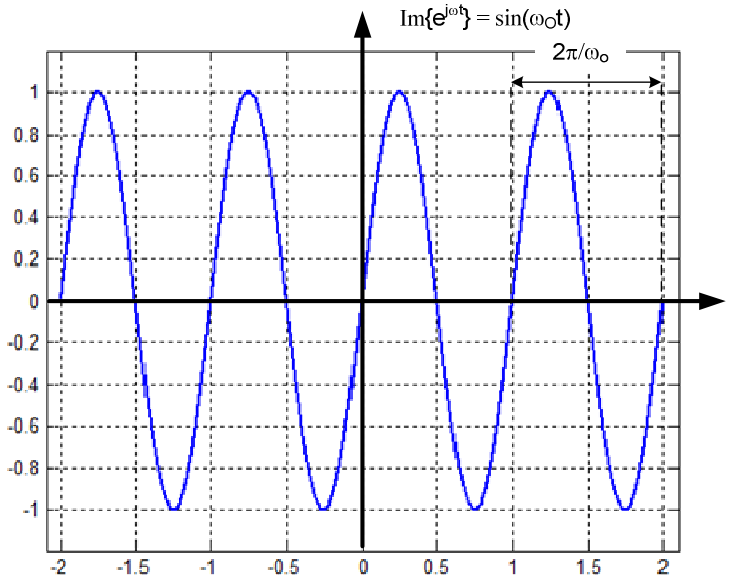
Ketika komponen real σ pada frekuensi komplek s adalah nol, fungsi eksponensial bisa dinyatakan sebagai

$$x(t) = e^{j\omega_0 t} = \cos \omega_0 t + j \sin \omega_0 t \qquad (2\text{-}9)$$

Dengan kata lain bisa dinyatakan bahwa bagian real dan imajiner dari eksponensial komplek adalah sinyal sinusoida murni. Contoh sinyal eksponensial komplek dengan komponen frekuensi real nol bisa dilihat seperti pada Gambar 2.5 berikut ini.



(a) Nilai real sinyal eksponensial komplek



(b) Nilai imajiner sinyal eksponensial komplek

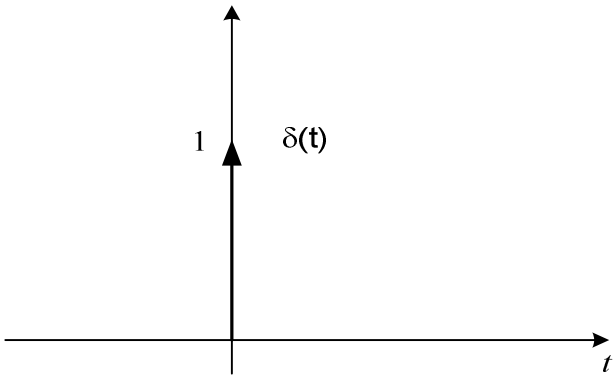
Gambar 2.5. Komponen real dan imajiner sinyal komplek dengan frekuensi real nol

Sinyal Impuls

Sinyal impulse, dalam hal ini adalah fungsi unit impulse δ(t), yang juga dikenal sebagai fungsi Dirac delta atau secara lebih sederhana dinyatakan sebagai fusngi delta function, bisa didefinisikan di dalam terminologi 2 sifat berikut.

- Amplitudo $\delta(t) = 0, \ t \neq 0$
- Area sinyal tertutup $\int_{-\infty}^{\infty} \delta(t) dt = 1$

Penggambaran secara langsung sebuah sinyal impulse pada sinyal waktu kontinyu sebetulnya relatif sulit, yang paling umum digunakan adalah sebuah penyederhanaan. Dengan membentuk garis vertikal dengan panah menghadap ke atas seperti pada Gambar 2.6, diharapkan cukup untuk merepresentasikan sebuah sinyal yang memiliki durasi sangat sempit dan hanya muncul sesaat dengan nilai magnitudo sama dengan 1.



Gambar 2.6. Contoh sinyal impuls

III. Perangkat Yang Diperlukan

- 1 (satu) buah PC lengkap sound card dan OS Windows dan perangkat lunak Matlab
- 1 (satu) flash disk dengan kapasitas yang cukup

IV. Langkah Percobaan

4.1 Pembangkitan Sinyal Waktu Kontinyu Sinusoida

Disini kita mencoba membangkitkan sinyal sinusoida untuk itu coba anda buat program seperti berikut:

```
Fs=100;
t=(1:100)/Fs;
s1=sin(2*pi*t*5);
plot(t,s1)
```

Sinyal yang terbangkit adalah sebuah sinus dengan amplitudo Amp = 1, frekuensi f = 5Hz dan fase awal θ = 0. Diharapkan anda sudah memahami tiga parameter dasar pada sinyal sinus ini. Untuk lebih memahami coba lanjutkan dengan langkah berikut :

1. Lakukan perubahan pada nilai s1:

   ```
   s1=sin(2*pi*t*10);
   ```

   Dan perhatikan apa yang terjadi, kemudian ulangi untuk mengganti angka 10 dengan 15, dan 20. Perhatikan apa yang terjadi, plot hasil percobaan anda.

2. Coba anda edit kembali program anda sehingga bentuknya persis seperti pada langkah1, kemudian lanjutkan dengan melakukan perubahan pada nilai amplitudo, sehingga bentuk perintah pada s1 menjadi:

   ```
   s1=5*sin(2*pi*t*5);
   ```

   Coba perhatikan apa yang terjadi? Lanjutkan dengan merubah nilai amplitudo menjadi 10, 15 dan 20. Apa pengaruh perubahan amplitudo pada bentuk sinyal sinus?

3. Kembalikan program anda sehingga menjadi seperti pada langkah pertama. Sekarang coba anda lakukan sedikit perubahan sehingga perintah pada s1 menjadi:

   ```
   s1=2*sin(2*pi*t*5 + pi/2);
   ```

Coba anda perhatikan, apa yang terjadi? Apa yang baru saja anda lakukan adalah merubah nilai fase awal sebuah sinyal dalam hal ini nilai θ = π/ 2 = 90^o. Sekarang lanjutkan langkah anda dengan merubah nilai fase awal menjadi 45^o, 120^o, 180^o, dan 270^o. Amati bentuk sinyal sinus terbangkit, dan catat hasilnya. Plot semua gambar dalam satu figure dengan perintah subplot.

4.2 Pembangkitan Sinyal Waktu Kontinyu Persegi

Disini akan kita bangkitkan sebuah sinyal persegi dengan karakteristik frekuensi dan amplitudo yang sama dengan sinyal sinus. Untuk melakukannya ikuti langkah berikut ini :

1. Buat sebuah m file baru kemudian buat program seperti berikut ini.

   ```
   Fs=100;
   t=(1:100)/Fs;
   s1=SQUARE(2*pi*5*t);
   plot(t,s1,'linewidth',2)
   axis([0 1 -1.2 1.2])
   ```

Coba anda lakukan satu perubahan dalam hal ini nilai frekuensinya anda rubah menjadi 10 Hz, 15 Hz, dan 20 Hz. Apa yang anda dapatkan? Plot semua gambar dalam satu figure dengan perintah subplot.

3. Kembalikan bentuk program menjadi seperti pada langkah pertama, Sekarang coba anda rubah nilai fase awal menjadi menjadi 45^o, 120^o, 180^o, dan 225^o. Amati dan catat apa yang terjadi dengan sinyal persegi hasil pembangkitan. Plot semua gambar dalam satu figure dengan perintah subplot.

4.3. Pembangkitan Sinyal Dengan memanfaatkan file *.wav

Kita mulai bermain dengan file *.wav. Dalam hal ini kita lakukan pemanggilan sinyal audio yang ada dalam hardisk kita. Langkah yang kita lakukan adalah seperti berikut :

1. Anda buat m file baru, kemudian buat program seperti berikut :

```
y1=wavread('namafile.wav');
Fs=10000;
wavplay(y1,Fs,'async') % Memainkan audio sinyal asli
```

2. Cobalah untuk menampilkan file audio yang telah anda panggil dalam bentuk grafik sebagai fungsi waktu. Perhatikan bentuk tampilan yang anda lihat. Apa yang anda catat dari hasil yang telah anda dapatkan tsb?

4.4. Pembangkitan Sinyal Kontinyu Fungsi Ramp

Sebagai langkah awal kita mulai dengan membangkitkan sebuah fungsi ramp. Sesuai dengan namanya, fungsi ramp berarti adalah tanjakan seperti yang telah ditulis pada persamaan (3). Untuk itu anda ikuti langkah berikut ini. Buat program baru dan anda ketikkan perintah seperti berikut :

```
%Pembangkitan Fungsi Ramp
y(1:40)=1;
x(1:50)=[1:0.1:5.9];
x(51:100)=5.9;
t1=[-39:1:0];
t=[0:1:99];
plot(t1,y,'b',t,x,'linewidt',4)
title('Fungsi Ramp')
xlabel('Waktu (s)')
ylabel('Amplitudo')
```

V. Tugas Selama Praktikum

1. Jawablah setiap pertanyaan yang ada pada setiap langkah percobaan tersebut diatas.
2. Buatlah program untuk menggambarkan “fungsi unit step” dalam m-file (beri nama tugas_1.m).
3. Anda buat pembangkitan sinyal eksponensial dengan suatu kondisi frekuensi realnya adalah nol, dan satu progam lain dimana frekuensi imajinernya nol.
4. Buat pembangkitan sinyal impuls dengan suatu kondisi sinyal terbangkit bukan pada waktu t = 0. Dalam hal ini anda bisa membangkitkan pada waktu t =1 atau 2, atau yang lainnya.

MODUL 3
PEMBANGKITAN SINYAL DISKRIT

I. Tujuan Instruksional Khusus

- Setelah melakukan praktikum ini, diharapkan mahasiswa dapat membangkitkan beberapa jenis sinyal diskrit yang banyak digunakan dalam analisa Sinyal dan Sistem.

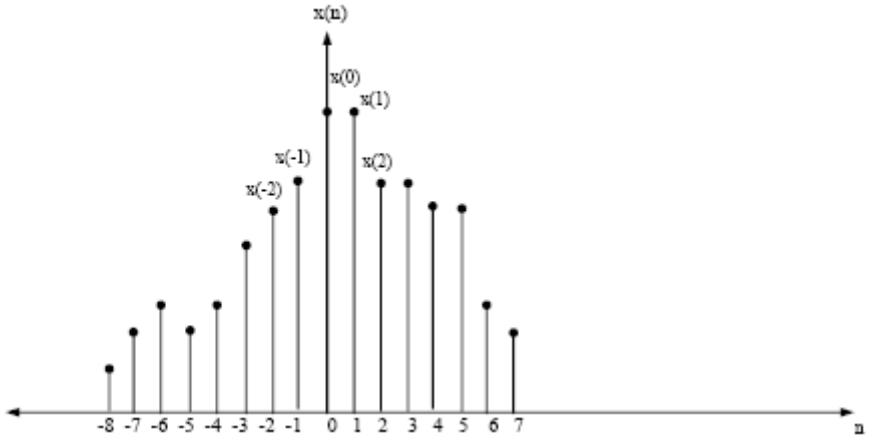
II. Teori Dasar Sinyal Diskrit

2.1. Konsep Sinyal Waktu Diskrit

Sinyal waktu diskrit atau lebih kita kenal sebagai sinyal diskrit memiliki nilai-nilai amplitudo kontinyu (pada suatu kondisi bisa juga amplitudonya diskrit), dan muncul pada setiap durasi waktu tertentu sesuai periode sampling yang ditetapkan. Pada teori system diskrit, lebih ditekankan pada pemrosesan sinyal yang berderetan. Pada sejumlah nilai x, dimana nilai yang *ke-n* pada deret *x(n)* akan dituliskan secara formal sebagai:

$$x = \{x(n)\}; \qquad -\infty < n < \infty \tag{3-1}$$

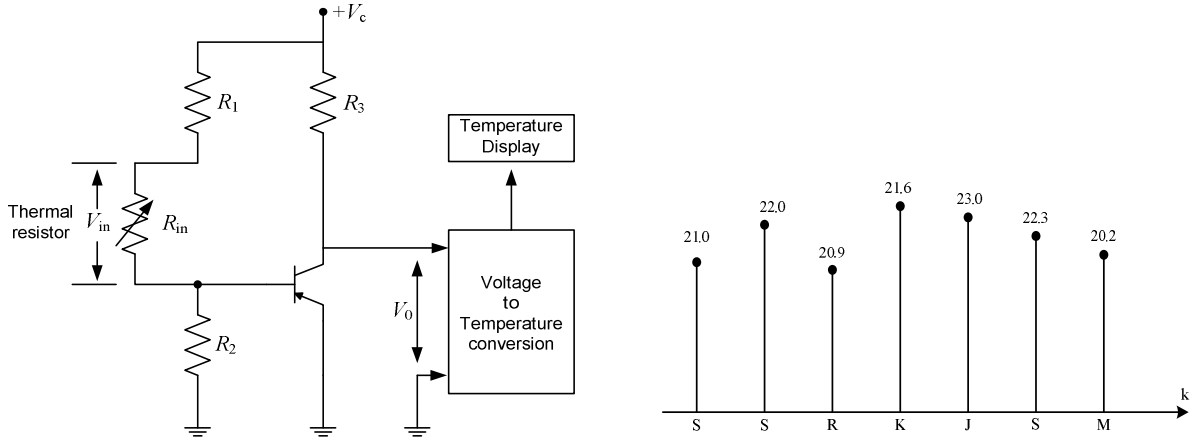
Dalam hal ini *x(n)* menyatakan nilai yang ke-n dari suatu deret, persamaan (3-1) biasanya tidak disarankan untuk dipakai dan selanjutnya sinyal diskrit diberikan seperti Gambar 1. Meskipun absis digambar sebagai garis yang kontinyu, sangat penting untuk menyatakan bahwa *x(n)* hanya merupakan nilai dari *n*. Fungsi *x(n)* tidak bernilai nol untuk *n* yang bukan integer; *x(n)* secara sederhana bukan merupakan bilangan selain integer dari *n*.



Gambar 3.1. Penggambaran secara grafis dari sebuah sinyal waktu diskrit

Sebuah ilustrasi tentang sistem pengambilan data temperatur lingkungan dengan sebuah termometer elektronik bisa dilihat seperti pada Gambar 3.2a. Dalam hal ini rangkaian tersusun dari thermal thermistor yang memiliki perubahan nilai resistansi seusai dengan perubahan temperatur lingkungan sekitarnya. Fluktuasi resistansi digunakan untuk mengukur temperatur yang ada, dan

pengambilan data dilakukan setiap hari. Gambaran data temperatur harian ini bisa diilustrasikan sebagai sebuah sekuen nilai-nilai sinyal waktu diskrit seperti pada Gambar 3.2b.



(a) Sensor monitoring temperatur lingkungan (b) Sinyal ouput diskrit

Gambar 3.2. Sistem sensor temperatur harian dan sinyal outputnya

2.2. Bentuk Dasar Sinyal Waktu Diskrit

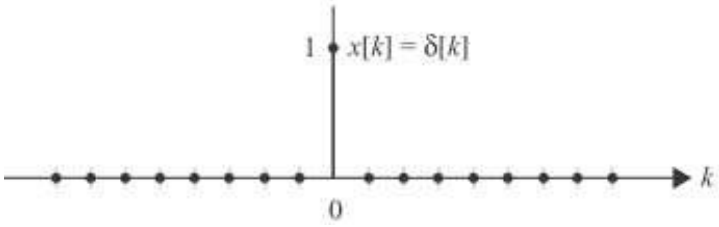
Seperti halnya sinyal waktu kontinyu yang memiliki bentuk-bentuk sinyal dasar, sinyal waktu diskrit juga tersusun dari fungsi dasar sinyal seperti sinyal impulse diskrit, sekuen step, sekuen ramp, sekuen rectangular, sinusoida diskrit dan exponensial diskrit.

Sekuen Impuls

Sinyal impuls waktu diskrit atau sinyal diskrit impulse juga dikenal sebagai suatu Kronecker delta function atau disebut juga sebagai DT unit sample function, didefinisikan dengan persamaan matematik seperti berikut.

$$\delta[k] = u[k] - u[k-1] = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \qquad (2\text{-}2)$$

Sedikit berbeda dengan fungsi impulse pada sinyal waktu kontinyu, fungsi simpulse pada sinyal waktu diskrit tidak memiliki ambiguity pada pendefinisiannya, karena dengan mengacu pada persamaan (2-2) cukup jelas bahwa sinyal ini merupakan sinyal yang hanya sesaat muncul sesuai dengan time sampling yang digunakan. Dan antar satu sampel ke sampel berikutnya ditentukan oleh periode samplingnya. Bentuk fungsi impulse untuk sinyal waktu diskrit bisa dilihat seperti pada Gambar 3.3



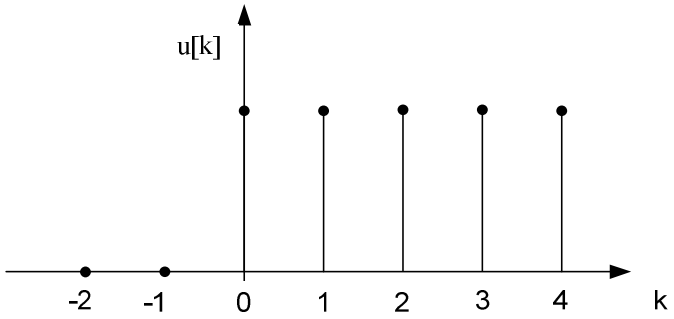
Gambar 3.3 Fungsi impulse sinyal waktu dikrit

Sekuen Step Waktu Diskrit

Sekuen step sinyal waktu diskrit bias direpresentasikan dalam persamaan matematik sebagai berikut:

$$u[k] = \begin{cases} 1 & k \geq 0 \\ 0 & k < 0 \end{cases} \qquad (3\text{-}3)$$

Dimana nilai u[k] akan konstan (bias bernilai 1 atau yang lainnya) setelah waktu k $\geq$ 0. Perbedaan dengan fungsi step waktu kontinyu adalah bahwa dalam sekuen step waktu diskrit, sinyal akan memiliki nilai pada setiap periode waktu tertentu, sesuai dengan periode samping yang digunakan. Bentuk sekuen step waktu diskrit bias dilihat seperti pada Gambar 3.4.



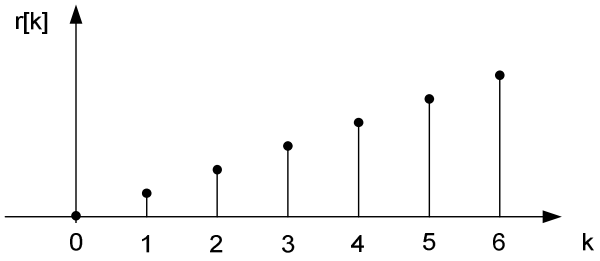
Gambar 3.4. Sekuen step waktu diskrit

Fungsi Ramp Diskrit

Seperti pada pembahasan sinyal waktu kontinyu, fungsi ramp untuk sinyal waktu diskrit bias dinyatakan dalam persamaan matematik sebagai berikut:

$$r[k] = ku[k] = \begin{cases} k & k \geq 0 \\ 0 & k < 0 \end{cases} \qquad (3\text{-}4)$$

Contoh sebuah sekuen fungsi ramp waktu diskrit dengan kemiringan (slope) bernilai k > 0 bisa dilihat seperti pada Gambar 3.5.



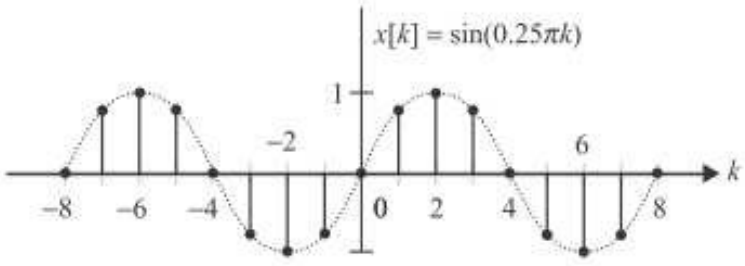
Gambar 3.6. Sekuen ramp waktu diskrit

Sinusoida Diskrit

Sinusoida diskrit bisa direpresentasikan dalam persamaan matematik sebagai berikut:

$$x[k] = \sin(\Omega_0 k+\theta) = \sin(2\pi f_0 k+\theta) \quad (3\text{-}5)$$

dimana Ω_0 adalah frekuensi angular pada waktu diskrit. Sinusoida diskrit bias dilihat seperti pada Gambar 2.xx. Di dalam pembahasan pada sinyal sinusoida waktu kontinyu, dinyatakan bahwa sinyal sinusoida sinyal $x(t) = \sin(\omega_0 t+\theta)$ selalu periodiks. Sementara di dalam sinyal waktu diskrit, sinyal sinusoida akan memenuhi kondisi periodic jika dan hanya jika nilai $\Omega_0/2\pi$ merupakan bilangan bulat.



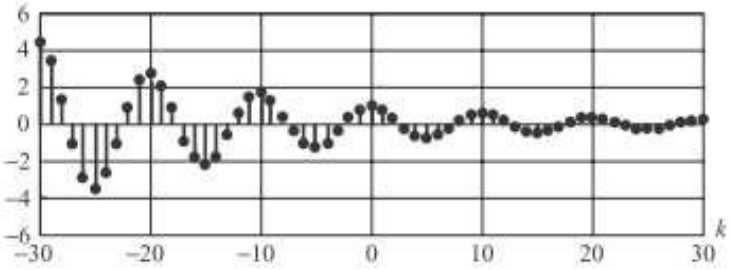
Gambar 3.7. Sinyal sinusoida waktu diskrit.

Fungsi Eksponensial Diskrit

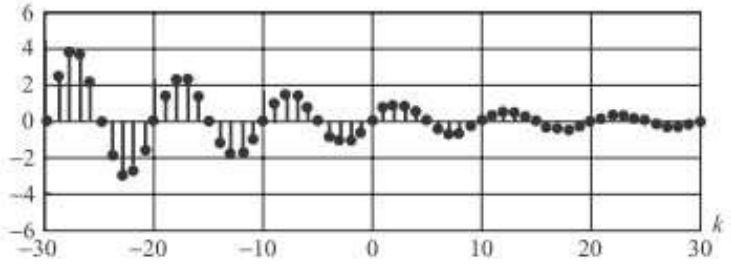
Fungsi eksponensial waktu diskrit dengan sebuah frekueni sudut sebesar Ω_0 didefinisikan sebagai berikut:

$$x[k] = e^{(\sigma + j\Omega_0)k} = e^{\sigma k}\left(\cos\Omega_0 k + j\sin\Omega_0 k\right) \quad (3\text{-}6)$$

Sebagai contoh fungsi sinyal eksponensial waktu diskrit, kita pertimbangkan sebuah fungsi eksponensial x[k] =exp(j0.2π − 0.05k), yang secara grafis bisa disajikan seperti pada Gambar 3.xx, dimana gabian (a) menunjukkan komponen real dan bagian (b) menunjukkan bagian imajiner pada sinyal komplek tersebut.



(a) Bagian real sinyal komplek eksponensial



(b) Bagian imajiner sinyal komplek eksponensial

Gambar 3.8. Ilustrasi sinyal eksponensial komplek waktu diskrit

III. Perangkat Yang Diperlukan

- 1 (satu) buah PC lengkap sound card dan OS Windows dan Perangkat Lunak Matlab
- 1 (satu) flash disk dengan kapasitas minimal 1 G

IV. Langkah Percobaan

4.1 Pembangkitan Sinyal Waktu Diskrit, Sekuen Step

Disini akan kita lakukan pembangkitan sinyal waktu diskrit. Sebagai langkah awal kita mulai dengan membangkitkan sebuah sekuen unit step. Sesuai dengan namanya, unit step berarti nilainya adalah satu satuan. Untuk itu anda ikuti langkah berikut ini.

1. Buat program baru dan anda ketikkan perintah seperti berikut:

```
%File Name: sd_1.m
%Pembangkitan  Sekuen Step
L=input('Panjang Gelombang (=40) =' )
P=input('Panjang Sekuen (=5) =' )
for n=1:L
if (n>=P)
        step(n)=1;
    else
```

```
        step(n)=0;
    end
    end
    x=1:L;
    stem(x,step)
```

Berikan penjelasan pada gambar yang dihasilkan.

2. Anda ulangi langkah pertama dengan cara me-*run* program anda dan masukan nilai untuk panjang gelombang dan panjang sekuen yang berbeda-beda yaitu L=40, P= 15 ; L=40, P=25 ; L=40, P=35. Plot hasil percobaan anda pada salah satu figure, dan catat apa yang terjadi?

4.2 Pembangkitan Sinyal Waktu Diskrit, Sekuen Pulsa

Disini akan kita bangkitkan sebuah sinyal waktu diskrit berbentuk sekuen pulsa, untuk itu ikuti langkah berikut ini

1. Buat program baru dengan perintah berikut ini.

```
%File Name: Sd_2.m
%Pembangkitan Sekuen Pulsa
L=input('Panjang Gelombang (=40) =' )
P=input('Posisi Pulsa (=5) =' )
for n=1:L
    if (n==P)
  step(n)=1;
else
  step(n)=0;
end
end
x=1:L;
stem(x,step)
axis([0 L -.1 1.2])
```

Berikan penjelasan pada gambar yang dihasilkan.

2. Jalankan program diatas berulang-ulang dengan catatan nilai L dan P dirubah-ubah sebagai berikut L=40, P= 15 ; L=40, P=25 ; L=40, P=35, perhatikan apa yang terjadi? Catat apa yang anda lihat.

4.3 Pembentukan Sinyal Sinus waktu Diskrit

Pada bagian ini kita akan dicoba untuk membuat sebuah sinyal sinus diskrit. Secara umum sifat dasarnya memiliki kemiripan dengan sinus waktu kontinyu. Untuk itu ikuti langkah berikut

1. Buat program baru dengan perintah seperti berikut.

```
%File Name: Sd_4.m
Fs=20;%frekuensi sampling
t=(0:Fs-1)/Fs;%proses normalisasi
s1=sin(2*pi*t*2);
stem(t,s1)
axis([0 1 -1.2 1.2])
```

2. Lakukan perubahan pada nilai Fs, sehingga bernilai 40, 60 dan 80. Plot hasil percobaan anda pada satu figure, dan catat apa yang terjadi.

4.4 Pembangkitan Sinyal Waktu Diskrit, Sekuen konstan

Disini akan kita bangkitkan sebuah sinyal waktu diskrit berbentuk sekuen pulsa, untuk itu ikuti langkah berikut ini

1. Buat program baru dengan perintah berikut ini.

```
%File Name: Sd_4.m
%Pembangkitan Sekuen Konstan
L=input('Panjang Gelombang (=20) =' )
sekuen(1:L)=1; % Besar Amlitudo
stem(sekuen)
xlabel('Jumlah Sekuen (n)')
ylabel('Amplitudo sekuen')
title('Sinyal Sekuen Konstan')
```

Berikan penjelasan pada gambar yang dihasilkan.

V. DATA DAN ANALISA

Anda telah melakukan berbagai langkah untuk percobaan pembangkitan sinyal diskrit. Langkah selanjutnya yang harus anda lakukan adalah:

1. Jawab setiap pertanyaan yang ada pada setiap langkah percobaan diatas.
2. Coba anda buat program pada m-file untuk membangkitkan sebuah sinyal sekuen rectanguler (persegi) yang berada pada posisi 1-4 , 2-6, 4-8 dan 6-10 dengan amplitudo sebesar 5. Plot hasil perconaan dalam 1 figure. Beri komentar bagaimana pengaruh perubahan posisi sinyal rectanguler yang telah anda coba?

MODUL 4
OPERASI DASAR SINYAL 1 (OPERASI DENGAN VARIABEL TAK BEBAS)

I. Tujuan Instruksional Khusus:

- Mahasiswa dapat memperlihatkan proses-proses aritmatika sinyal seperti penguatan,pelemahan perkalian, penjumlahan dan pengurangan serta dapat menerapkan sebagai proses dasar dari pengolah sinyal.

II. Operasi Dasar Pada Sinyal dengan Variabel Tak Bebas

Masalah mendasar pada proses pembelajaran pengolahan sinyal adalah bagaimana menggunakan sistem untuk melakukan pengolahan atau melakukan manipulasi terhadap sinyal. Pembicaraan ini biasanya melibatkan kombinasi pada beberapa operasi dasar. Ada dua kelas di dalam operasi dasar sinyal. Yang pertama adalah operasi yang dibentuk oleh variabel tidak bebas yang meliputi amplitude scaling (penguatan / pelemahan), addition, mutiplication. Yang kedua adalah operasi yang dibentuk dengan variabel bebas, meliputi time scaling, reflection, dan time shifting

.

2.1. Amplitude Scaling

Apmlitude Scaling bisa berupa penguatan jika faktor pengali lebih besar dari 1, atau menjadi pelemahan jika faktor pengali kurang dari 1.

Penguatan Sinyal

Peristiwa penguatan sinyal seringkali diumpai pada perangkat audio seperti radio, tape, bahkan pada transmisi gelombang radio yang berkaitan dengan multipath, dimana masing-masing sinyal datang dari Tx ke Rx akan saling menguatkan apabila fase sinyal sama. Fenomena ini dapat juga direpresentasikan secara sederhana sebagai sebuah operasi matematika sebagai berikut:

$$y(t) = amp\ x(t) \qquad (4\text{-}1)$$

dimana:

y(t) = sinyal output

amp = konstanta penguatan sinyal

x(t) = sinyal input

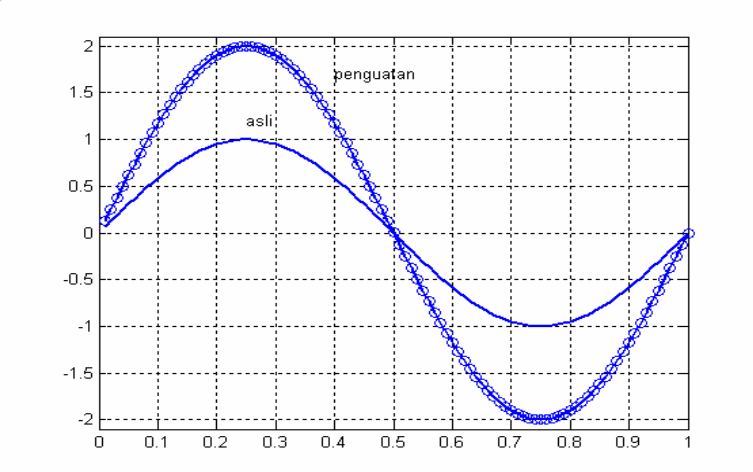
Bentuk diagram blok dari sebuah operasi pernguatan sinyal dapat diberikan pada gambar berikut ini.

Gambar 4.1. Diagram blok penguatan suatu sinyal

Besarnya nilai konstanta sinyal amp >1, dan penguatan sinyal seringkali dinyatalan dalam besaran deci Bell, yang didefinisikan sebagai:

$$amp\_dB = 10 \log(output/input) \quad (4\text{-}2)$$

Dalam domain waktu, bentuk sinyal asli dan setelah mengalami penguatan adalah seperti gambar berikut.

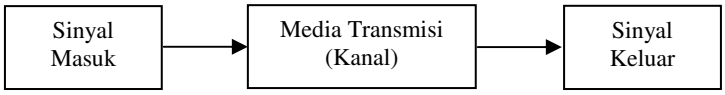


Gambar 4.2. Penguatan Sinyal

Pelemahan Sinyal

Apabila sebuah sinyal dilewatkan suatu medium seringkali mengalami berbagai perlakuan dari medium (kanal) yang dilaluinya. Ada satu mekanisme dimana sinyal yang melewati suatu medium mengalami pelemahan energi yang selanjutnya dikenal sebagai atenuasi (pelemahan atau redaman) sinyal.

Bentuk diagram blok dari sebuah operasi pernguatan sinyal dapat diberikan pada gambar berikut ini.

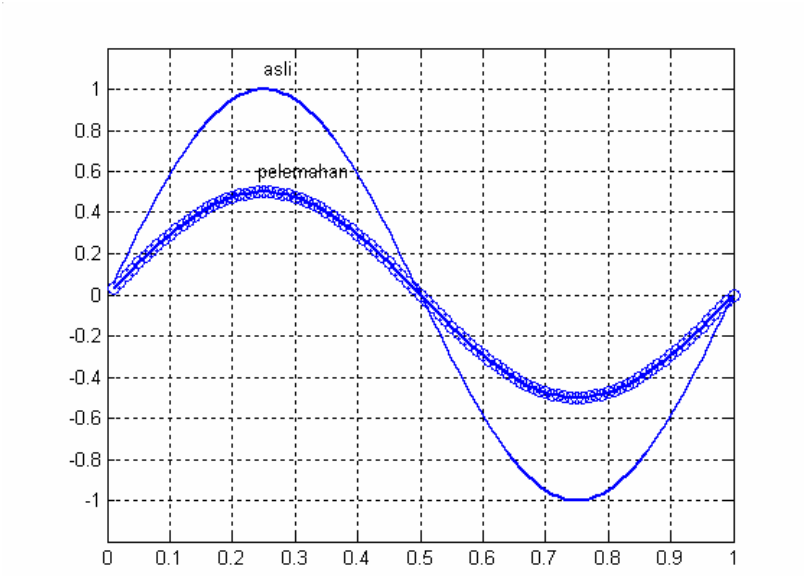


Gambar 4.3 Operasi Pelemahan suatu sinyal

Dalam bentuk operasi matematik sebagai pendekatannya, peristiwa ini dapat diberikan sebagai berikut:

$$y(t) = att\ x(t) \quad (4\text{-}3)$$

Dalam hal ini nilai $att < 1$, yang merupakan konstanta pelemahan yang terjadi. Kejadian ini sering muncul pada sistem transmisi, dan munculnya konstanta pelemahan ini dihasilkan oleh berbagai proses yang cukup komplek dalam suatu media transmisi.

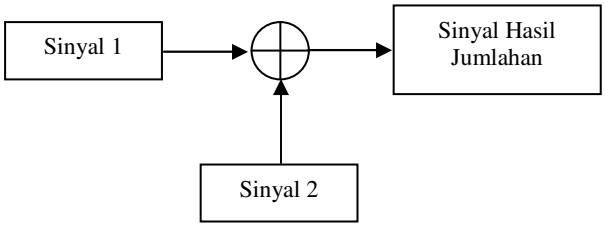


Gambar 4.4. Pelemahan Sinyal

Dari gambar tersebut dapat dilihat bahwa proses penguatan dan pelemahan sinyal merupakan dua hal yang hampir sama. Dalam pengatan sinyal amplitudo sinyal output lebih tinggi dibanding sinyal input, sementara pada pelemahan sinyal amplitudo sinyal output lebih rendah disbanding sinyal input. Tetapi pada kedua proses operasi ini bentuk dasar sinyal tidak mengalami perubahan.

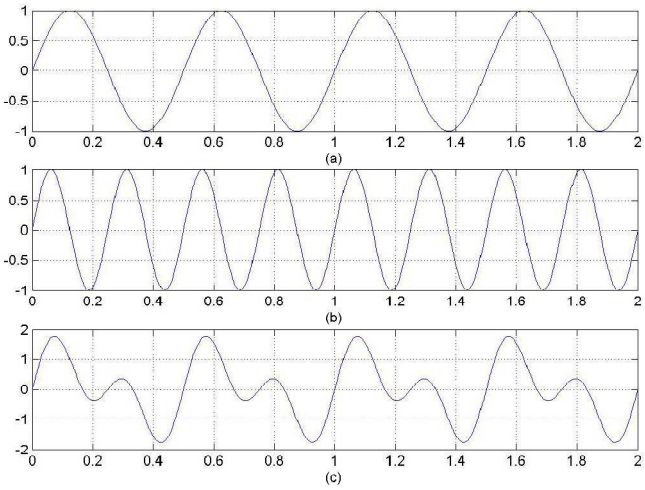
2.2. Addition

Proses penjumlahan sinyal seringkali terjadi pada peristiwa transmisi sinyal melalui suatu medium. Sinyal yang dikirimkan oleh pemancar setelah melewati medium tertentu misalnya udara akan mendapat pengaruh kanal, pengaruh tersebut tentunya akan ditambahkan pada sinyal aslinya. Misal Sinyal informasi yang terpengaruh oleh noise atau sinyal lain dari kanal, maka secara matematis pada sinyal tersebut pasti terjadi proses penjumlahan. Sehingga pada bagian penerima akan mendapatkan sinyal sebagai hasil jumlahan sinyal asli dari pemancar dengan sinyal yang terdapat pada kanal tersebut.



Gambar 4.5. Diagram blok operasi penjumlahan dua sinyal.

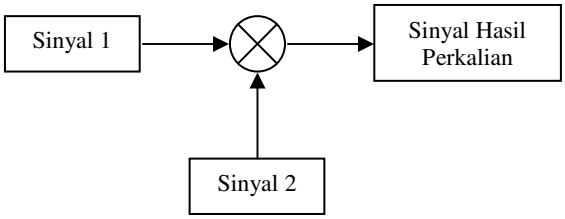
Contoh penjumlahan dari 2 buah sinyal dengan amplitudo sama tetapi frekuensi berbeda bisa dilihat pada Gambar 4.6.



Gambar 4.6. Contoh operasi penjumlahan dua sinyal.

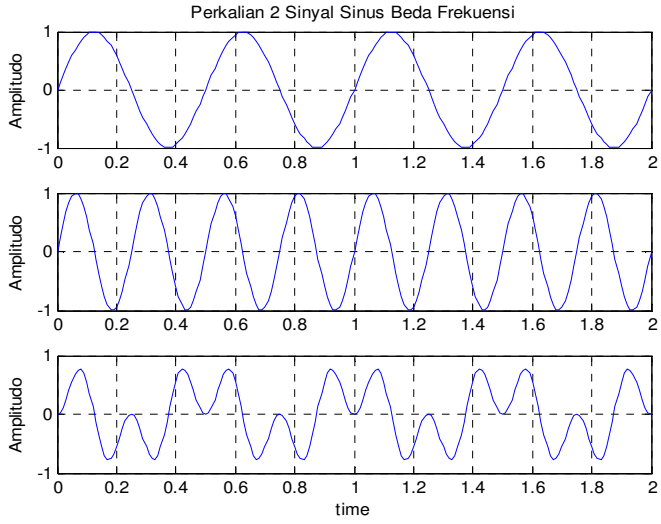
2.3. Multiplication

Perkalian merupakan bentuk operasi yang sering anda jumpai dalam kondisi real. Pada rangkaian *mixer*, rangkaian *product modulator frequency multiplier*, proses windowing pada *speech processing,* sehingga operasi perkalian sinyal merupakan bentuk standar yang seringkali dijumpai. Bentuk diagram blok operasi perkalian dua buah sinyal dapat diberikan seperti pada Gambar 4.7.



Gambar 4.7. Diagram blok operasi perkalian dua sinyal.

Contoh perkalian dua sinyal beda frekuensi, bisa dilihat pada Gambar 4.8.



Gambar 4.8. Contoh Perkalian 2 Sinyal Frekuensi Berbeda

III. Perangkat Yang Diperlukan

- 1 (satu) buah PC lengkap sound card dan OS Windows dan perangkat lunak Matlab
- 1 (satu) flash disk dengan kapasitas 1 G atau lebih

IV. Langkah Percobaan

4.1. Penguatan Sinyal

1. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;
t=0:1/T:2;
f1=1;
y1=sin(2*pi*t);
subplot(2,1,1)
plot(t,y1)
```

2. Lanjutkan dengan langkah berikut ini

```
a=input('nilai pengali yang anda gunakan (>1): ');
y1_kuat=a*sin(2*pi*t);
subplot(2,1,2)
plot(t,y1_kuat)
```

Masukan nilai ‘a’ berturut-turut : 1.5 ; 4; 5.5 dan 8. Apa yang anda dapatkan? (beri penjelasan secara lengkap). Plot setiap hasil running pada satu figure. Nilai penguatan sinyal juga seringkali dituliskan dalam desi-Bell (dB), untuk penguatan 1.5 kali berapa nilainya dalam dB?

3. Ulangi langkah 1 dan 2, tetapi dengan nilai y1 dany1_kuat dalam besaran dB. Dan plot gambar (dalam dB) dan buatlah analisa dari apa yang anda amati dari gambar tersebut? Jangan lupa dalam setiap penggambaran anda cantumkan nilai dB setiap percobaan.

4.2 Proses Penguatan pada Sinyal Audio

Sekarang dilanjutkan dengan file *.wav. Dalam hal ini akan dilakukan penguatan sinyal audio yang telah dipanggil. Langkah yang akan dilakukan adalah seperti berikut :

1. Anda buat file kuat_1.m seperti berikut

```
%File Name: audio_1.m
%Description: how to read and play a wav file
y1=wavread('audio1.wav');
Fs=8192;
wavplay(y1,Fs,'async') % Memainkan audio sinyal asli
subplot(211)
plot(y1)
```

2. Lakukan penambahan perintah seperti dibawah ini

```
amp =1.5;
y2=amp*y1;
wavplay(y1,Fs,'async') % Memainkan audio sinyal setelah penguatan
subplot(212)
plot(y2)
```

3. Apakah anda mengamati sesuatu yang baru pada sinyal audio anda? Cobalah anda rubah nilai amp sebesar 2, 4, 8 dan 10.
4. Plot file audio yang telah anda panggil dalam bentuk grafik sebagai fungsi waktu, baik untuk sinyal asli maupun setelah penguatan dalam satu figure.
5. Tambahkan noise pada sinyal audio dengan perintah sebagai berikut :

```
var = 0.1;
N=length(y1) ;
noise_1=var*randn(N,1);%membangkitkan noise Gaussian
y_1n=y1 + noise_1;%menambahkan noise ke file
subplot(311)
plot(y_1n)
```

6. Lakukan penguatan 5 kali pada sinyal y_1n, tetapi pada sinyal aslinya saja, dengan perintah:

```
y_1n_kuat_sinyal=5*y1 + noise_1;%menambahkan noise ke file
subplot(312)
  plot(y_1n_kuat_sinyal)
```

7. Kemudian kuatkan sinyal yang sudah diberi noise tersebut sebesar 5 kali, dengan perintah:

```
y_1n_kuat_semua=5*(y1 + noise_1);%menambahkan noise ke file
subplot(313)
plot(y_1n_kuat_semua)
```

Bandingkan gambar 1, 2 dan 3, kemudian berikanlah analisa sesuai teori sinyal yang telah anda pelajari.

4.3 Pelemahan Sinyal

1. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;
t=0:1/T:2;
f1=1;
y1=sin(2*pi*t);
subplot(2,1,1)
plot(t,y1)
```

2. Lanjutkan dengan langkah berikut ini

```
a=input('nilai pengali yang anda gunakan (<1): ');
y1_lemah=a*sin(2*pi*t);
subplot(2,1,2)
plot(t,y1_kuat)
```

Masukan nilai 'a' berturut-turut : 0.2 ; 0.5 ; 0.7 dan 0.9. Apa yang anda dapatkan? (berikan penjelasan secara lengkap)

4.4 Proses Pelemahan pada Sinyal Audio

Sekarang dilanjutkan dengan file *.wav. Dalam hal ini akan dilakukan pelemahan sinyal audio yang telah dipanggil. Langkah yang akan dilakukan adalah seperti berikut:

1. Anda buat file lemah_1.m seperti berikut

```
%Description: how to read and play a wav file
y1=wavread('audio1.wav');
Fs=8192;
wavplay(y1,Fs,'async') % Memainkan audio sinyal asli
subplot(211)
plot(y1)
```

2. Lakukan penambahan perintah seperti dibawah ini

```
amp =0.5;
y2=amp*y1;
wavplay(y1,Fs,'async') % Memainkan audio sinyal setelah pelemahan
subplot(212)
plot(y2)
```

3. Apakah anda mengamati sesuatu yang baru pada sinyal audio anda? Cobalah anda rubah nilai amp senilai 0.2, 0.4, 0.6 dan 0.8.
4. Plot file audio yang telah anda panggil dalam bentuk grafik sebagai fungsi waktu, baik untuk sinyal asli maupun setelah pelemahan dalam satu figure.
5. Tambahkan noise pada sinyal audio dengan perintah sebagai berikut:

```
var = 0.1;
N=length(y1) ;
noise_1=var*randn(N,1);%membangkitkan noise Gaussian
y_1n=y1 + noise_1;%menambahkan noise ke file
subplot(311)
plot(y_1n)
```

6. Lakukan pelemahan 0.5 kali pada sinyal y_1n, tetapi pada sinyal aslinya saja, dengan perintah:

```
y_1n_lemah_sinyal=0.5*y1 + noise_1;%menambahkan noise ke file
subplot(312)
plot(y_1n_kuat_sinyal)
```

8. Kemudian lemahkan sinyal yang sudah diberi noise tersebut sebesar 0.5 kali, dengan perintah:

```
y_1n_lemah_semua=0.5*(y1 + noise_1);%menambahkan noise ke file
subplot(313)
plot(y_1n_lemah_semua)
```

Bandingkan gambar 1, 2 dan 3, kemudian berikanlah analisa sesuai teori sinyal yang telah anda pelajari.

4.1 Perkalian Dua Sinyal

Operasi perkalian duabuah sinyal dapat dilakukan dengan mengikuti langkah-langkah berikut:

1. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;%banyak sampel
t=0:1/T:2;%time
f=1; %frekuensi
a1=4 ; %amplitudo sinyal
pha=pi/2;
y1=a1*sin(2*pi*f*t+pha);
subplot(3,1,1)
plot(t,y1)
```

2. Bangkitkan gelombang kedua dengan langkah tambahan berikut ini:

```
a2=4 ;
y2=a2*sin(2*pi*f*t+pha);
subplot(3,1,2)
plot(t,y2)
```

3. Lakukan proses perkalian pada kedua sinyal y1 dan y2 diatas. Selengkapnya bentuk programnya adalah seperti berikut:

```
y3=y1*y2 ;
subplot(3,1,3)
plot(t,y3)
```

Buat program perkalian sinyal (1s/d3) tersebut diatas dalam satu m-file, program diatas adalah hasil perkalian dua buah sinyal dengan frekuensi dan beda fase sama tetapi amplitudonya berbeda. Plot hasil Running program tersebut, kesimpulan apa yang dapat diambil dari hasil percobaan tersebut, jelaskan sebagai bahan analisa.

Selanjutnya buat program dalam m-file yang berbeda sebagai berikut:

4. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;%banyak sampel
t=0:1/T:2;%time
f1=1; %frekuensi
a=4 ; %amplitudo sinyal
pha=pi/2;
y1=a*sin(2*pi*f1*t+pha);
subplot(3,1,1)
plot(t,y1)
```

5. Bangkitkan gelombang kedua dengan langkah tambahan berikut ini:

```
f2=2 ;
y2=a*sin(2*pi*f2*t+pha);
subplot(3,1,2)
plot(t,y2)
```

6. Lakukan proses perkalian pada kedua sinyal y1 dan y2 diatas. Selengkapnya bentuk programnya adalah seperti berikut:

```
y3=y1*y2 ;
subplot(3,1,3)
plot(t,y3)
```

Buat program perkalian sinyal (4s/d6) tersebut diatas dalam satu m-file, program diatas adalah hasil perkalian dua buah sinyal dengan amplitudo dan beda fase sama tetapi frekuensinya berbeda. Plot hasil Running program tersebut, kesimpulan apa yang dapat diambil dari hasil percobaan tersebut, jelaskan sebagai bahan analisa.

Selanjutnya buat program dalam m-file yang berbeda sebagai berikut :

7. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;%banyak sampel
t=0:1/T:2;%time
f=1; %frekuensi
a=4 ; %amplitudo sinyal
pha1=pi/2;
y1=a*sin(2*pi*f*t+pha1);
subplot(3,1,1)
plot(t,y1)
```

8. Bangkitkan gelombang kedua dengan langkah tambahan berikut ini:

```
Pha2=2pi/3;
```

```
y2=a*sin(2*pi*f*t+pha2);
subplot(3,1,2)
plot(t,y2)
```

9. Lakukan proses perkalian pada kedua sinyal y1 dan y2 diatas. Selengkapnya bentuk programnya adalah seperti berikut:

```
y3=y1*y2 ;
subplot(3,1,3)
plot(t,y3)
```

Buat program perkalian sinyal (7s/d9) tersebut diatas dalam satu m-file, program diatas adalah hasil perkalian dua buah sinyal dengan amplitudo dan beda fase sama tetapi frekuensinya berbeda. Plot hasil Running program tersebut, kesimpulan apa yang dapat diambil dari hasil percobaan tersebut, jelaskan sebagai bahan analisa.

4.2 Penjumlahan Dua Sinyal

Operasi penjumlahan duabuah sinyal dapat dilakukan dengan mengikuti langkah berikut:

1. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;%banyak sampel
t=0:1/T:2;%time
f=1; %frekuensi
a1=4 ; %amplitudo sinyal
pha=pi/2;
y1=a1*sin(2*pi*f*t+pha);
subplot(3,1,1)
plot(t,y1)
```

2. Bangkitkan gelombang kedua dengan langkah tambahan berikut ini:

```
a2=4 ;
y2=a2*sin(2*pi*f*t+pha);
subplot(3,1,2)
plot(t,y2)
```

3. Lakukan proses penjumlahan pada kedua sinyal y1 dan y2 diatas. Bentuk program selengkapnya adalah seperti berikut:

```
y3=y1+y2 ;
subplot(3,1,3)
plot(t,y3)
```

Buat program penjumlahan sinyal (1s/d3) tersebut diatas dalam satu m-file, program diatas adalah hasil penjumlahan dua buah sinyal dengan frekuensi dan beda fase sama tetapi amplitudonya

berbeda. Plot hasil Running program tersebut, kesimpulan apa yang dapat diambil dari hasil percobaan tersebut, jelaskan sebagai bahan analisa.

Selanjutnya buat program dalam m-file yang berbeda sebagai berikut:

4. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;%banyak sampel
t=0:1/T:2;%time
f1=1; %frekuensi
a=4 ; %amplitudo sinyal
pha=pi/2;
y1=a*sin(2*pi*f1*t+pha);
subplot(3,1,1)
plot(t,y1)
```

5. Bangkitkan gelombang kedua dengan langkah tambahan berikut ini:

```
f2=2 ;
y2=a*sin(2*pi*f2*t+pha);
subplot(3,1,2)
plot(t,y2)
```

6. Lakukan proses penjumlahan pada kedua sinyal y1 dan y2 diatas. Selengkapnya bentuk programnya adalah seperti berikut:

```
y3=y1+y2 ;
subplot(3,1,3)
plot(t,y3)
```

Buat program penjumlahan sinyal (4s/d6) tersebut diatas dalam satu m-file, program diatas adalah hasil penjumlahan dua buah sinyal dengan amplitudo dan beda fase sama tetapi frekuensinya berbeda. Plot hasil Running program tersebut, kesimpulan apa yang dapat diambil dari hasil percobaan tersebut, jelaskan sebagai bahan analisa.

Selanjutnya buat program dalam m-file yang berbeda sebagai berikut:

7. Bangkitkan gelombang pertama dengan langkah berikut:

```
T=100;%banyak sampel
t=0:1/T:2;%time
f=1; %frekuensi
a=4 ; %amplitudo sinyal
pha1=pi/2;
y1=a*sin(2*pi*f*t+pha1);
subplot(3,1,1)
```

```
plot(t,y1)
```

8. Bangkitkan gelombang kedua dengan langkah tambahan berikut ini:

```
Pha2=2pi/3;
y2=a*sin(2*pi*f*t+pha2);
subplot(3,1,2)
plot(t,y2)
```

9. Lakukan proses penjumlahan pada kedua sinyal y1 dan y2 diatas. Bentuk program selengkapnya adalah seperti berikut:

```
y3=y1+y2 ;
subplot(3,1,3)
plot(t,y3)
```

Buat program penjumlahan sinyal (7s/d9) tersebut diatas dalam satu m-file, program diatas adalah hasil penjumlahan dua buah sinyal dengan amplitudo dan beda fase sama tetapi frekuensinya berbeda. Plot hasil Running program tersebut, kesimpulan apa yang dapat diambil dari hasil percobaan tersebut, jelaskan sebagai bahan analisa.

Penambahan Noise Gaussian pada Sinyal Audio

Untuk melakukan proses penambahan pada file.wave, file wave tersebut harus berada dalam satu folder dengan m-file yang akan digunakan untuk memprosesnya. Untuk itu coba anda cari file *.wav apa saja yang ada di PC anda, copykan ke folder dimana Matlab anda bekerja.

1. Untuk contoh kasus ini ikuti langkah pertama dengan membuat file coba_audio_1.m seperti berikut.

```
%File Name:coba_audio_1.m
y1=wavread('audio3.wav');
Fs=8192;
Fs1 = Fs;
wavplay(y1,Fs1,'sync') % Sinyal asli dimainkan
```

2. Tambahkan perintah berikut ini setelah langkah satu diatas.

```
N=length(y1);%menghitung dimensi file wav
var = 0.1;
noise_1=var*randn(N,1);%membangkitkan noise Gaussian
y_1n=y1 + noise_1;%menambahkan noise ke file
wavplay(y_1n,Fs1,'sync') % Sinyal bernoise dimainkan
```

3. Apakah anda melihat ada sesuatu yang baru dengan langkah anda? Coba anda lakukan sekali lagi langkah 2 dengan nilai var 0.2, 0.4, 06, dst. Coba amati apa yang terjadi?
4. Cobalah untuk menampilkan file audio yang telah anda panggil dalam bentuk grafik sebagai fungsi waktu, baik untuk sinyal asli atau setelah penambahan noise dalam satu figure.

V. Tugas Selama Praktikum

Penguatan dan Pelemahan Sinyal

Anda telah melakukan berbagai langkah untuk percobaan operasi dasar sinyal. Yang harus anda lakukan adalah menjawab setiap pertanyaan yang ada pada langkah percobaan. Tulis semua komentar dan analisa sebagai penjelasan dari hasil percobaan anda.

1. Apa arti penguatan dan pelemahan sinyal dalam simulasi tersebut diatas ? jelaskan berdasarkan amplitudonya.
2. Jelaskan pengaruh penguatan dan pelemahan sinyal pada sinyal yang ditambah dengan noise ?

Pernjumlahan dan Perkalian Sinyal

1. Buat program perkalian 2 buah sinyal dengan berbagai perubahan (besar perubahan terserah anda masing-masing) dengan ketentuan :

- Amplitudo berbeda, frekuensi dan beda fase tetap.
- Frekuensi berbeda, amplitudo dan beda fase tetap
- Beda fase berbeda, amplitudo dan frekuensi tetap.

Jalankan program dan plot masing-masing m-file, serta jelaskan sebagai analisa tentang hasil proses perkalian du sinyal.

2. Ulangi tugas 1 untuk proses penjumlahan dan pengurangan dua buah sinyal.

MODUL 5
OPERASI DASAR SINYAL 2 (OPERASI DENGAN VARIABEL BEBAS)

I. Tujuan Instruksional Khusus:

- Mahasiswa dapat memperlihatkan proses-proses aritmatika sinyal seperti *time shifting*, *time scaling* dan *reflection* sebagai proses dasar dari pengolah sinyal.

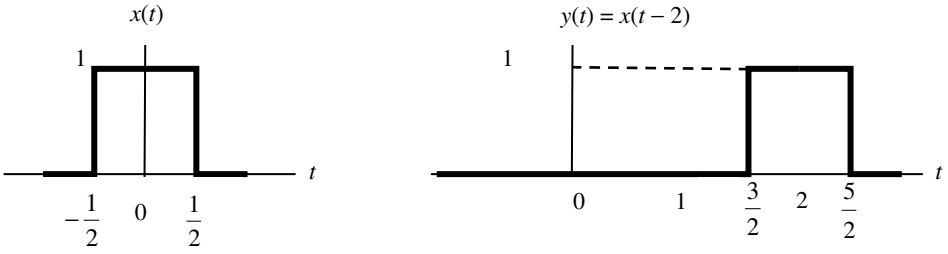
II. Operasi Sinyal Yang dibentuk dengan Variabel Bebas

2.1. Time Shifting

Kita tetapkan x(t) sebagai suatu sinyal waktu kontinyu. Selanjutnya kita tetapkan bahwa y(t) sebagai output dari suatu operasi pegeseran waktu, dan mendefinisikannya sebagai:

$$y(t) = x(t - to) \qquad (5\text{-}1)$$

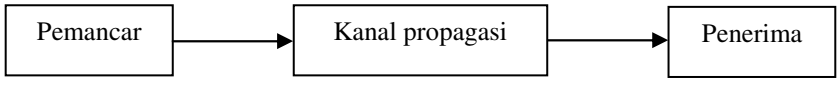
Sehingga kita dapatkan bahwa y(t) merupakan sebuah versi tergeser waktu dari x(t), dan dalam hal ini *t*o merupakan besarnya pergeseran. Jika nilai *t*o > 0, kita akan mendapatkan bentuk pergeseran sinyal ke kanan, sedangkan jika nilai *t*o < 0 akan diperoleh bentuk pergeseran ke kiri.



(a) Sinyal Asli (b) Sinyal tergeser waktu

Gambar 5.1. Operasi pergeseran waktu (time shifting)

Di dalam bidang telekomunikasi, operasi pergeseran bisa digunakan untuk merepresentasikan sebuah proses delay propagasi sinyal. Sebuah gelombang radio dari pemancar dikirimkan pada t = 0, untuk sampai ke penerima yang cukup jauh, kira-kira 300 meter maka bagian penerima akan menangkap sinyal tersebut dalam bentuk versi sinyal tertunda selama $\pm 1\mu$ detik (10^{-6} detik). Tentu saja bukan sinyal tersebut juga mengalami proses pelemahan, dan mungkin juga mangalami bentuk gangguan yang lainnya.



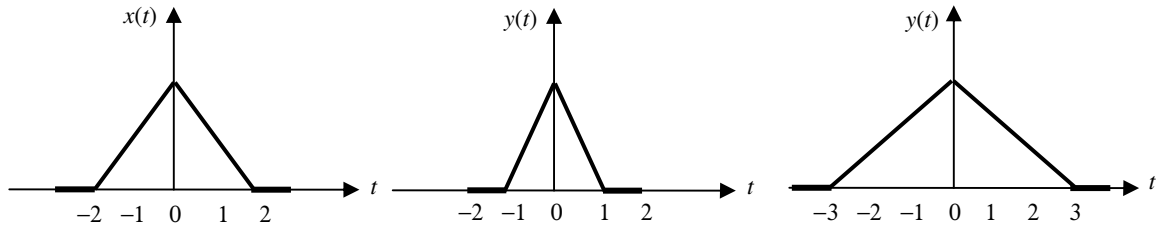
Gambar 5.2. Contoh kejadian pergeseran sinyal pada propagasi

2.2. Time Scaling

Kita tetapkan $x(t)$ sebagai sebuah sinyl waktu kontinyu, selanjutnya anda tetapkan bahwa $y(t)$ adalah output dari sebuah proses pensekalaan yang dilakukan dengan variable bebas, dalam hal ini waktu, t dengan sebuah factor penskalaan bernilai a. Maka hubungan antara $y(t)$ dan $x(t)$ dapat dinyatakan di dalam persamaan:

$$y(t) = x(at) \tag{5-2}$$

Jika $a > 1$, sinyal $y(t)$ akan memiliki bentuk seperti $x(t)$ dengan versi terkompresi. Jika $0 < a < 1$, maka sinyal $y(t)$ merupakan versi ekspansi atau versi pembentangan (*strected*) dari sinyal $x(t)$. Kedua efek operasi ini dikenal sebagai proses *time scaling*, dan bisa dilihat seperti pada Gambar 5.3.



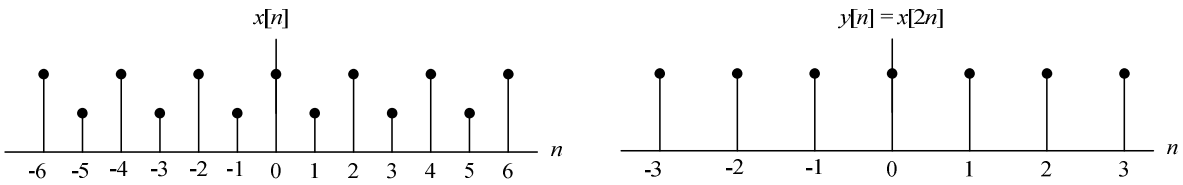
(a) Sinyal asli (b) Time scaling dengan $a > 1$ (c) Time scaling dengan $a < 1$

Gambar 5.3. Operasi time scaling

Di dalam versi sinyal waktu diskrit, operasi *time scaling* bias dinyatkaan dalam persamaan matematik seperti berikut:

$$y[n] = x[kn], \textit{ dimana } k > 0 \tag{5-3}$$

yang dalam hal ini hanya didefinisikan dengan integer pada nilai k. Jika nilai $k > 1$, memungkinkan terjadinya hilangnya komponen nilai pada pada sinyal waktu diskrit $y[kn]$, seperti diilustraikan pada Gambar 5.2, untuk nilai $k = 2$. Sampel-sampel $x[k]$ untuk $n = \pm 1, \pm 3, \ldots$ dst akan hilang karena penempatan $k = 2$ pada $x[kn]$ menyebabkan sampel-sampel ini terlewati. Pada contoh kasus berikut ini dimana $x[n]$ bernilai 1 untuk n = ganjil, dan $x[n]$ bernilai 0 untuk n genap. Maka ketika kita melakukantime scaling dengan $y[n] = x[kn] = x[2n]$, akan menghasilkan nilai 0 untuk semua nilai n. Sebab, $y[n]$ terdiri dari nilai-nilai $x[2]$, $x[4]$, $x[6]$, … dst.



(a) Sinyal waktu diskrit $x[n]$ (a) Versi terkompresi dengan $k = 2$

Gambar 5.3. Time scaling pada sinyal waktu diskrit

Proses time scaling banyak ditemui pada pengolahan sinyal wicara, dimana pada suatu kondisi diperlukan untuk meningkatkan jumlah sampel untuk pembentukan sinyal dari data yang diperoleh

dengan tujuan menghasilkan sinyal yang lebih *smooth*. Proses ini selanjutnya berkembang menjadi teknik yang dikenal dengan *up sampling* dan *interpolasi*. Pada suatu kondisi lainnya, perlu untuk mengurangi jumlah sampel dengan tujuan mempercepat proses komputasi tanpa mengorbankan kualitas sinyal. Proses ini kemudian berkembang menjadi *down sampling* dan *decimation*.

2.3. Reflection

Kita tetapkan $x(t)$ untuk menandai sebuah sinyal waktu kontiyu. Dan selanjutnya $y(t)$ ditetapkan sebagai hasil operasi yang diperoleh melaui penukaran waktu 't' dengan '$-t$', yang merupakan sebuah pembalikan urutan proses sinyal dari belakang ke depan. Sehingga kita memiliki persamaan:

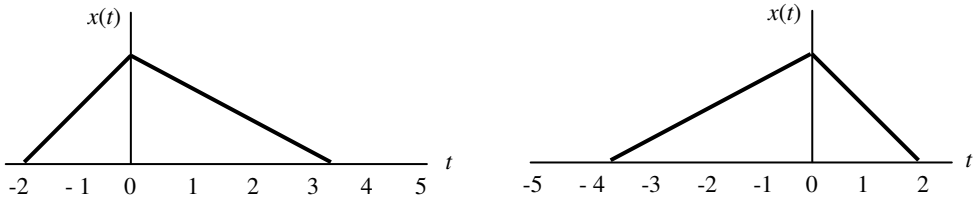
$$y(t) = x(-t) \tag{5-4}$$

Dalam hal ini persamaan diatas merupakan sebuah operasi pemantulan (reflection), yang mengacu pada suatu titik di $t = 0$.

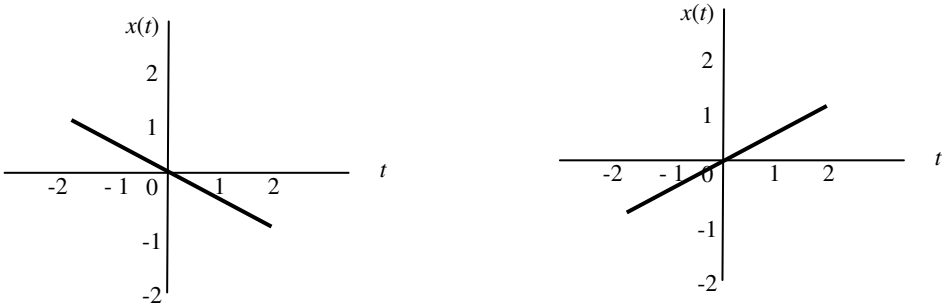
Ada dua kondisi yang menjadi kasus khusus pada operasi refleksi:

- *Sinyal genap*, untuk suau kondisi dimana $x(-t) = x(t)$ belaku untuk semua nilai t. Dalam hal ini sinyal hasil refleksi memiliki nilai yang sama dengan sinyal sebelum proses refleksi.
- *Sinyal ganjil*, untuk suatu kondisi dimana $x(-t) = -x(t)$ berlaku untuk semua nilai t. Dalam hal ini sinyal hasil refleksi merupakan versi negative dari sinyal sebelum proses refleksi.

Dua hal ini juga berlaku untuk sinyal waktu diskrit.



(a) Refleksi pada sinyal dengan fungsi genap



(b) Refleksi pada sinyal dengan fungsi ganjil

Gambar 5.4. Operasi refleksi sinyal

Di dalam aplikasi teknologi telekomunikasi operasi *time reflection* dimanfaatkan untuk proses estimasi dan ekualisasi kanal dengan cara mengembangkan proses refleksi sinyal menjadi sebuah

teknik yang dikenal sebagai *time reversal communication*. Masalah *time reversal* tidak dibahas lebih jauh, karena memerlukan pemahaman berbagai teknik propagasi dan estimasi kanal yang cukup panjang, dan teknologi ini mulai dikembangkan mulai akhir tahun 90-an.

III. Perangkat Yang Diperlukan

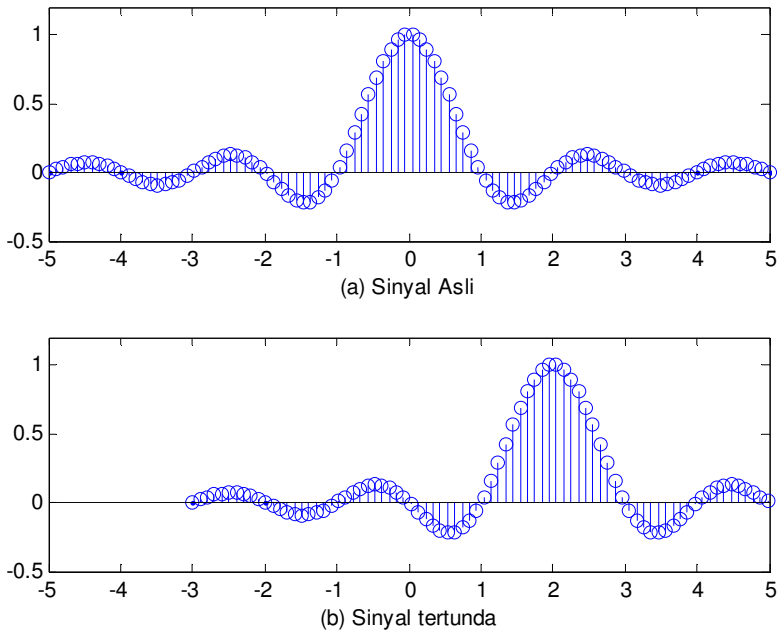
- 1 (satu) buah PC lengkap sound card dan OS Windows dan perangkat lunak Matlab
- 1 (satu) flash disk dengan kapasitas 1 G atau lebih

IV. Langkah Percobaan

4.1. Time Shifting

1. Anda buat sebuah program operasi pergeseran sinyal (time shifting) sederhana seperti pada listing dibawah ini.

```
t = linspace(-5,5);
y = sinc(t);
subplot(211);
stem(t,y,'linewidth',2);
xlabel('(a) Sinyal Asli');
axis([-5 5 -0.5 1.2])
subplot(212);
stem(t-2,y);
xlabel('(b) Sinyal tertunda');
axis([-5 5 -0.5 1.2])
```



Gambar 5.5. hasil operasi pergeseran ke kanan

2. Amati bentuk sinyal yang dihasilkan konsultasikan ke dosen pengampu jika anda belum paham pada tampilan yang dihasilkan program tersebut.
3. Coba anda lakukan sedikit modifikasi dengan cara merubah variable pergeseran dari 2 menjadi 3, 5, 10, dsb. Anda amati perubahan bentuk sinyal yang dihasilkan. Jika gambar yang dihasilkan tidak sesuai dengan yang anda harapkan, coba anda rubah nilai *axis*([*-5 5 -0.5 1.2*]) sesuai dengan menyesuaikan dengan nilai sumbu mendatar (*sb - x*) agar sinyalnya bisa terlihat.
4. Lakukan perubahan nilai variable pergeseran dari positif menjadi negative, dan amati bentuk pergeseran yang dihasilkan.

4.2. Time Scaling (Down Sampling)

1. Anda buat program baru time scaling dengan tujuan memperkecil jumlah sampel pada suatu sekuen, atau yang dikenal dengan down sampling. Anda coba contoh sederhana beikut ini.

```
x = [1 2 3 4 5 6 7 8 9 10];
y = downsample(x,3)
```

anda ketikkan pada matlab command line

```
x,y
x =
   1   2   3   4   5   6   7   8   9   10
y
   =   1   4   7   10
```

Dalam hal ini program anda melakukan pengambilan sampel ke-1, sample ke-4, ke-7,ke-10 untuk disimpan ke variable y.

2. Anda lakukan sedikit modifikasi seperti berikut ini

```
y = downsample(x,3,2)
```

danperhatikan hasilnya apakah seperti berikut ini ?

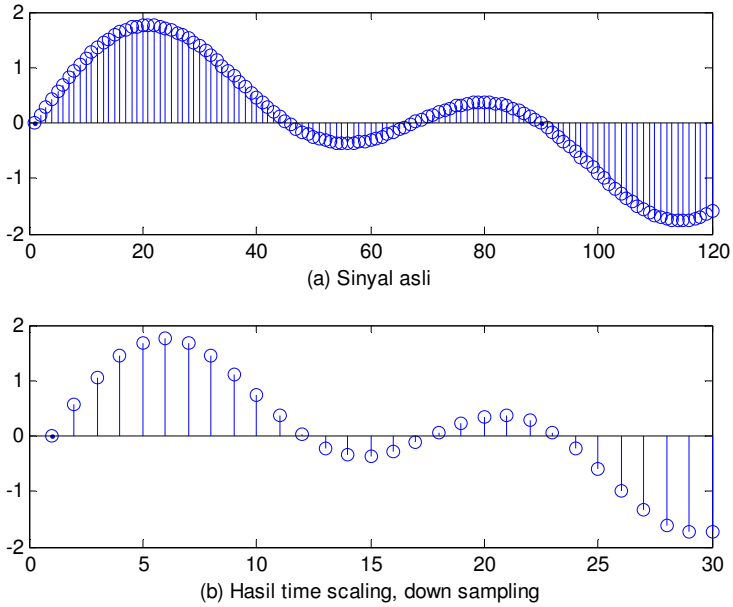
```
y =
    3   6   9
```

Proses down sampling pada program ini dilakukan dengan memberikan phase offset senilai 2, yaitu ada penggeseran sampel yang diambil sebesar 2 sampel ke atas. Sehingga pengambilan sampel dilakukan pada sampel ke-3, ke-6, danke-9.

3. Anda buat sebuah program untuk melakukan time scaling dengan teknik yang berbeda, dalam hal ini tujuannya adalah mendapatkan bentuk sinyal yang lebih halus dengan teknik *Up Sampling*. Anda bisa memanfaatkan kode program berikut ini.

```
t = 0:.00025:1;  % Time vector
x = sin(2*pi*30*t)+ sin(2*pi*60*t);
```

```
k=6;%faktor decimation
y = decimate(x,k);
subplot(211);
gbatas=120;
stem(x(1:gbatas)), axis([0 120 -2 2])   % Original signal
xlabel('(a)batas Sinyal asli')
subplot(212);
stem(y(1:gbatas/k))                     % Decimated signal
xlabel('(b) Hasil time scaling, down sampling')
```



Gambar 5.6. Proses time scaling dengan mengurangi jumlah sampel

4. Amati bentuk sinyal yang dihasilkan, jelaskan perbedaan sinyal bagian atas dan sinyal bagian bawah.
5. Anda rubah nilai decimasi, dengan merubah nilai k = 6, 8, 10, atau 12. Dan anda amati perubahan yang dihasilkan.

4.3. Time Scaling (Up sampling)

1. Anda buat program baru time scaling dengan tujuan memperbanyak jumlah sampel pada suatu sekuen, atau yang dikenal dengan upsamping. Anda coba contoh sederhana beikut ini.

```
x = [1 2 3 4];
y = upsample(x,3);
```

Amati nilai x dan y dengan mengetikkan perintah berikut pada Matlab command line.

```
x,y
```

Apakah ouputnya seperti berikut?

```
x =
   1   2   3   4
y =
   1   0   0   2   0   0   3   0   0   4   0   0
```

Pada program anda menghasilkan sebuah penambahan nilai sampel menjadi 3 kali lebih banyak. Dalam hal ini yang disisipkan adalah nilai 0.

2. Anda lakukan perubahan program anda menjadi sebagai berikut:

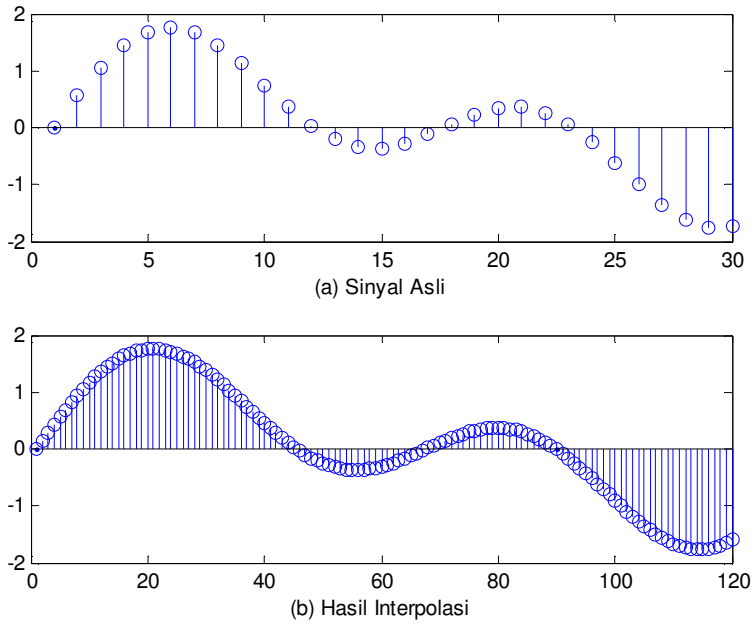
```
x = [1 2 3 4];
y = upsample(x,3,2);
x,y
x =
   1   2   3   4
y =
   0   0   1   0   0   2   0   0   3   0   0   4
```

Dalam hal ini ada semacam pengaturan fase offset senilai 2, yang ditandai dengan proses penyisipan nilai-nilai 0 sebanyak 2 sampel pada sebelum sample pertama.

6. Anda buat program time scaling dengan tujuan mendapatkan penghalusan sinyal menggunakan teknik yang berbeda, dalam hal ini *interpolation, untuk sementara anda bisa* mennganggapnya sebagai *upsamling*. Anda bisa memanfaatkan kode program berikut ini.

```
clc;
t = 0:0.001:1; % Time vector
x = sin(2*pi*30*t) + sin(2*pi*60*t);
k=4;%interpilasi dengan faktor 4
y = interp(x,k);
subplot(211)
gbatas=30;
stem(x(1:gbatas));
xlabel('(a) Sinyal Asli');
subplot(212)
stem(y(1:gbatas*k));
xlabel('(b) Hasil Interpolasi');
```

7. Anda rubah factor interpolasi untuk menghasilkan efek yang berbeda. Dalam hal ini anda ganti nilai k = 6, 8, 10 atau 12. Amati gambar sinyal yang dihasilkan dan berikan penjelasan.

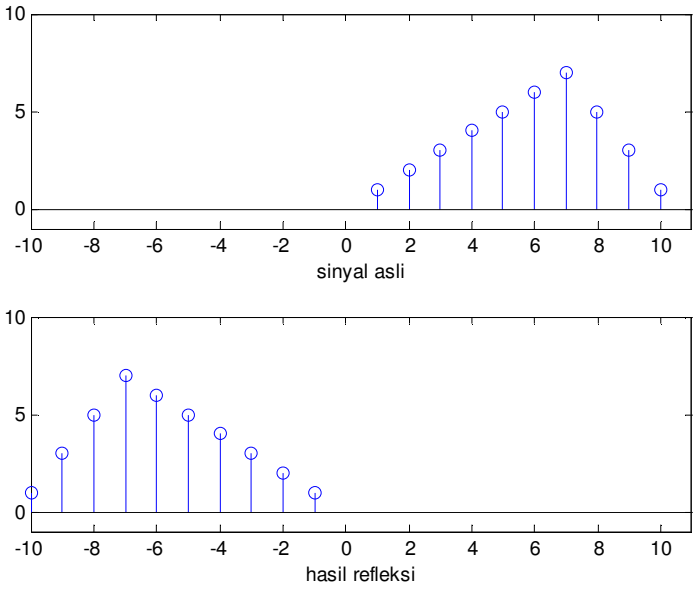


Gambar 5.7. Proses time scaling dengan memperbanyak jumlah sampel

4.4. Time Reflection

1. Anda buat program time reflection dengan memanfaatkan kode program berikut ini.

```
%ss_coba_r.m
clc;
clear all;
close all;
x=[1 2 3 4 5 6 7 5 3 1];
x_max=length(x);
for i=1:x_max,
   k=x_max-i+1;
   y(i)=x(1,k);
end
n=1:x_max
subplot(211)
stem(n,x);axis([-10 11 -1 10])
xlabel('sinyal asli')
subplot(212)
stem(n-(x_max+1),y);axis([-10 11 -1 10])
xlabel('hasil refleksi')
```



Gambar 5.8. Proses refleksi pada sinyal

2. Anda rubah program diatas untuk menghasilkan sebuah proses refleksi sinyal dengan pusat pencerminan bukan pada sumbu y, atau pada x tidak sama dengan nol. Dalam hal ini anda bisa melakukannya pada posisi x =2, atau x=4. Dan amati bentuk sinyal yang dihasilkan.

MODUL 6
PROSES SAMPLING

I. Tujuan Instruksional Khusus:

- Siswa memahami pengaruh pemilihan jumlah sample dan pengaruhnya pada proses recovery sinyal

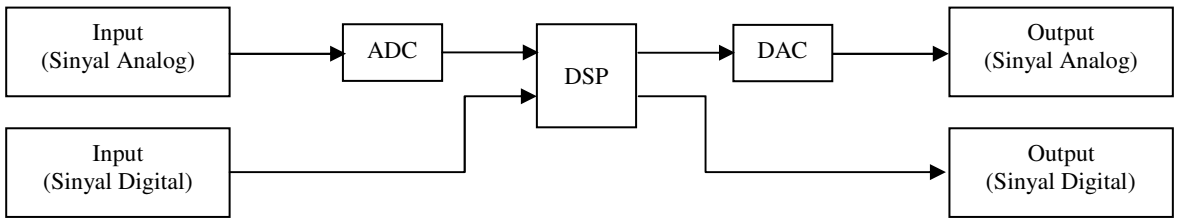
II. Teori Sampling

2.1. Analog to Digital Conversion

Dalam proses pengolahan sinyal analog, sinyal input masuk ke Analog Signal Processing (ASP), diberi berbagai perlakukan (misalnya pemfilteran, penguatan, dsb.) dan outputnya berupa sinyal analog.

Gambar 6.1. Sistem Pengolahan Sinyal Analog

Proses pengolahan sinyal secara digital memiliki bentuk sedikit berbeda. Komponen utama system ini berupa sebuah processor digital yang mampu bekerja apabila inputnya berupa sinyal digital. Untuk sebuah input berupa sinyal analog perlu proses awal yang bernama digitalisasi melalui perangkat yang bernama *analog-to-digital conversion* (ADC), dimana sinyal analog harus melalui proses sampling, quantizing dan coding. Demikian juga output dari processor digital harus melalui perangkat *digital-to-analog conversion* (DAC) agar outputnya kembali menjadi bentuk analog. Ini bisa kita amati pada perangkat seperti PC, digital sound system, dsb. Secara sederhana bentuk diagram bloknya adalah seperti Gambar 6.2.



Gambar 6.2. Sistem Pengolahan Sinyal Digital

2.2. Proses Sampling

Berdasarkan pada penjelasan diatas kita tahu betapa pentingnya satu proses yang bernama sampling. Setelah sinyal waktu kontinyu atau yang juga popoler kita kenal sebagai sinyal analog

disampel, akan didapatkan bentuk sinyal waktu diskrit. Untuk mendapatkan sinyal waktu diskrit yang mampu mewakili sifat sinyal aslinya, proses sampling harus memenuhi syarat Nyquist.

$$fs > 2fi \qquad (6\text{-}1)$$

dimana:

fs = frekuensi sinyal sampling

fi = frekuensi sinyal informasi yanga kan disampel

Fenomena aliasing proses sampling akan muncul pada sinyal hasil sampling apabila proses frekuensi sinyal sampling tidak memenuhi criteria diatas. Perhatikan sebuah sinyal sinusoida waktu diskrit yang memiliki bentuk persamaan matematika seperti berikut:

$$x(n) = A \sin(\omega n + \theta) \qquad (6\text{-}2)$$

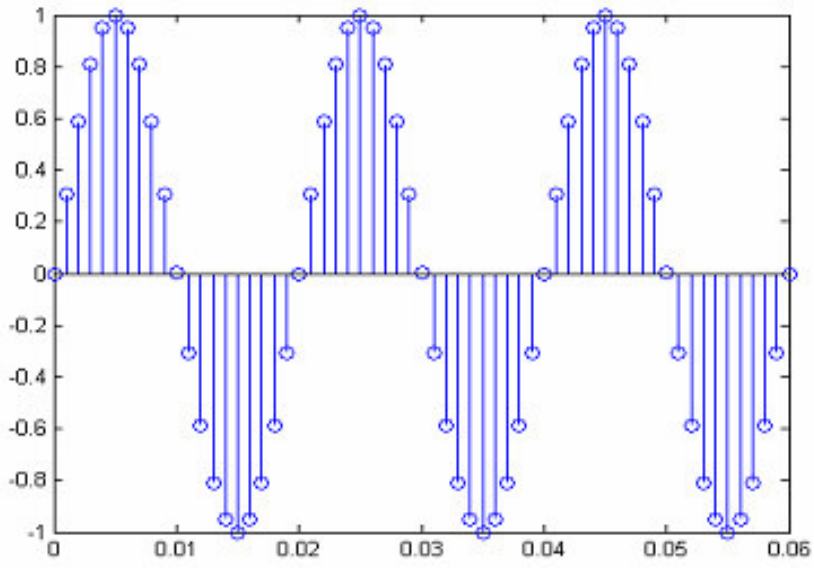
dimana:

A = amplitudo sinyal

ω = frekuensi sudut

θ = fase awal sinyal

Frekuensi dalam sinyal waktu diskrit memiliki satuan radian per indek sample, dan memiliki ekuivalensi dengan $2\pi f$.



Gambar 6.3. Sinyal sinus diskrit

Sinyal sinus pada Gambar 3 tersusun dari 61 sampel, sinyal ini memiliki frekuensi f = 50 dan disampel dengan Fs = 1000. Sehingga untuk satu siklus sinyal sinus memiliki sample sebanyak Fs/f = 1000/50 = 20 sampel. Berbeda dengan sinyal waktu kontinyu (C-T), sifat frekuensi pada sinyal waktu diskrit (D-T) adalah:

1. Sinyal hanya periodik jika f rasional. Sinyal periodic dengan periode N apabila berlaku untuk untuk semua n bahwa x(n+N) = x(n). Periode fundamental *NF* adalah nilai N yang terkecil.
   Sebagai contoh:
   agar suatu sinyal periodic maka $\cos(2\pi(N+n) + \theta) = \cos(2\pi n + \theta) = \cos(2\pi n + \theta + 2\pi k)$

$$\Leftrightarrow 2\pi N = 2\pi k \Leftrightarrow f = \frac{k}{N} \Leftrightarrow f \qquad harus\, rasional$$

2. Sinyal dengan fekuensi beda sejauh k2π (dengan k bernilai integer) adalah identik. Jadi berbeda dengan kasus pada C-T, pada kasus D-T ini sinyal yang memiliki suatu frekuensi unik tidak berarti sinyal nya bersifat unik.

   Sebagai contoh:

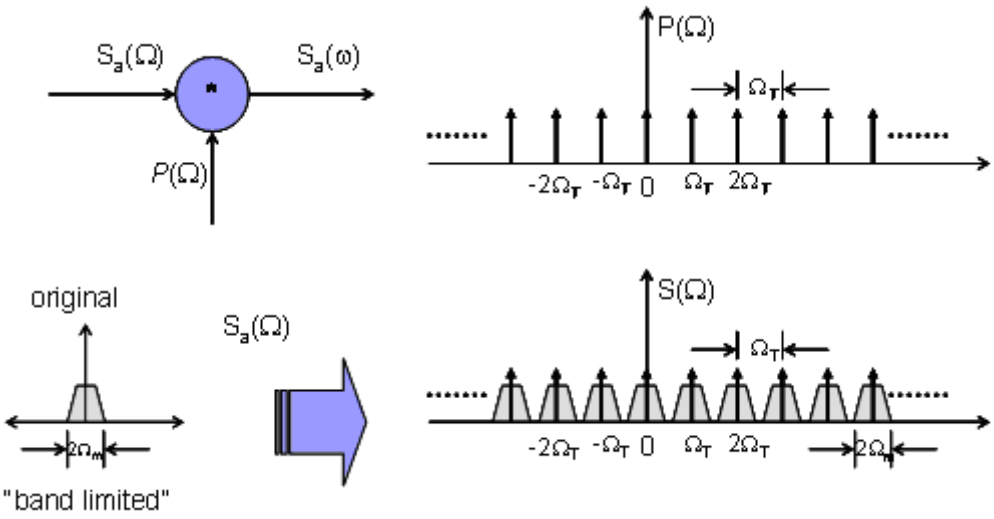
   $\cos[(\omega o + 2\pi)n + \theta] = \cos(\omega o + 2\pi)$

   karena $\cos(\omega o + 2\pi) = \cos(\omega o)$. Jadi bila $x_k(n) = \cos(\omega_o n + 2\pi)$, k = 0,1,…. Dimana $\omega k = \omega_o n + 2k\pi$, maka xk(n) tidak bisa dibedakan satu sama lain.

   Artinya x1(n) = x2(n) = x3(n)….= xk(n). Sehingga suatu sinyal dengan frekuensi berbeda akan berbeda jika frekuensinya dibatasi pada daerah $-\pi < \omega < \pi$ atau $-1/2 < f < 1/2$.

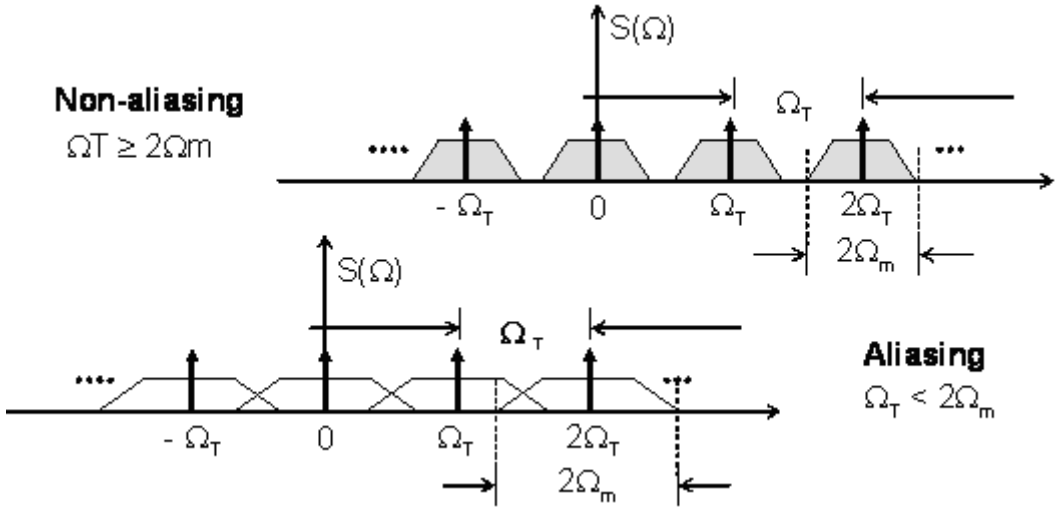
2.3. Proses Aliasing

Seperti telah dijelaskan diatas bahwa proses aliasing akan terjadi jika frekuensi sampling tidak sesuai dengan aturan Nyquist. Gambar 6.4 memperlihatkan proses sampling jika dilihat dari kawasan frekuensi. Karena transformasi Fourier dari deretan impuls adalah juga suatu deretan impuls, maka konvolusi antara spektrum sinyal S(Ω) dengan impuls $\delta(\Omega - k\Omega_T)$ menghasilkan pergeseran spektrum sejauh $k\Omega_T$. Sebagai akibatnya akan terjadi pengulangan (tiling) spektrum di seluruh rentang frekuensi pada posisi kelipatan dari frekuensi pencuplikan. Gambar 6.4 bagian kiri bawah menunjukkan spektrum dari sinyal yang lebar pitanya Ω_m yang kemudian mengalami proses pengulangan akibat proses sampling.



Gambar 6.4. Pencuplikan dilihat dari kawasan frekuensi

Jika jarak antar pengulangan atau grid pengulangan cukup lebar, seperti diperlihatkan pada Gambar 6.5 bagian atas, yang juga berarti bahwa frekuensi samplingnya cukup besar, maka tidak akan terjadi tumpang tindih antar spektrum yang bertetangga. Kondisi ini disebut sebagai *non-aliasing*. Selanjutnya sifat keunikan dari transformasi Fourier akan menjamin bahwa sinyal asal dapat diperoleh secara sempurna. Sebaliknya, jika Ω_T kurang besar, maka akan terjadi tumpang tindih antar spektrum yang mengakibatkan hilangnya sebagian dari informasi. Peristiwa ini disebut *aliasing*, seperti diperlihatkan pada Gambar 6.5 bagian bawah.



Gambar 6.5. Kondisi non-aliasing dan aliasing pada proses pencuplikan

Pada kondisi ini, sinyal tidak dapat lagi direkonstruksi secara eksak. Dengan memahami peristiwa aliasing dalam kawasan frekuensi, maka batas minimum laju pencuplikan atau batas Nyquist

dapat diperoleh, yaitu sebesar $\Omega_{Nyquist} = \Omega_m$. Hasil ini dirumuskan sebagai teorema Shannon untuk pencuplikan sebagai berikut:

> Teorema Pencuplikan Shannon. Suatu sinyal pita-terbatas dengan lebar Ω_m dapat direkonstruksi secara eksak dari cuplikannya jika laju pencuplikan minimum dua kali dari lebar pita tersebut, atau $\Omega_T \geq 2\Omega_m$

Sebagai contoh, manusia dapat mendengar suara dari frekuensi 20 Hz sampai dengan sekitar 20kHz, artinya lebar pita dari suara yang mampu didengar manusia adalah sekitar 20 kHz. Dengan demikian, pengubahan suara menjadi data dijital memerlukan laju pencuplikan sedikitnya 2×20kHz = 40 kHz atau 40.000 cuplikan/detik supaya sinyal suara dapat direkonstruksi secara sempurna, yang berarti juga kualitas dari suara hasil perekaman dijital dapat dimainkan tanpa distorsi.

III. Perangkat Yang Diperlukan

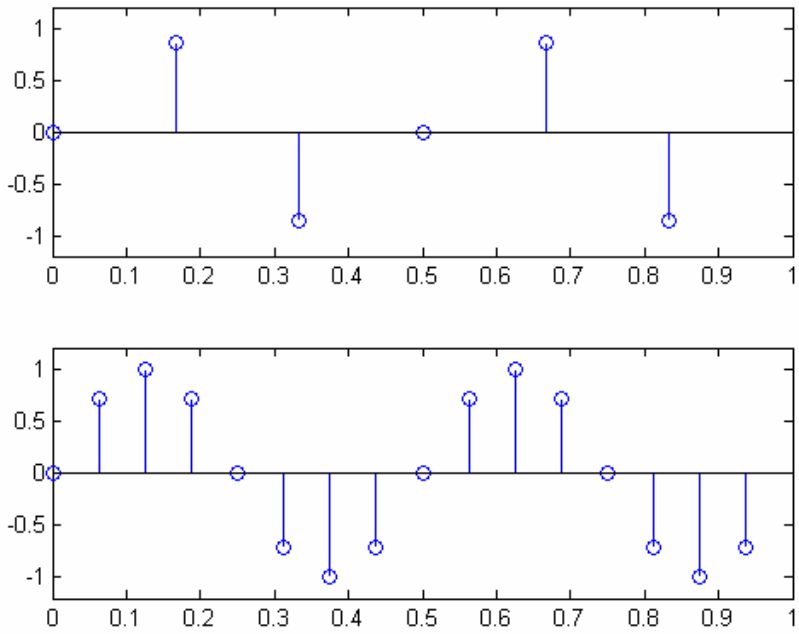
- PC yang dilengkapi dengan perangkat multimedia (sound card, Microphone, Speaker active, atau headset)
- Sistem Operasi Windows dan Perangkat Lunak Matlab yang dilengkapi dengan tool box DSP.

IV. Langkah Percobaan

4.1 Pengamatan Pengaruh Pemilihan Frekuensi Sampling Secara Visual

Prosedur yang akan anda lakukan mirip dengan yang ada di percobaan 2, tetapi disini lebih ditekankan pad akonsep pemahaman fenomena sampling. Untuk itu anda mulai dengan membuat program baru dengan perintah seperti berikut :

```
%sampling_1.m
Fs=8;%frekuensi sampling
t=(0:Fs-1)/Fs;%proses normalisasi
s1=sin(2*pi*t*2);
subplot(211)
stem(t,s1)
axis([0 1 -1.2 1.2])
Fs=16;%frekuensi sampling
t=(0:Fs-1)/Fs;%proses normalisasi
s2=sin(2*pi*t*2);
subplot(212)
stem(t,s2)
axis([0 1 -1.2 1.2])
```



Gambar 6.6. Pengaruh jumlah sample berbeda terhadap satu periode sinyal terbangkit

Lakukan perubahan pada nilai Fs, pada sinyal s1 sehingga bernilai 10, 12, 14, 16, 20, dan 30. Catat apa yang terjadi ? Apa pengaruh fs terhadap jumlah sample ? Apa pengaruh jumlah sample berbeda untuk satu periode sinyal terbangkit?

4.2 Pengamatan Pengaruh Pemilihan Frekuensi Sampling pada Efek Audio

Disini kita akan mendengarkan bagaimana pengaruh frekuensi sampling melalui sinyal audio. Untuk itu anda harus mempersiapkan PC anda dengan speaker aktif yang sudah terkonek dengan sound card. Selanjutnya anda ikuti langkah berikut :

1. Buat program bari sampling_2.m dengan perintah seperti berikut ini.

```
%sampling_2.m
clear all;
Fs=1000;
t=0:1/Fs:0.25;
f=100;
x=sin(2*pi*f*t);
%sound(x,Fs)
plot(x)
```

2. Setelah anda menjalankan program tersebut, apa yang anda dapatkan? Selanjutnya coba anda rubah nilai f = 200, 250,300, 350, 400 dan 850. Plot hasil running program dari masing-masing nilai f (dengan subplot). Apa yang anda dapatkan? Beri penjelasan tentang kejadian tersebut.

4.3 Pengamatan Efek Aliasing pada Audio

Tentunya anda bosan dengan sesuatu yang selalu serius, marilah kita sedikit bernafas melepaskan ketegangan tanpa harus meninggalkan laboratorium tempak praktikum. Caranya?

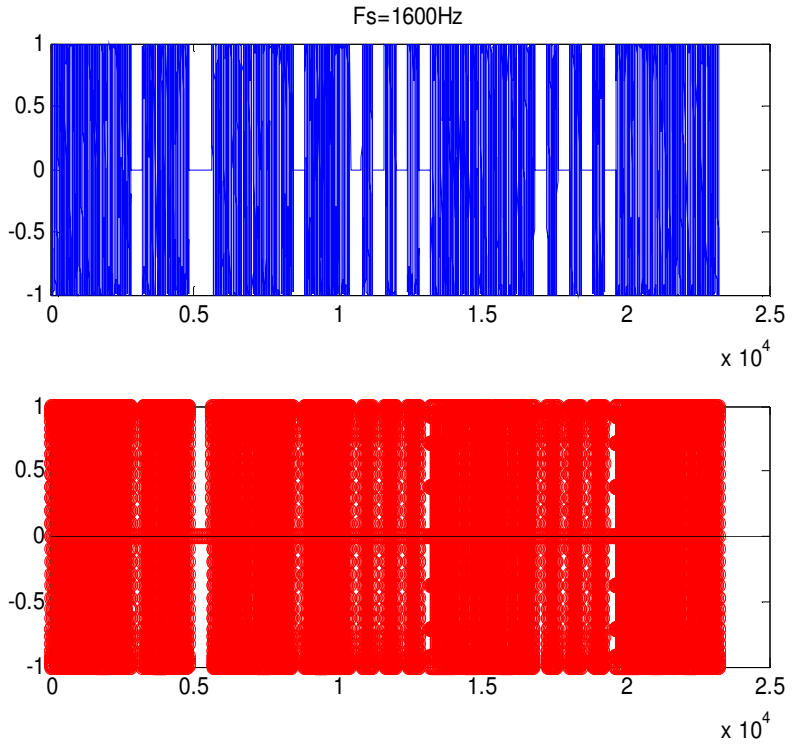
1. Anda susun sebuah lagu sederhana dengan cara membuat program baru berikut ini.

```
clc ; clf ;
Fs=16000;
t=0:1/Fs:0.25;
c=sin(2*pi*262*t);
d=sin(2*pi*294*t);
e=sin(2*pi*330*t);
f=sin(2*pi*249*t);
g=sin(2*pi*392*t);
a=sin(2*pi*440*t);
b=sin(2*pi*494*t);
c1=sin(2*pi*523*t);
nol = [zeros(size(t))];
nada1 = [c,e,c,e,f,g,g,nol,b,c1,b,c1,b,g,nol,nol];
nada2 = [c,e,c,e,f,g,g,nol,b,c1,b,c1,b,g,nol];
nada3 = [c,nol,e,nol,g,nol,f,f,g,f,e,c,f,e,c,nol];
nada4 = [c,nol,e,nol,g,nol,f,f,g,f,e,c,f,e,c];
lagu=[nada1,nada2,nada3,nada4];
sound(lagu,Fs)
subplot(211)
plot (lagu)
subplot(212)
stem (lagu)
```

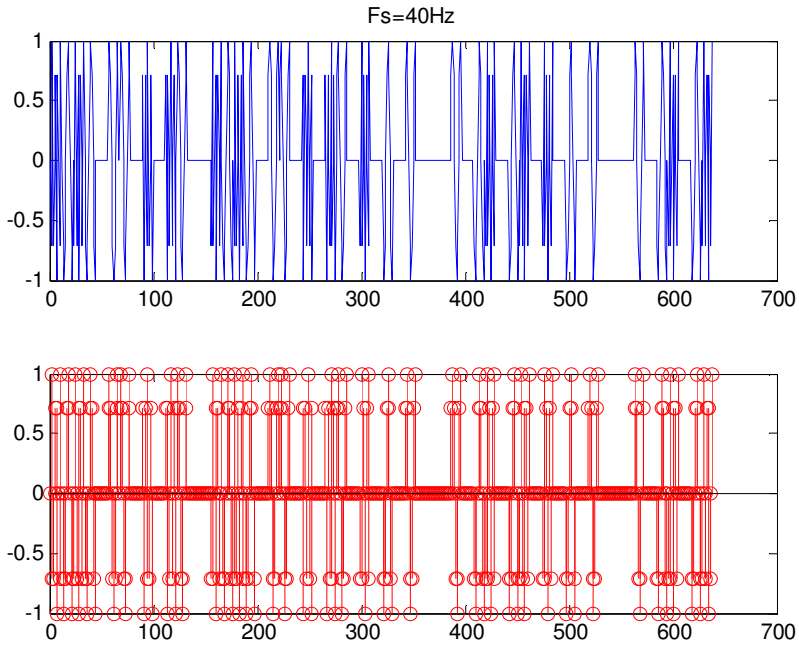
2. Pada bagian akhir program anda tambahkan perintah berikut

```
wavwrite(lagu,‘gundul.wav’)
```

3. Coba anda minimize Matlab anda, cobalah gunakan Windows Explorer untuk melihat dimana file gundul.wav berada. Kalau sudah terlihat coba click kanan pada gundul.wav dan bunyikan.
4. Coba anda edit program anda diatas, dan anda lakukan perubahan pada nilai frekuensi sampling Fs=14000, menjadi Fs =10000, 2000 dan 800. Plot perubahan frekuensi tersebut. Apa yang anda dapatkan?



Gambar 6.7. Penggambaran sinyal lagu



Gambar 6.8. Pengaruh pemilihan Fs pada sinyal lagu

V. Analisa Data

Catat semua peristiwa yang terjadi dari hasil percobaan anda, buat laporan dan analisa mengapa muncul fenomena seperti diatas? Fenomena itu lebih dikenal dengan nama apa? Apa yang menyebabkannya?

MODUL 7
OPERASI KONVOLUSI SINYAL DISKRIT

I. Tujuan Instruksional Khusus:

- Siswa dapat memahami proses operasi konvolusi pada dua sinyal diskrit dan pengaruhnya terhadap hasil konvolusi.

II. Teori Konvolusi

2.1 Konvolusi dua Sinyal

Konvolusi antara dua sinyal diskrit $x[n]$ dan $v[n]$ dapat dinyatakan sebagai

$$x[n] * v[n] = \sum_{i=-\infty}^{\infty} x[i]v[n-i] \qquad (7\text{-}1)$$

Bentuk penjumlahan yang ada di bagian kanan pada persamaan (7-1) disebut sebagai *convolution sum.* Jika $x[n]$ dan $v[n]$ memiliki nilai 0 untuk semua integer pada $n < 0$, selanjutnya $x[i] = 0$ untuk semua integer pada $i < 0$ dan $v[i-n] = 0$ untuk semua integer n – i < 0 (atau $n < i$). Sehingga jumlahan pada persamaan (7-1) akan menempati dari nilai $i = 0$ sampai dengan $i = n$, dan operasi konvolusi selanjutnya dapat dituliskan sebagai:

$$x[n] * v[n] = \begin{cases} 0 & , n = -1,-2,... \\ \sum_{i=-\infty}^{\infty} x[i]v[n-i] & , n = 0,1,2,... \end{cases} \qquad (7\text{-}2)$$

2.2. Mekanisme Konvolusi

Komputasi pada persamaan (1) dan (2) dapat diselesaikan dengan merubah *discretetime index* n sampai dengan i dalam sinyal x[n] dan v[n]. Sinyal yang dihasilkan x[i] dan v[i] selanjutnya menjadi sebuah fungsi *discrete-time index* i. Step berikutnya adalah menentukan v[n-i] dan kemudian membentuk pencerminan terhadap sinyal v[i]. Lebih tepatnya v[-i] merupakan pencerminan dari v[i] yang diorientasikan pada sumbu vertikal (axis), dan v[n-i] merupakan v[-i] yang digeser ke kanan deng an step n. Saat pertama kali *product* (hasil kali) x[i]v[n-i] terbentuk, nilai pada konvolusi x[n]*v[n] pada titik n dihitung dengan menjumlahkan nilai x[i]v[n-i] sesuai rentang i pada sederetan nilai integer tertentu.

Untuk lebih jelasnya permasalahan ini akan disajikan dengan suatu contoh penghitung konvolusi pada dua deret nilai integer berikut ini.

Sinyal pertama: x[i]= 1 2 3

Sinyal kedua : v[i]= 2 1 3

• Step pertama adalah pembalikan sinyal kedua, v[n] sehingga didapatan kondisi seperti berikut:

Sinyal pertama : x[i] = 1 2 3

Sinyal kedua : v[-i] = 3 1 2

• Step ke dua adalah pergeseran dan penjumlahan

Sinyal pertama : 1 2 3

Sinyal kedua : 3 1 2

---------- x

product and sum : 0 0 2 0 0 = 2

• Step ke tiga adalah pergeseran satu step dan penjumlahan

Sinyal pertama : 1 2 3

Sinyal kedua : 3 1 2

---------- x

product and sum : 0 1 4 0 = 5

• Step ke empat adalah pergeseran satu step dan penjumlahan

Sinyal pertama : 1 2 3

Sinyal kedua : 3 1 2

---------- x

product and sum : 3 2 6 = 11

• Step ke lima adalah pergeseran satu step dan penjumlahan

Sinyal pertama : 1 2 3

Sinyal kedua : 3 1 2

---------- x

product and sum : 0 6 3 0 = 9

• Step ke enam adalah pergeseran satu step dan penjumlahan

Sinyal pertama : 1 2 3

Sinyal kedua : 3 1 2

---------- x

product and sum : 0 0 9 0 0 = 9

• Step ke tujuh adalah pergeseran satu step dan penjumlahan

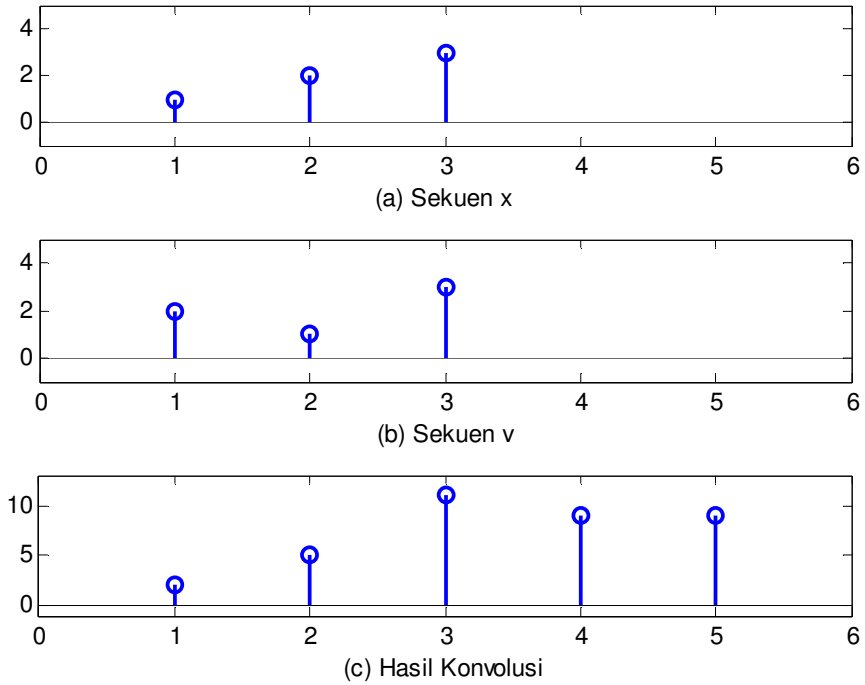
Sinyal pertama : 1 2 3

Sinyal kedua : 3 1 2

---------- x

product and sum : 0 0 0 0 0 0 = 0

Dari hasil product and sum tersebut hasilnya dapat kita lihat dalam bentuk deret sebagai berikut: 2 5 11 9 9 Disini hasil penghitungan product and sum sebelum step pertama dan step ke tujuh dan selanjutnya menunjukkan nilai 0, sehingga tidak ditampilkan. Secara grafis dapat dilihat seperti berikut ini:



Gambar 7.1. Mekanisme konvolusi

Pada gambar 1 bagian atas, menunjukkan sinyal $x[n]$, bagian kedua menunjukkan sinyal $v[n]$, sedangkan bagian ketiga atau yang paling bawah merupakan hasil konvolusi.

III. Perangkat Yang Diperlukan

- PC yang dilengkapi dengan perangkat multimedia (sound card, Microphone, Speaker active, atau headset)
- Sistem Operasi Windows dan Perangkat Lunak Matlab yang dilengkapi dengan tool box DSP

IV. Langkah Percobaan

4.1. Konvolusi Dua Sinyal Unit Step

Disini kita akan membangkitkan sebuah sinyal unit step diskrit yang memiliki nilai seperti berikut:

$$x[n] = v[n] = \begin{cases} 1 & untuk\ 0 \le n \le 4 \\ 0 & untuk\ yang\ lain \end{cases}$$

Dan melakukan operasi konvolusi yang secara matematis dapat dituliskan sebagai berikut:

$$x[n]*v[n]$$

Untuk itu langkah yang harus dilakukan adalah sebagai berikut:

1. Bangkitkan sinyal x[n] dengan mengetikkan perintah berikut:

```
L=input('Panjang gelombang(>=10) : ');
P=input('Lebar pulsa (lebih kecil dari L): ');
for n=1:L
    if n<=P
        x(n)=1;
    else
        x(n)=0;
    end
end
 t=1:L;
subplot(3,1,1)
stem(t,x)
```

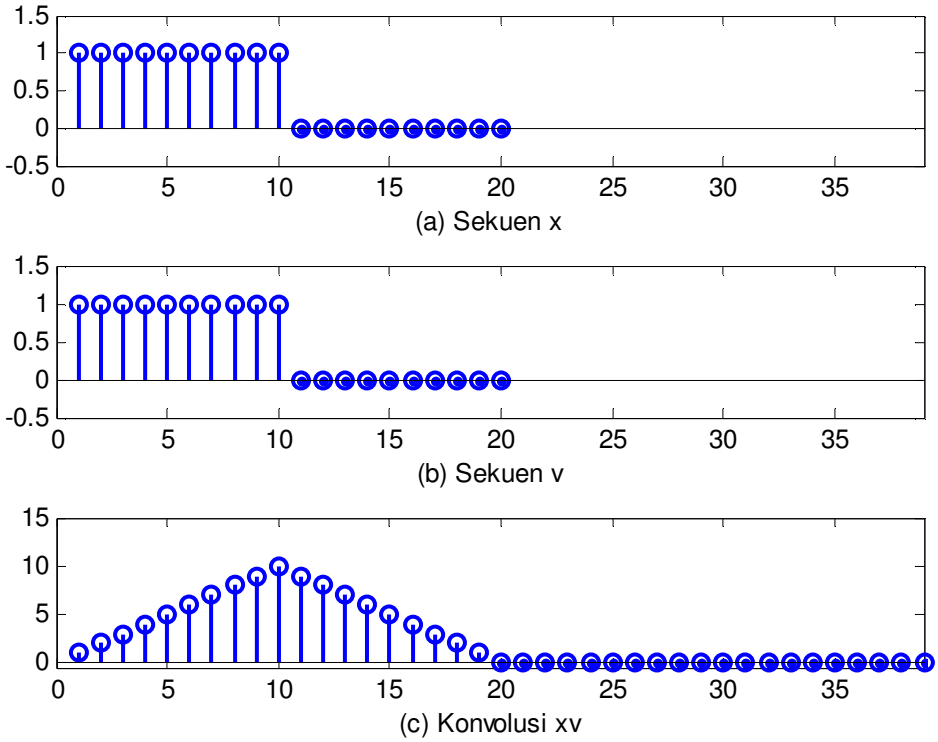
2. Jalankan program dan tetapkan nilai L=20 dan P=10.
3. Selanjutnya masukkan pembangkitan sekuen unit step ke dua dengan cara menambahkan syntax berikut ini di bawah program anda pada langkah pertama:

```
for n=1:L
    if n<=P
        v(n)=1;
    else
        v(n)=0;
    end
end
t=1:L;
subplot(3,1,2)
stem(t,v)
```

4. Coba jalankan program dan tambahkan perintah berikut:

```
subplot(3,1,3)
stem(conv(x,v))
```

5. Coba anda jalankan seperti pada langkah kedua, dan apakah hasilnya seperti ini Gambar 7.2 ?
6. Ulangi langkah ke 5 dan rubahlah nilai untuk *L*=12, 15, dan 12. Sedangkan untuk *P* masukkan nilai 10, 5, dan 12, apa yang terjadi?



Gambar 7.2. Contoh hasil konvolusi 2 sinyal unit step

4.2. Konvolusi Dua Sinyal Sekuen konstan

Disini kita mencoba untuk membangkitkan dua sinyal sekuen konstan dan melakukan operasi konvolusi untuk keduanya. Langkah yang harus anda lakukan adalah sebagai berikut:

1. Bangkitkan sinyal *x*[*n*] dengan mengetikkan perintah berikut:

```
%Pembangkitan Sekuen Konstan Pertama
L1=21;
for n=1:L1;
if (n>=2);
   st1(n)=1;
else
   st1(n)=0;
end
```

```
end
t1=[0:1:(L1-1)];
subplot(311)
stem(t1,st1)
title('Konvolusi 2 sinyal sekuen konstan')
```

2. Jalankan program dan tetapkan nilai *L*2 = 11
3. Selanjutnya masukkan pembangkitan sekuen konstan ke dua dengan cara menambahkan syntax berikut ini di bawah program anda pada langkah pertama:

```
%Pembangkitan Sekuen Konstan Kedua
L2=21;
for n=1:L2;
if (n>=2);
   st2(n)=1;
else
   st2(n)=0;
end
end
t2=[0:1:(L2-1)];
subplot(312)
stem(t2,st2)
xlabel('Jumlah Sample')
```

4. Coba jalankan program dan tambahkan perintah berikut:

```
subplot(313)
c=conv(st1,st2)
stem(c)
```

5. Coba anda jalankan seperti pada langkah kedua, dan berikan penjelasan hasilnya.
6. Ulangi langkah ke 5 dan rubahlah nilai untuk *L*1=15, 10 dan amplitudo 2. Sedangkan untuk *L*2 = 18, 8 dan amplitudo 3, apa yang terjadi?

4.3. Konvolusi Dua Sinyal Sinus Diskrit

Disini kita mencoba untuk membangkitkan dua sinyal sinus diskrit dan melakukan operasi konvolusi untuk keduanya. Langkah yang harus anda lakukan adalah sebagai berikut:

1. Buat program untuk membangkitkan dua gelombang sinus seperti berikut:

```
L=input('Banyaknya titik sampel(>=20): ');
f1=input('Besarnya frekuensi gel 1 adalah Hz: ');
f2=input('Besarnya frekuensi gel 2 adalah Hz: ');
teta1=input('Besarnya fase gel 1(dalam radiant): ');
```

```
teta2=input('Besarnya fase gel 2(dalam radiant): ');
A1=input('Besarnya amplitudo gel 1: ');
A2=input('Besarnya amplitudo gel 2: ');
%Sinus pertama
t=1:L;
t=2*t/L;
y1=A1*sin(2*pi*f1*t + teta1*pi);
subplot(3,1,1)
stem(y1)
%Sinus kedua
t=1:L;
t=2*t/L;
y2=A2*sin(2*pi*f2*t + teta2*pi);
subplot(3,1,2)
stem(y2)
```

2. Coba anda jalankan program anda dan isikan seperti berikut ini:

   Banyaknya titik sampel(>=20): 20
   Besarnya frekuensi gel 1 adalah Hz: 1
   Besarnya frekuensi gel 2 adalah Hz: 2
   Besarnya fase gel 1(dalam radiant): 0
   Besarnya fase gel 2(dalam radiant): 0.25
   Besarnya amplitudo gel 1: 1
   Besarnya amplitudo gel 2: 1

   Perhatikan tampilan yang dihasilkan. Apakah ada kesalahan pada program anda?

3. Lanjutkan dengan menambahkan program berikut ini pada bagian bawah program yang anda buat tadi.

```
subplot(3,1,3)
stem(conv(y1,y2))
```

4. Jalankan program anda, dan kembali lakukan pengisian seperti pada langkah ke 3. Lihat hasilnya apakah anda melihat tampilan seperti berikut gambar 4.
5. Ulangi langkah ke 4, dengan menetapkan nilai sebagai berikut: L=50. $w1 = w2 = 2$, $teta1$=1.5, $teta2 = 0.5$, dan $A1 = A2 = 1$. Apa yang anda dapatkan? Apakah anda mendapatkan hasil yang berbeda dari program sebelumnya? Mengapa ?

V. Data dan Analisa

Anda telah melakukan berbagai langkah untuk percobaan operasi konvolusi. Yang harus anda lakukan adalah menjawab setiap pertanyaan yang ada pada langkah percobaan. Tulis semua komentar dan analisa sebagai penjelasan dari hasil percobaan anda.

1. Apa arti konvolusi 2 buah sinyal diskrit dalam simulasi tersebut diatas ? Bagaimana jika amplitudo masing-masing sinyal diubah ? Jelaskan !!!
2. Jelaskan pengaruh konvolusi terhadap sinyal asli ?

MODUL 8
OPERASI KONVOLUSI SINYAL WAKTU KONTINYU

I. Tujuan Instruksional Khusus:

- Siswa dapat memahami proses operasi konvolusi pada dua sinyal kontinyu dan pengaruhnya terhadap hasil konvolusi.

II. Teori Konvolusi Sinyal Waktu Kontinyu

2.1 Konvolusi Sinyal Kontinyu

Representasi sinyal dalam impuls artinya adalah menyatakan sinyal sebagai fungsi dari impuls, atau menyatakan sinyal sebagai kumpulan dari impuls-impuls. Sembarang sinyal diskret dapat dinyatakan sebagai penjumlahan dari impuls-impuls diskret dan sembarang sinyal kontinyu dapat dinyatakan sebagai integral impuls.

Secara umum, sebuah sinyal diskret sembarang $x[n]$ dapat dinyatakan sebagai penjumlahan impuls-impuls:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{8-1}$$

Seperti pada sistem diskret, sebuah sinyal kontinyu sembarang dapat dinyatakan sebagai integral dari impuls-impuls:

$$x(t) = \int_{-\infty}^{\infty} x(\tau)\delta(t-\tau)d\tau \tag{8-2}$$

Keluaran sebuah sistem disebut juga respon. Jika sinyal berupa unit impulse masuk ke dalam sistem, maka sistem akan memberi respon yang disebut respon impuls (impulse response). Respon impuls biasa diberi simbol h. Jika sistemnya diskret, respon impulsnya diberi simbol $h[n]$ dan jika sistemnya kontinyu, respon impulsnya diberi simbol $h(t)$.

Jika $h[n]$ adalah respon impuls sistem linier diskrit, dan $x[n]$ adalah sinyal masukan maka sinyal keluaran adalah

$$\begin{aligned} y[n] &= x[n] * h[n] \\ &= \sum_{k=-\infty}^{\infty} x[k]h[n-k] \end{aligned} \tag{8-3}$$

Rumusan di atas disebut penjumlahan konvolusi. Jika $h(t)$ adalah respon impuls sistem linier kontinyu, dan x(t) adalah sinyal masukan maka sinyal keluaran adalah

$$\begin{aligned} y(t) &= x(t) * h(t) \\ &= \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \end{aligned} \tag{8-4}$$

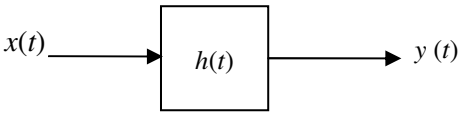
Rumusan di atas disebut integral konvolusi.

Operasi konvolusi mempunyai beberapa sifat operasional:

1. Komutatif : $x(t)*h(t) = h(t)* x(t)$
2. Asosiatif : $x(t)*(y(t)*z(t)) = (x(t)*y(t))*z(t)$
3. Distributif : $x(t)*(y(t) + z(t)) = (x(t)*y(t)) + (x(t)*z(t))$

2.2. Mekanisme Konvolusi Sinyal Kontinyu

Misalnya kita memiliki sebuah sistem linear time invariant (LTI) dengan tanggapan impuls $h(t)$ yang bisa disederhanakan seperti pada diagarm blok berikut ini.



Gambar 8.1. Sistem linear time invariant

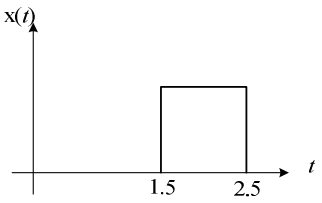
Jika sinyal input dalam hal ini adalah $x(t)$, maka sinyal outputnya bisa dinyatakan dalam persamaan berikut ini.

$$y(t) = x(t) * h(t) = h(t) * x(t)$$

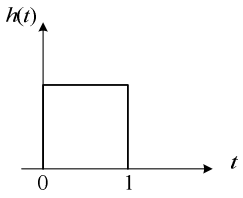
$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau \qquad (8\text{-}5)$$

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau$$

Untuk lebih mudahnya di dalam pemahaman, bisa dilakukan dengan pendekatan grafik. Misalnya kita memiliki $x(t)$ dan $h(t)$ dengan bentuk seperti pada Gambar 8.2 dibawah.



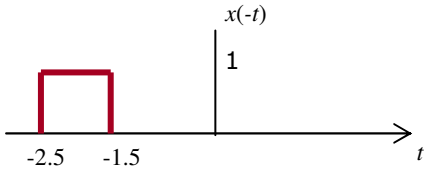
(a) Input sistem LTI



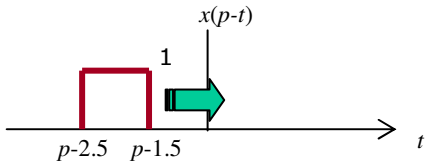
(b) Respon Impulse sistem LTI

Gambar 8.2. Sinyal input dan respon impulse

Kita mulai dengan melakukan refleksi sinyal input x(t) terhadap sumbu y(x=0), sehingga diperoleh hasil secara grafik seperti pada Gambar 8.3a.



Gambar 8.3a. Proses refleksi sinyal input, atau proses flip x(t)



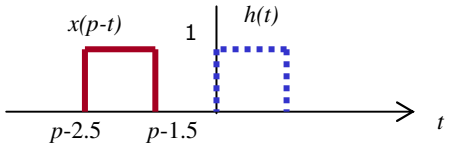
Gambar 8.3b. Proses pergeseran sinyal input, atau $x(p - t)$

Dilanjutkan dengan proses integrasi kedua fungsi untuk mendapatkan nilai pada posisi tersebut, dalam hal ini bisa dinyatakan dalam persamaan berikut

$$y(p) = \int_{-\infty}^{\infty} h(t)x(p-t)dt \qquad (8\text{-}5)$$

Agar lebih mudah kita evaluasi pada beberapa nilai berikut ini.

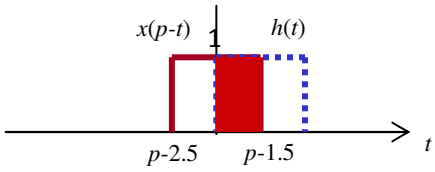
1. Untuk p-1.5< 0 atau p<1.5



$$y(p) = 0$$

Gambar 8.4a. Proses konvolusi posisi 1

2. Untuk p-1.5 > 0 dan p-2.5 < 0, atau 1.5< p <2.5



Gambar 8.4b. Proses konvolusi posisi 2

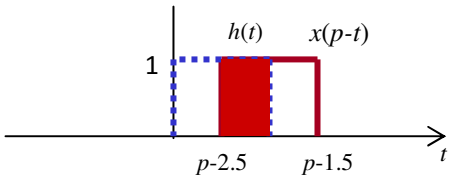
$$y(p) = \int_{-\infty}^{\infty} h(t)x(p-t)dt, \quad y(p) \neq 0 \; untuk \quad 0 < t < p-1.5$$

$$y(p) = \int_{0}^{p-1.5} h(t)x(p-t)dt$$

$$y(p) = \int_{0}^{p-1.5} 1.1dt$$

$$y(p) = [t]_{0}^{p-1.5} = p-1.5$$

3. Untuk p-1.5 > 1 dan p-2.5<1, atau 2.5 < p <3.5



Gambar 8.4c. Proses konvolusi posisi 3

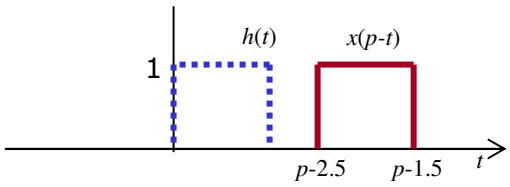
$$y(p)=\int_{-\infty}^{\infty} h(t)x(p-t)dt, \quad y(p)\neq 0 \; untuk \quad p-2.5<t<1$$

$$y(p)=\int_{p-2.5}^{1} h(t)x(p-t)dt$$

$$y(p)=\int_{p-2.5}^{1} 1.1dt$$

$$y(p)=[t]_{p-2.5}^{1}=1-(p-2.5)=3.5-p$$

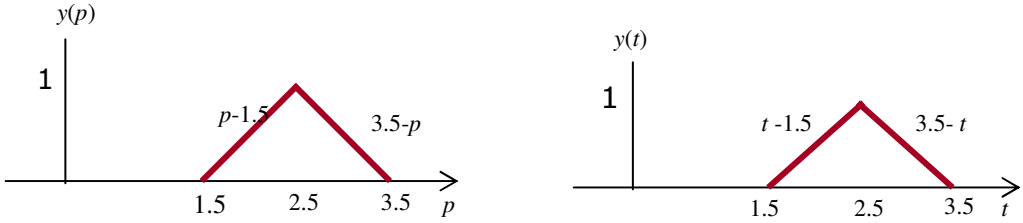
4. Untuk p - 2.5 > 1, p > 3.5



Gambar 8.4d. Proses konvolusi posisi 4

$$y(p)=0$$

- Sehigga pada akhirnya dengan menganalogikan nilai y(t) = y(p) menjadi sepeti pada gambar berikut



Gambar 8.4e. Hasil proses konvolusi

III. Perangkat Yang Diperlukan

- PC yang dilengkapi dengan perangkat multimedia (sound card, Microphone, Speaker active, atau headset)
- Sistem Operasi Windows dan Perangkat Lunak Matlab yang dilengkapi dengan tool box DSP

IV. Langkah Percobaan

4.1 Konvolusi Dua Sinyal Sinus

Disini kita mencoba untuk membangkitkan dua sinyal sinus dan melakukan operasi konvolusi untuk keduanya. Langkah yang harus anda lakukan adalah sebagai berikut:

1. Buat program untuk membangkitkan dua gelombang sinus seperti berikut:

```
L=input('Banyaknya titik sampel(>=20): ');
f1=input('Besarnya frekuensi gel 1 adalah Hz: ');
f2=input('Besarnya frekuensi gel 2 adalah Hz: ');
teta1=input('Besarnya fase gel 1(dalam radiant): ');
teta2=input('Besarnya fase gel 2(dalam radiant): ');
A1=input('Besarnya amplitudo gel 1: ');
A2=input('Besarnya amplitudo gel 2: ');
%Sinus pertama
t=1:L;
t=2*t/L;
y1=A1*sin(2*pi*f1*t + teta1*pi);
subplot(3,1,1)
stem(y1)
%Sinus kedua
t=1:L;
t=2*t/L;
y2=A2*sin(2*pi*f2*t + teta2*pi);
subplot(3,1,2)
stem(y2)
```

2. Coba jalankan program anda dan isikan seperti berikut ini:

```
Banyaknya titik sampel(>=20): 20
Besarnya frekuensi gel 1 adalah Hz: 1
Besarnya frekuensi gel 2 adalah Hz: 0.5
Besarnya fase gel 1(dalam radiant): 0
Besarnya fase gel 2(dalam radiant): 0.5
Besarnya amplitudo gel 1: 1
Besarnya amplitudo gel 2: 1
```

Perhatikan tampilan yang dihasilkan. Apakah ada kesalahan pada program anda?

3. Lanjutkan dengan menambahkan program berikut ini pada bagian bawah program yang anda buat tadi.

```
subplot(3,1,3)
stem(conv(y1,y2))
```

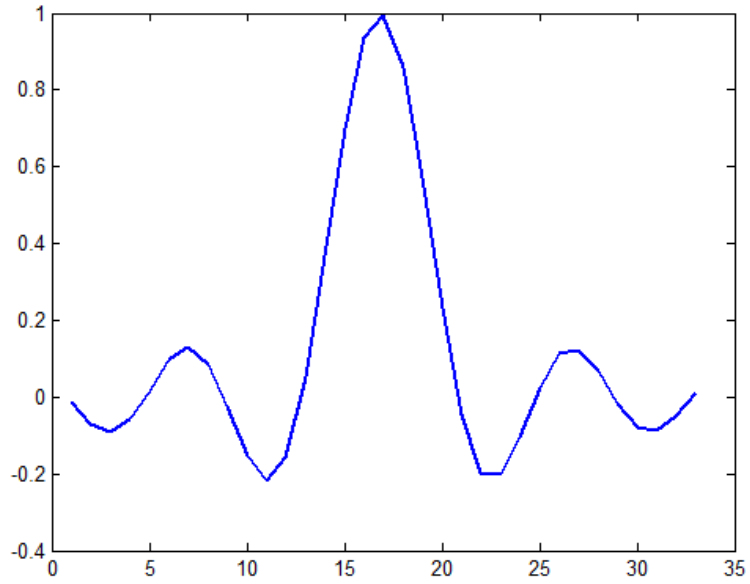
4. Jalankan program anda, dan kembali lakukan pengisian seperti pada langkah ke 3. Lihat hasilnya apakah anda melihat tampilan seperti berikut?
5. Ulangi langkah ke 4, dengan menetapkan nilai sebagai berikut: L=50. w1=w2=2, teta1=1.5, teta2=0.5, dan A1=A2=1. Apa yang anda dapatkan? Apakah anda mendapatkan hasil yang berbeda dari program sebelumnya? Mengapa ?

4.2 Konvolusi Sinyal Bernoise dengan Raise Cosine

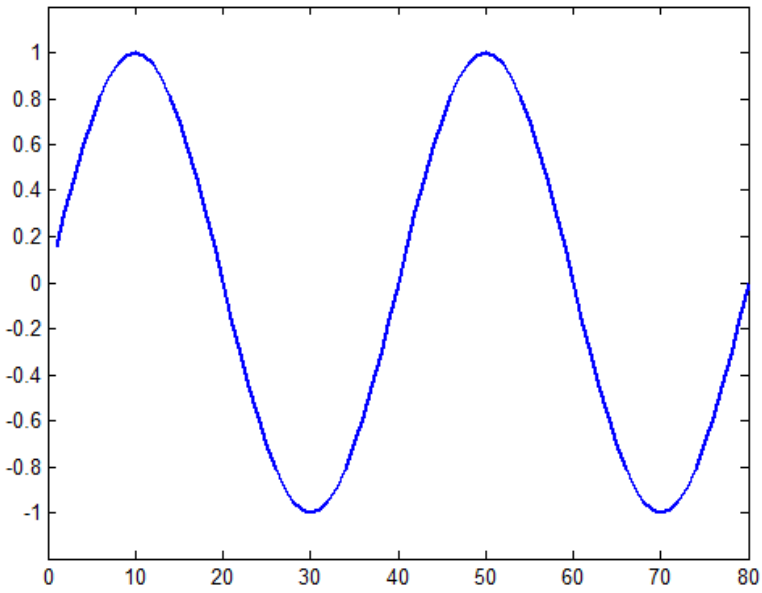
Sekarang kita mulai mencoba utnuk lebih jauh melihat implementasi dari sebuah operasi konvolusi. Untuk itu ikuti langkah-langkah berikut.

1. Bangkitkan sinyal raise cosine dan sinyal sinus dengan program berikut.

```
%File Name: Noisin.m
%convolusi sinyal sinus bernoise dengan raise cosine;
n=-7.9:.5:8.1;
y=sin(4*pi*n/8)./(4*pi*n/8);
figure(1);
plot(y,'linewidth',2)
t=0.1:.1:8;
x=sin(2*pi*t/4);
figure(2);
plot(x,'linewidth',2)
```



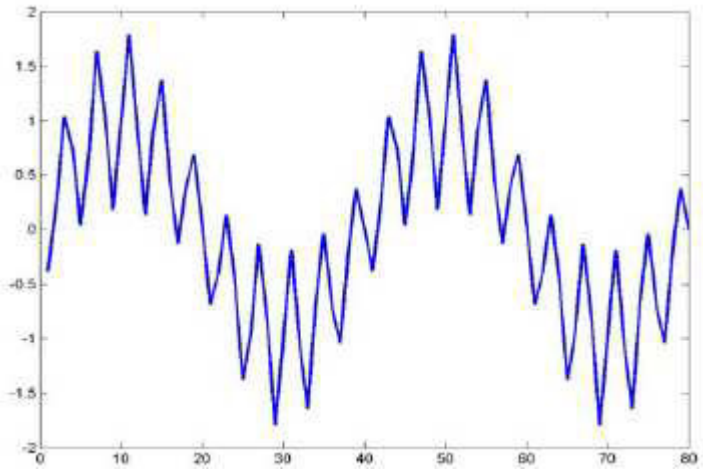
Gambar 8.5. Sinyal raise cosine



Gambar 8.6. Sinyal Sinus Asli

2. Tambahkan noise pada sinyal sinus.

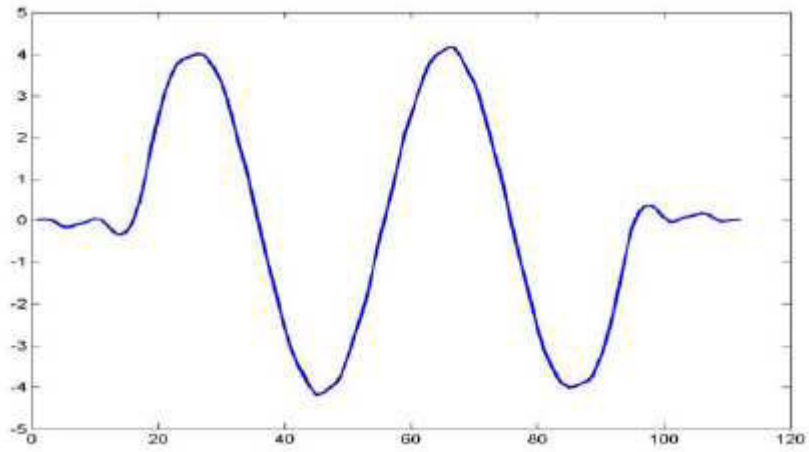
```
t=0.1:.1:8;
noise=0.5*randn*sin(2*pi*10*t/4) ;
x_n=sin(2*pi*t/4)+noise;
figure(3);
plot(x_n,'linewidth',2)
```



Gambar 8.7. Sinyal sinus bernoise

3. Lakukan konvolusi sinyal sinus bernoise dengan raise cosine, perhatikan apa yang terjadi?

```
xy=conv(x_n,y);
figure(4);
plot(xy,'linewidth',2)
```



Gambar 8.8. Sinyal Hasil konvolusi

V. Data dan Analisa

Anda telah melakukan berbagai langkah untuk percobaan operasi konvolusi sinyal kontinyu. Yang harus anda lakukan adalah menjawab setiap pertanyaan yang ada pada langkah percobaan. Tulis semua komentar dan analisa sebagai penjelasan dari hasil percobaan anda.

1. Apa arti konvolusi 2 buah sinyal kontinyu dalam simulasi tersebut diatas? bagaimana jika amplitudo masing-masing sinyal diubah? Beri penjelasan.
2. Jelaskan pengaruh konvolusi terhadap sinyal asli?

MODUL 9
ANALISA SINYAL DOMAIN FREKUENSI

I. Tujuan Instruksional Khusus:

- Mengamati sinyal dalam domain waktu dan domain frekuensi dengan menggunakan library FFT

II. Teori Analisa Sinyal pada Domain Frekuensi

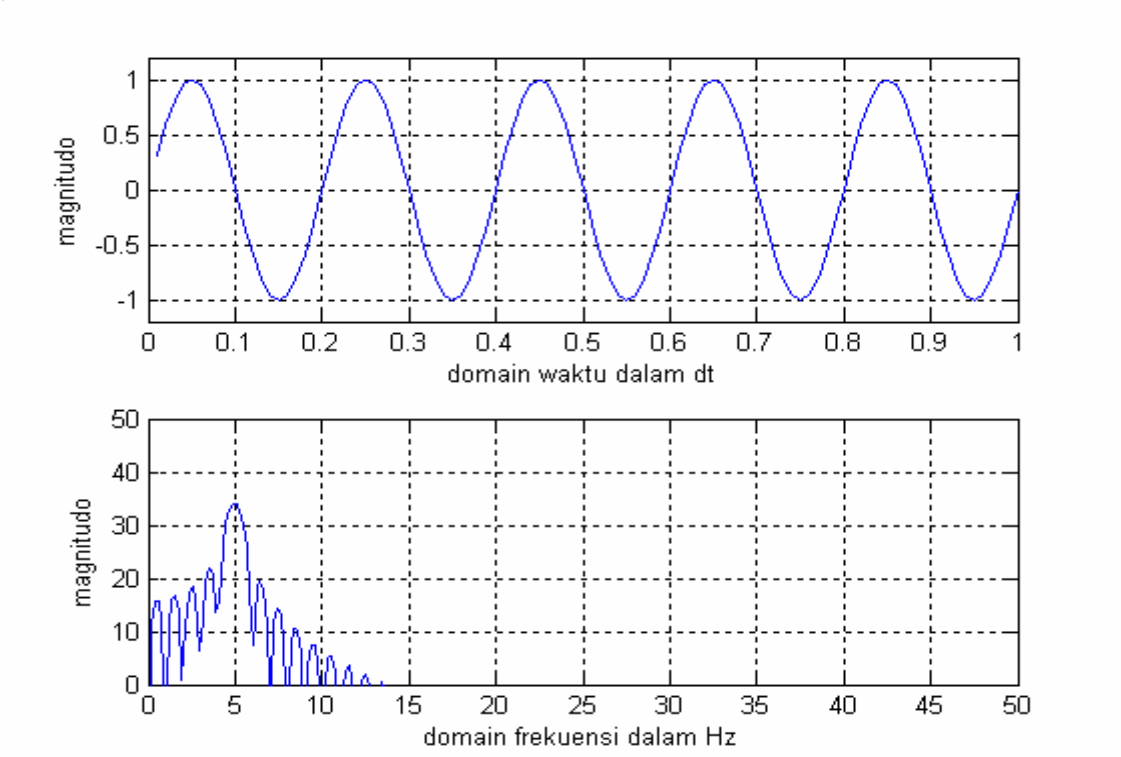
2.1 Transformasi Fourier

Satu bentuk transformasi yang umum digunakan untuk merubah sinyal dari domain waktu ke domain frekuensi adalah dengan transformasi Fourier:

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{j\omega t} dt \qquad (9\text{-}1)$$

Persamaan ini merupakan bentuk transformasi Fourier yang siap dikomputasi secara langsung dari bentuk sinyal x(t).

Sebagai contoh, anda memiliki sinyal sinus dengan frekuensi 5 Hz dan amplitudo 1 Volt. Dalam domain waktu anda akan melihat seperti pada Gambar 9.1 bagian atas. Sementara dalam domain frekuensi akan anda dapatkan seperti pada bagian bawah. Untuk memperoleh hasil seperti gambar tersebut anda dapat memanfaatkan *library fft* yang tersedia pada Matlab.



Gambar 9.1. Sinyal sinus dalam domain waktu dan domain frekuensi

2.2 Analisa Spektrum

Salah asatu proses analisa di dalam domain frekuensi bisa dilakukan dengan cara menghitung frekuensi dari suatu sinyal, dalam hal ini bisa memanfaatkan bentuk waktu diskrit dari analisa Fourier dapat digunakan, yang kemudian lebih disempurnakan dengan suatu algoritma yang kita kenal sebagai *Fast Fourier transform* (FFT). Secara umum teknik ini merupakan pendekatan yang terbaik untuk transformasi. Dalam hal ini input sinyal ke window ditetapkan memmiliki panjang 2m. Anda dapat memilih analisis window yang akan digunakan. Output dari syntax *fft*(*x*,*n*) merupakan sebuah vector komplek, dengan n amplitudo komplek dari 0 Hz sampai dengan sampling frekuensi yang digunakan.

III. Perangkat Yang Diperlukan

- PC multimedia yang sudah dilengkapi dengan OS Windows
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box DSP

IV. Langkah Percobaan

4.1. Fenomena Gibb

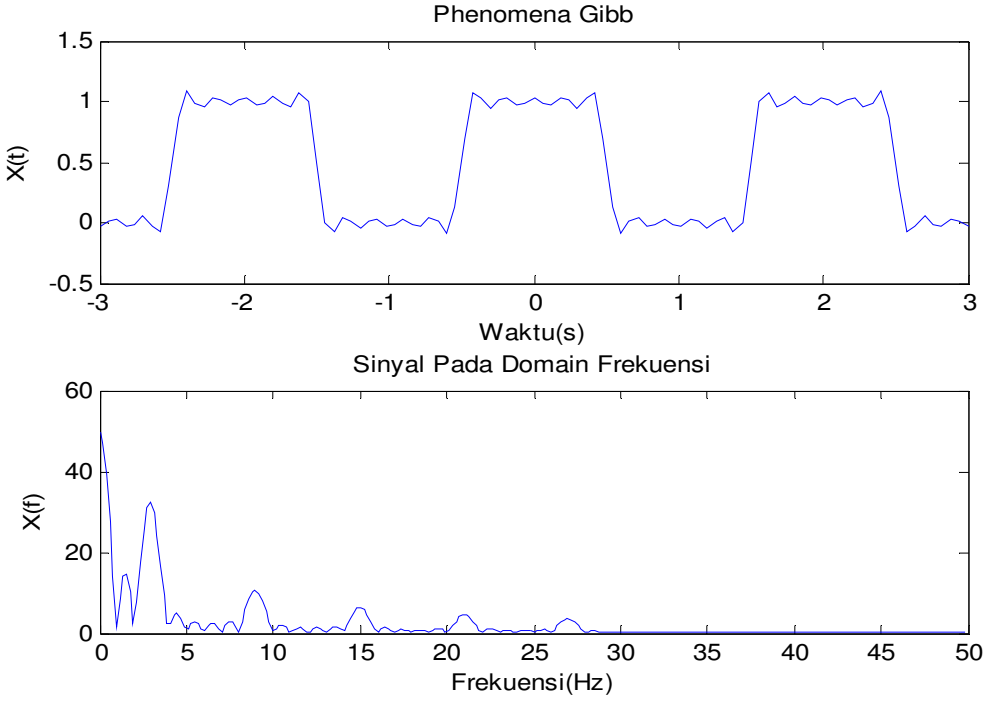
Kita mulai dengan mencoba memahami suatu masalah yang popular dalam pengolahan sinyal, yaitu fenomena Gibb. Untuk memahami bagaimana penjelasan fenomena tersebut, anda ikuti langkah berikut.

1. Bangkitkan sebuah sinyal sinus dengan cara seperti berikut

```
%File name: fen_Gibb.m
clc;clf;
t=-3:6/100:3;
N=input('Masukan Jumlah Sinyal Yang Dikehendaki = ');
c0=0.5;
w0=pi;
Fs=100;
xN=c0*ones(1,length(t));
for n=1:2:N;
   theta=((-1)^((n-1)/2)-1)*pi/2;
   xN=xN+2/n/pi*cos(n*w0*t+theta);
end
subplot(211)
plot(t,xN)
title('Phenomena Gibb')
xlabel('Waktu(s)')
ylabel('X(t)')
%Transformasi
```

```
xf=fft(xN,512);
w=(0:255)/256*(Fs/2);
subplot(212)
plot(w,abs(xf(1:256)))
title('Sinyal Pada Domain Frekuensi')
xlabel('Frekuensi(Hz)')
ylabel('X(f)')
```

Masukan jumlah sinyal N = 10, maka akan terlihat tampilan seperti gambar 2 :



Gambar 9.2. Sinyal dan phenpmena Gibb

2. Jalankan lagi program anda, dengan cara memberi jumlah masukan sinyal yang berbeda, misalnya 15, 35 dan 50. Apa yang anda dapatkan?
3. Dari langkah percobaan anda ini, fenomena apa yang didapatkan tentang sinyal persegi ? Apa kaitannya dengan sinyal sinus? Perhatikan pada gambar hasil running program akan tampak terjadinya phenomena Gibb.

4.2. Pengamatan Frekuensi Pada Sinyal Tunggal

Disini anda akan mengamati bentuk sinyal dalam domain waktu dan domain frekuensi dengan memanfaatkan library fft yang ada dalam DSP Toolbox Matlab. Apabila ada yang kurang jelas dengan perintah yang diberikan dalam petunjuk, jangan pernah sungkan menanyakan kepada dosen pengajar. Selanjutnya ikuti langkah berikut.

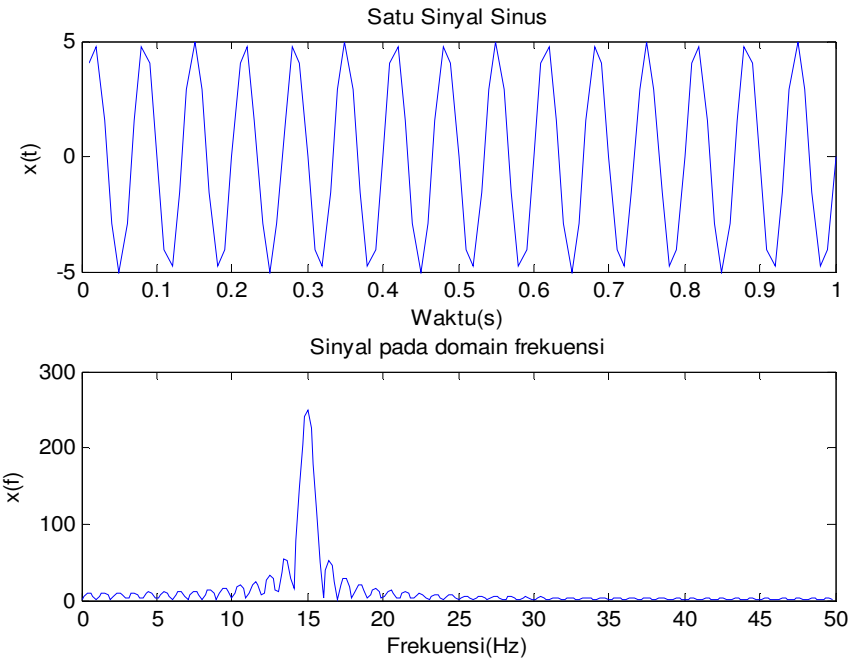
1. Bangkitkan sinyal sinus yang memiliki frekuensi f = 15 Hz, dan amplitudo 5 Volt.

```
Fs=100;
t=(1:100)/Fs;
f=15; A=5 ;
s=A*sin(2*pi*f*t);
subplot(2,1,1)
plot(t,s)
xlabel('Waktu (s)')
```

2. Lanjutkan langkah ini dengan memanfaatkan fungsi fft untuk mentranformasi sinyal ke dalam domain frekuensi

```
S=fft(s,512);
w=(0:255)/256*(Fs/2);
subplot(2,1,2)
plot(w,abs(S(1:256)))
xlabel('Frekuensi (Hz)')
```

3. Jalankan program anda maka akan terlihat hasil seperti Gambar 3.
4. Cobalah anda merubah nilai f=10, 20 dan 30 Apa yang anda lihat pada gambar sinyal anda?
5. Cobalah merubah nilai amplitudo dari 5 volt menjadi 7, 15 atau 20. Apa yang terjadi pada sinyal anda?



Gambar 9.3. Sinyal sinus tunggal

4.3. Pengamatan Frekuensi Pada Kombinasi 2 Sinyal

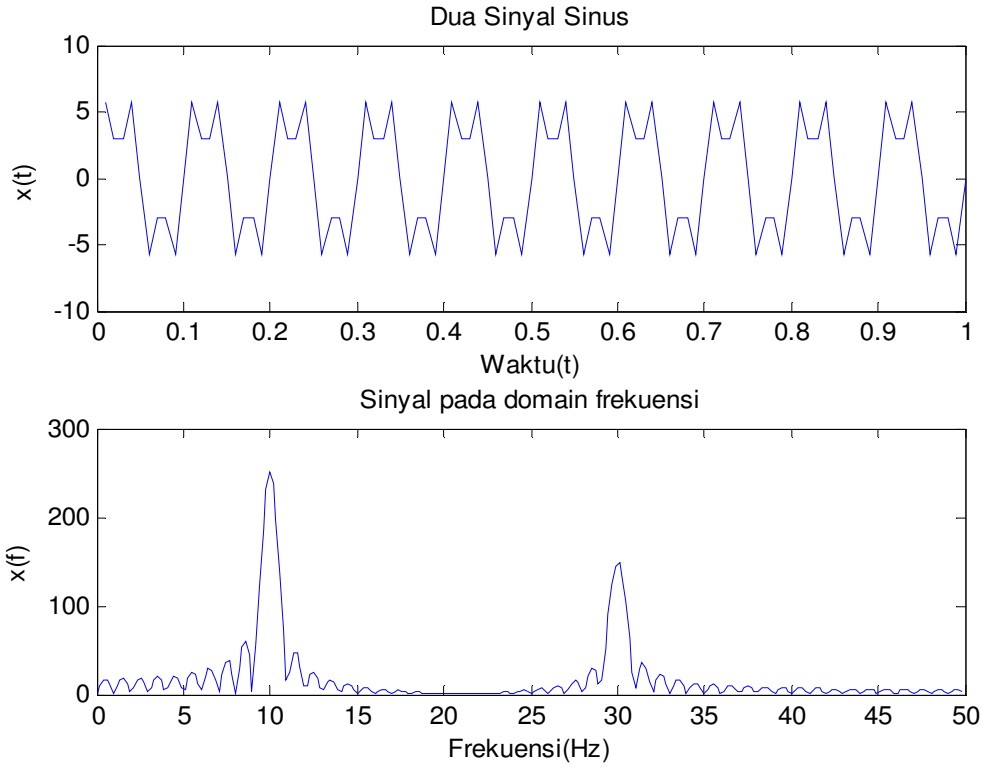
Anda telah mengetahui cara mengamati sinyal dalam doain waktu dan frekuensi. Pada percobaan berikut ini anda coba bangkitkan 2 sinyal sinus dengan frekuensi f1 dan f2. Sementara nilai amplitudo dapat anda lihat pada listing program berikut ini.

1. Caranya adalah dengan mengetik program berikut ini

```
clc;clf;
fs=100;
t=(1:400)/fs;
f1=10;
s1=5*sin(2*pi*f1*t);
f2=30;
s2=3*sin(2*pi*f2*t);
s=s1+s2;
subplot(211)
plot(t,s)
title('Dua Sinyal Sinus')
xlabel('Waktu(s)')
ylabel('x(t)')
S=fft(s,512);
w=(0:255)/256*fs/2;
Sab=abs(S);
subplot(212)
plot(w,Sab(1:256))
title('Sinyal pada domain frekuensi')
xlabel('Frekuensi(Hz)')
ylabel('x(f)')
```

Jalankan program, maka akan terlihat hasil seperti Gambar 4.

2. Ubahlah nilai f2 =20, 35 dan 50, sedangkan f1 dan amplitudo tetap. Apa yang anda dapatkan dari langkah ini? Plot semua hasil running program.
3. Coba ubah nilai amplitudo pada sinyal pertama menjadi 7 , 10 atau 15, sedangkan f1 dan f2 tetap seperti program pertama. Apa yang anda dapatkan dari langkah ini? Plot semua hasil running program anda.



Gambar 9.4. Dua sinyal sinus dalam domain frekuensi

4.4. Pengamatan Frekuensi Pada Kombinasi 4 Sinyal

1. Pada percobaan berikut ini anda coba bangkitkan 4 sinyal sinus dengan frekuensi f1, f2, f3, dan f4. Sementara nilai amplitudo dapat anda lihat pada listing program berikut ini.

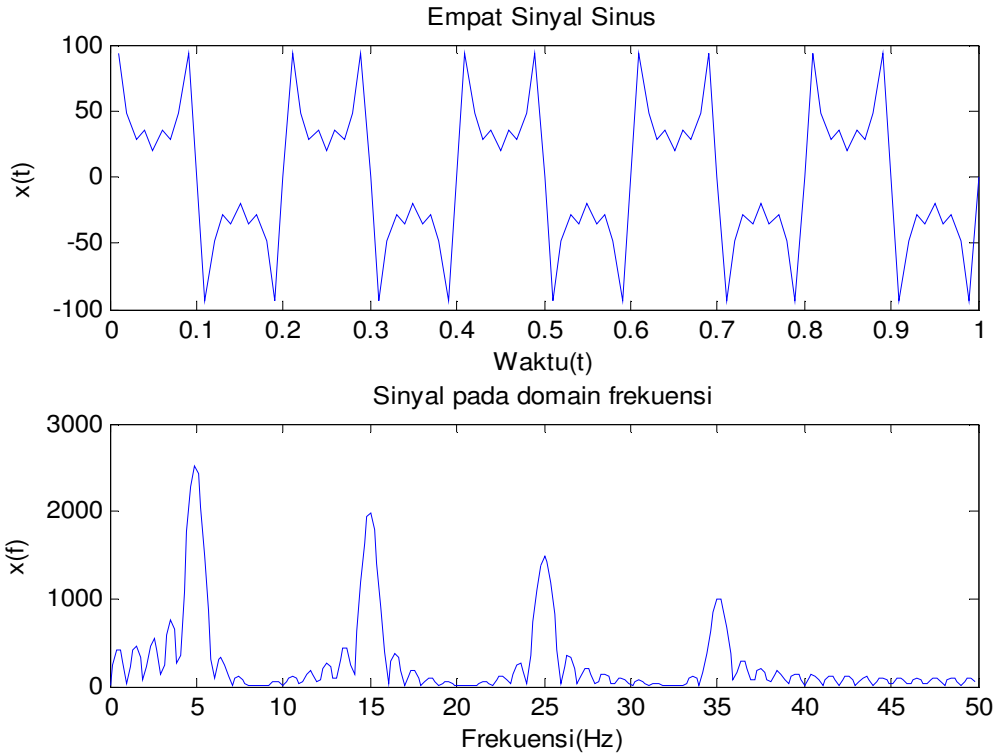
   Caranya adalah dengan mengetik program berikut ini :

```
clc;clf;
fs=100;
t=(1:100)/fs;
f1=5;
s1=50*sin(2*pi*f1*t);
f2=15;
s2=40*sin(2*pi*f2*t);
f3=25;
s3=30*sin(2*pi*f3*t);
f4=35;
s4=20*sin(2*pi*f4*t);
s=s1+s2+s3+s4;;
subplot(211)
plot(t,s)
title('Empat Sinyal Sinus')
xlabel('Waktu(s)')
```

```
ylabel('x(t)')
S=fft(s,512);
w=(0:255)/256*fs/2;
Sab=abs(S);
subplot(212)
plot(w,Sab(1:256))
title('Sinyal pada domain frekuensi')
xlabel('Frekuensi(Hz)')
ylabel('x(f)')
```

Perhatikan bentuk sinyal yang dihasilkan dari langkah tersebut diatas pada Gambar 5 :

2. Ubah nilai f2 =20, f3 = 30 dan f4 =30, sedangkan amplituda tetap. Apa yang anda dapatkan dari langkah ini? Plot semua hasil running dari program anda.



Gambar 9.5. Empat sinyal sinus dalam domain frekuensi

4.5. Pengamatan Frekuensi Pada Kombinasi 6 Sinyal

Pada percobaan berikut ini anda coba bangkitkan 4 sinyal sinus dengan frekuensi f1, f2, f3, f4, f5, dan f6. Sementara nilai amplitudo dapat anda lihat pada listing program berikut ini. Caranya adalah dengan mengetik program berikut ini

```
Fs=100;
t=(1:200)/Fs;
f1=2;
```

```
s1=20*sin(2*pi*f1*t);
f2=5;
s2=15*sin(2*pi*f2*t);
f3=15;
s3=10*sin(2*pi*f3*t);
f4=20;
s4=7*sin(2*pi*f4*t);
f5=35;
s5=5*sin(2*pi*f5*t);
f6=45;
s6=3*sin(2*pi*f6*t);
s=s1+s2+s3+s4+s5+s6;
subplot(211)
plot(t,s)
title('Enam Sinyal Sinus')
xlabel('Waktu(s)')
ylabel('x(t)')
S=fft(s,512);
w=(0:255)/256*fs/2;
Sab=abs(S);
subplot(212)
plot(w,Sab(1:256))
title('Sinyal pada domain frekuensi')
xlabel('Frekuensi (Hz)')
ylabel('x(f)')
```

Catat dan amati bentuk sinyal yang dihasilkan dari langkah anda tersebut. Apa yang anda dapatkan dari langkah ini? Plot semua hasil running dari program anda.

4.6. Pengamatan Frekuensi Pada Sinyal Audio

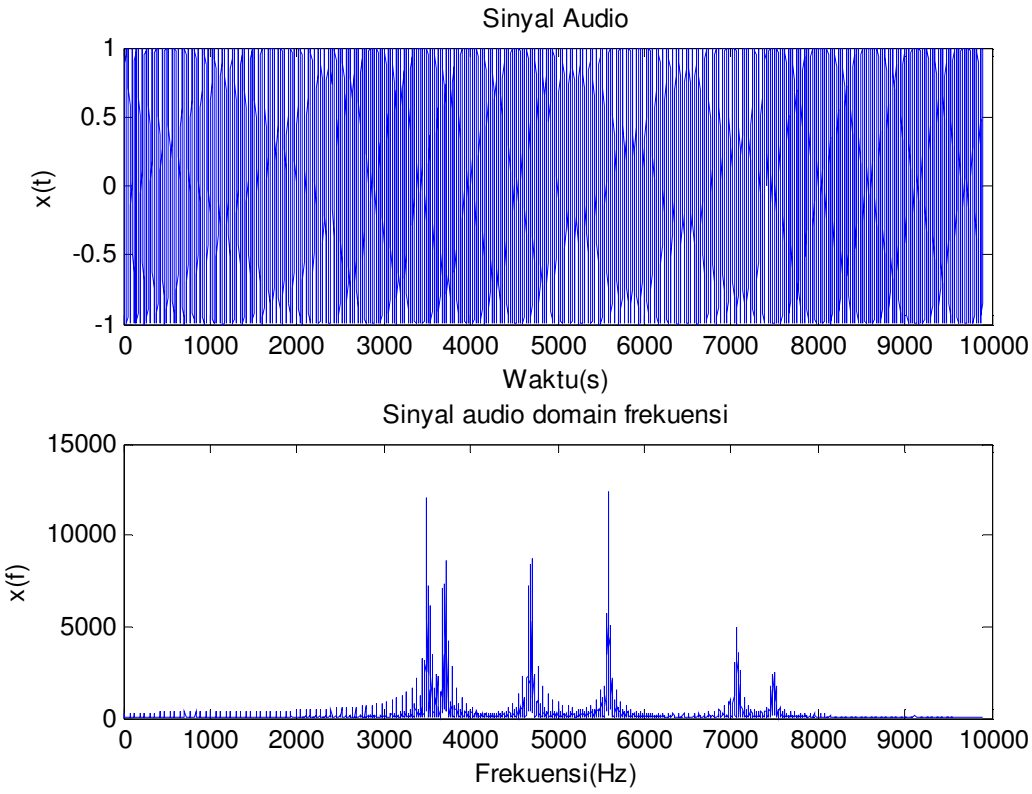
Disini dicoba untuk melihat sinyal yang lebih real dalam kehidupan kita. Untuk itu ikuti langkah berikut.

1. Buat program pemanggil file audio *.wav sebagai berikut :

```
clc;clf;
[y,Fs]=wavread('gundul_pacul.wav');
Fs=16000;
subplot(211)
plot(y(100:10000))
```

```
title('Sinyal Audio')
xlabel('Waktu(s)')
ylabel('x(t)')
S=fft(y);
%w=(0:255)/256*fs/2;
Sab=abs(S);
subplot(212)
plot(Sab(100:10000))
% plot(Sab)
title('Sinyal audio domain frekuensi')
xlabel('Frekuensi(Hz)')
ylabel('x(f)')
```

2. Jalankan program anda maka akan terlihat hasil seperti Gambar 6

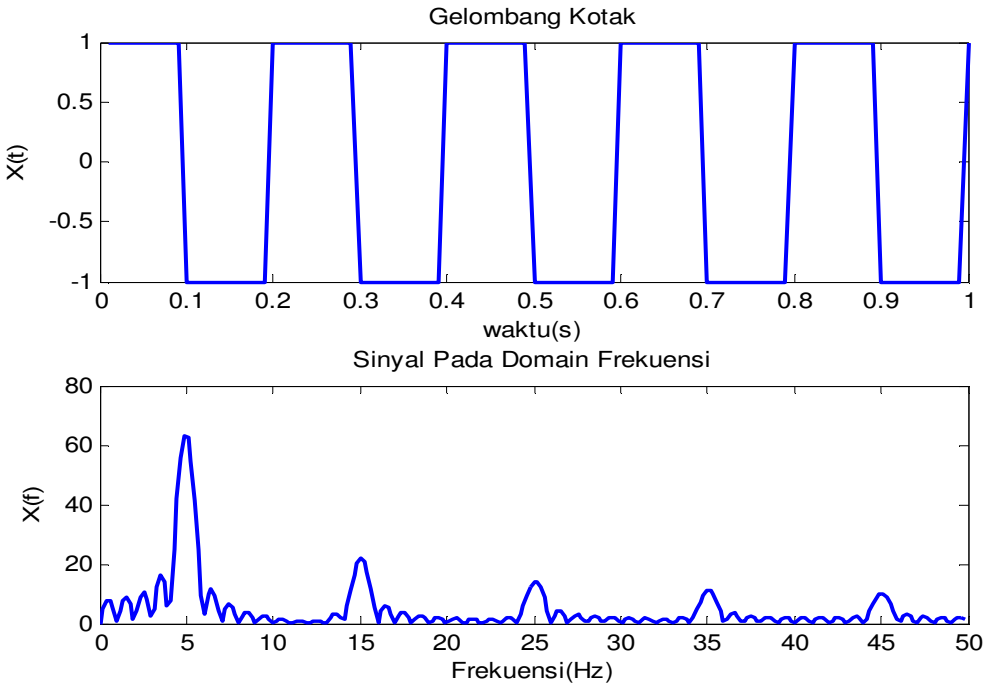


Gambar 9.6. Sinyal audio dalam domain waktu dan frekuensi

3. Ubah nilai Fs menjadi 4000, simpan lagi file wav dengan nama ”baru.wav”, menggunakan perintah wavwrite(y,’baru.wav’) kemudian panggil lagi dan plot hasil sinyalnya dalam domain waktu dan frekuensi seperti program diatas.

4.7. Pengamatan Frekuensi Pada Sinyal Kotak

1. Buat program untuk merubah sinyal kotak dalam domain waktu, menjadi sebuah sinyal dalam domain frekuensi seperti yang terlihat pada Gambar 7. Ketentuan program sebagai berikut : Fs=100, f=5, amplitudo=1.



Gambar 9.7. Sinyal kotak dalam domain waktu dan domain frekuensi

2 Ubahlah frekuensi berturut-turut 10, 20,30,40 dan 50, dengan Fs dan amplitudo tetap, kemudian plot semua gambar hasil running program anda.

V. Analisa Data

Seperti biasa diakhir pertemuan anda harus menyelesaikan laporan, dan jangan lupa menjawab pertanyan berikut :

1. Apa sebenarnya fenomena Gibb itu?
2. Apa hubungan sinyal persegi dengan sinyal sinus?
3. Jika anda hubungkan dengan mata kuliah teknik modulasi digital, coba anda jelaskan mengapa sinyal persegi tidak langsung digunakan memodulasi carrier?
4. Coba anda buat record suara anda, terserah berupa vokal atau ucapan yang lain, dan amati bentuk spektrumnya.

MODUL 10
TRANSFORAMSI DOMAIN FREKUENSI KE WAKTU

I. Tujuan Instruksional Khusus:

- Melakukan transformasi sinyal dalam domain frekuensi ke dalam domain waktu menggunakan library IFFT

II. Teori Inverse Fourier Transform

Satu bentuk transformasi yang umum digunakan untuk merubah sinyal dari domain frekuensi ke domain waktu adalah Inverse Transformasi Fourier, rumus transformasi tersebut merupakan pasangan dari transformasi fourier seperti yang telah dibahas pada modul sebelumnya. IFT atau dalam pelaksanaan komputasi menggunakan matlab dikenal dengan IFFT (Inverse Fast Fourier Transform), seperti pada persamaan dibawah ini.

$$x(t) = \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega \qquad (10\text{-}1)$$

Persamaan ini merupakan bentuk library yang dapat digunakan langsung untuk proses komputasi menggunakan perintah IFFT.

III. Perangkat Yang Diperlukan

- PC multimedia yang sudah dilengkapi dengan OS Windows
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box DSP

IV. Langkah Percobaan

4.1. Proses Konversi Sederhana Pada Sinyal Dasar

1. Anda buat sebuah program pembangkitan sinyal sinus dalam domain waktu dan domain frekuensi seperti berikut

```
clc
t=0:.001:2;
x_t=sin(2*pi*t);
figure(1);
plot(x_t);axis([0 2010 -1.2 1.2])
X_F=fft(x_t);
figure(2);
plot(abs(X_F));axis([-100 2100 -10 1100])
```

2. Tambahkan perintah berikut ini untuk mengkonversikan kembali dari domain frekuensi ke dalam domain waktu.

```
x_tt=ifft(X_F);
figure(3);
plot(x_tt);axis([0 2010 -1.2 1.2])
```

3. Anda buat sebuah program pembangkitan sinyal persegi dalam domain waktu diskrit dan domain frekuensi seperti berikut

```
x=[1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0];
stem(x)
figure(1);
stem(x);axis([0 20 -.5 1.5])
Xf=fft(x);
figure(2);
stem(Xf)
stem(abs(Xf));axis([0 20 -.5 5.5])
figure(3);
xtt=ifft(Xf);
stem(xtt);axis([0 20 -.5 1.5])
```

4. Tambahkan perintah berikut ini untuk mengkonversikan kembali dari domain frekuensi ke dalam domain waktu.

```
figure(3);
xtt=ifft(Xf);
stem(xtt);axis([0 20 -.5 1.5])
```

Untuk sementara anda tidak perlu berfikir tentang satuan yang digunakan di dalam sumbu koordinat, karena dalam hal ini masih dalam satuan sampel.

5. Buat programbaru untuk pembangkitan sinyal persegi dengan cara seperti berikut

```
clc
Fs=10000;
t = 0:1/Fs:.0625;
y = square(2*pi*30*t);
figure(1);
plot(t,y);axis([0 0.07 -1.5 1.5])
xlabel('(a) Sinyal input kotak');
```

```
Yf = fft(y);
Yf_dB = 20*log10(abs(Yf));
figure(2)
f=1:(length(Yf)/2);
Yf_dBx=Yf_dB(1:length(Yf)/2);
plot(0.5*Fs*(f/length(Yf)),Yf_dBx);
xlabel('(b) Konversi ke domain frekuensi');
```

6. Tambahkan kode program berikut ini untuk konversi ke domain waktu.

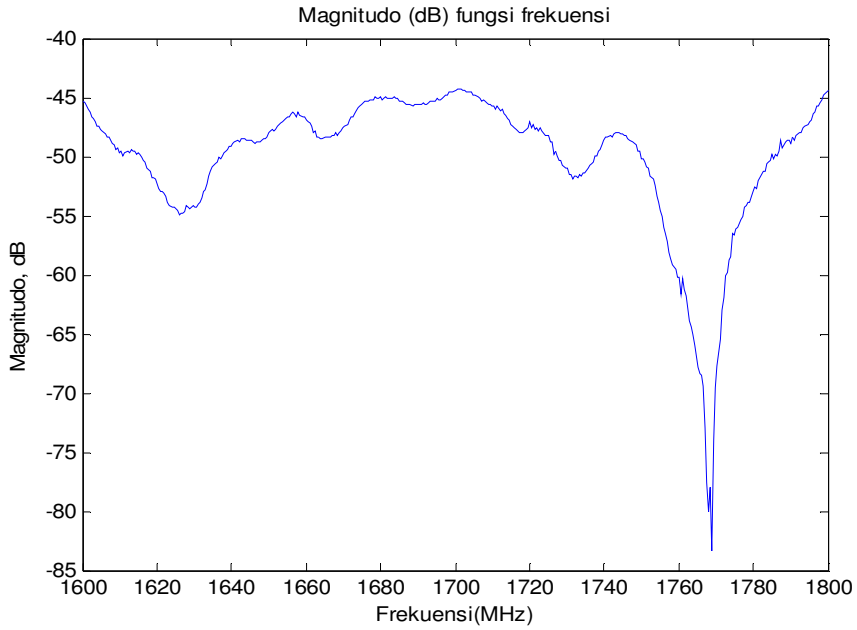
```
figure(3)
ytt=ifft(Yf);
plot(t,ytt);axis([0 0.07 -1.5 1.5])
xlabel('(c) Hasil pengembalian ke domain waktu');
```

Berikan penjelasan dari gambar yang dihasilkan, dan anda coba untuk prose berkutnya adalah melakukan pembangkitan sinyal segitiga dan sinyal gigi gergaji. Anda lakukan langkah-langkah seperti sebelumnya.

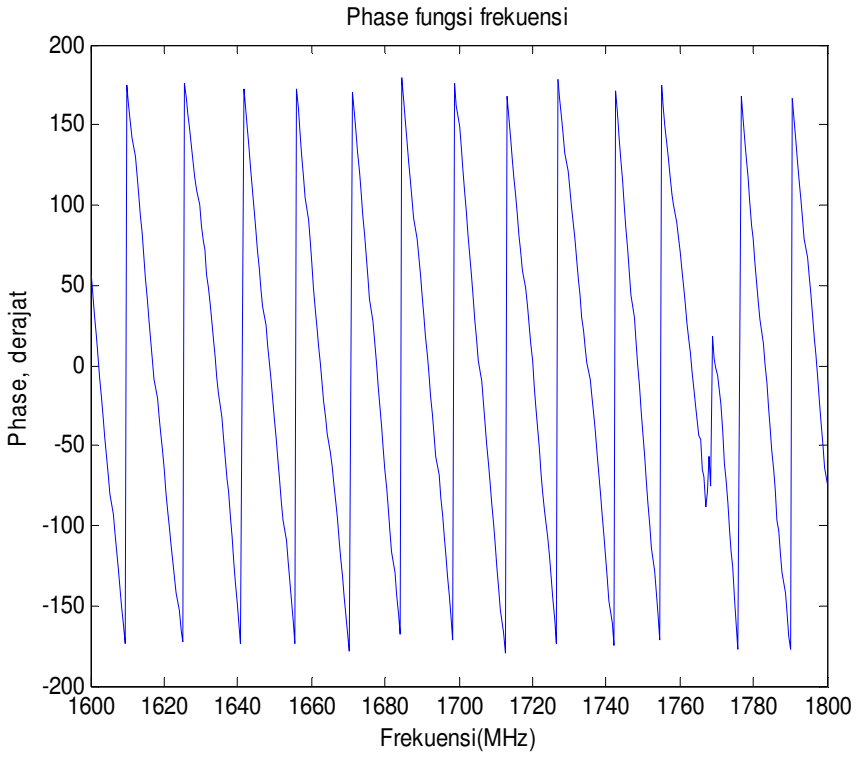
4.2. Pengolahan dari Input data Text

Data sudah disiapkan dalam bentuk .txt, pada data tersebut terdapat 401 sampel data dengan masing-masing data merupakan besaran vektor yaitu mempunyai amplitudo dan fase. Amplitudo dalam rasio dB dan fase dalam besaran derajat. Buat program dalam m-file Urutan program seperti berikut:

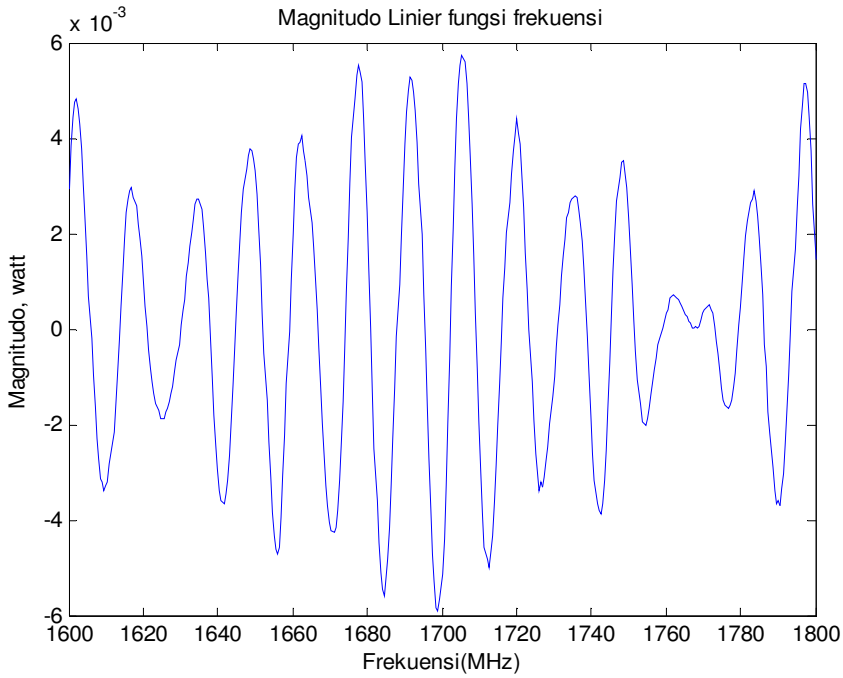
1. Lakukan pemanggilan data dengan perintah **load namafile,** beri variable dengan nama **data.**
2. Data yang dipanggil terdiri dari 3 kolom (kolom1: frekuensi, kolom2 : magnitudo dalam dB, kolom3 : phase dalam derajat)
3. Ubah magnitudo (dB) kedalam satuan linier dengan proses **antilog**
4. Buat window untuk proses IFFT, dengan cara menggunakan fungsi hamming sebagai window-nya: **hamming(panjang data).** Beri nama variabel **wind**.
5. Kalikan magnitudo linier dengan window, beri variabel hasil perkalian tersebut dengan nama **htw**
6. Lakukan proses IFFT, dengan fungsi IFFT2(htw,panjang data,1) dan beri nama variabelnya dengan **mifft**
7. Ambil absolute dari **mifft** tersebut dengan nama variabel **mifftabs**
8. Ubah nilai mifftabs kedalam dB, dengan proses **log** kemudian beri variabel dengan nama **mifftdb**
9. Tampilkan semua hasil running program, beberapa hasilnya akan tampil seperti pada Gambar berkut ini.



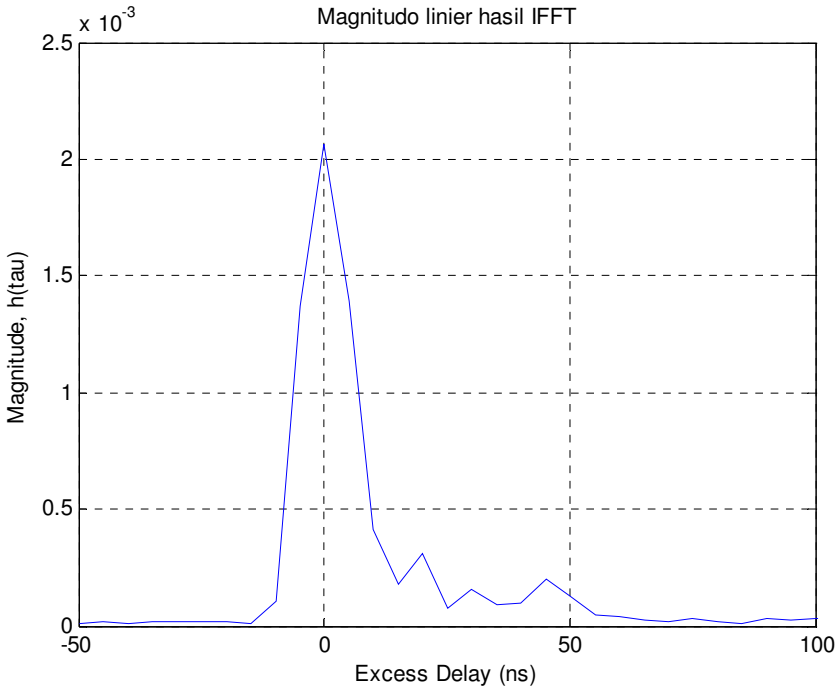
Gambar 9.1 Tampilan Magnitudo (dB) fungsi frekuensi



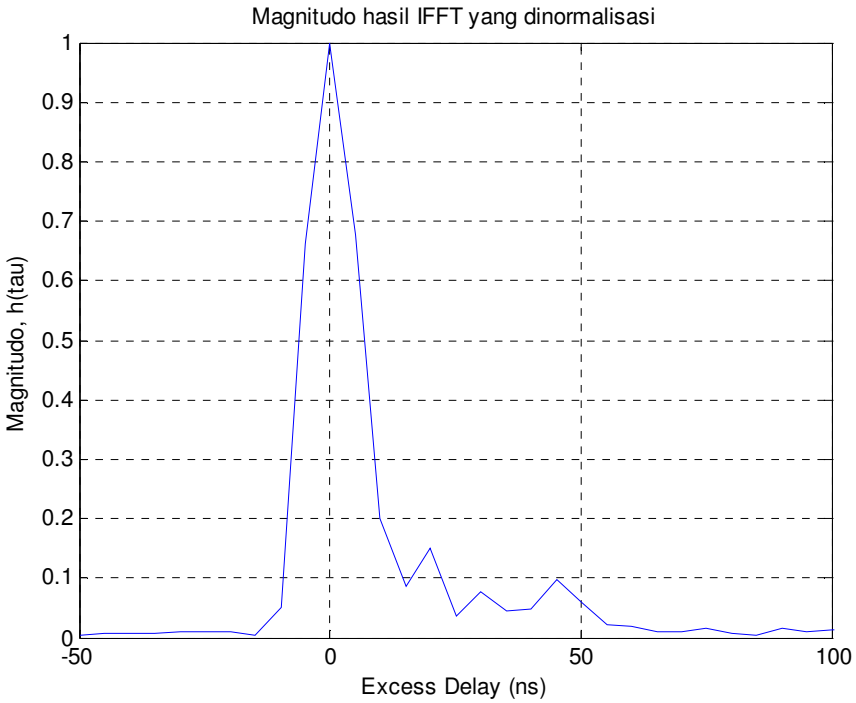
Gambar 9.2. Tampilan Phase fungsi frekuensi



Gambar 9.3. Magnitudo Linier fungsi frekuensi



Gambar 9.4. Magnitudo linier hasil IFFT



Gambar 9.5. Magnitudo hasil IFFT yang dinormalisasi

Untuk membantu anda dalam praktikum, anda bisa menggunakan listing program berikut ini:

```
%===========
%Proses IFFT
%===========
clf;clc; clear all;
data=load('D113S21.txt');%Memanggil File data
f=data(:,1);%Frekuensi
m=data(:,2);%Magnitude
ph=data(:,3);%Phase
n=401;%Jumlah data
window=hamming(n);  % Window Hamming
mrec=((10.00).^(data(:,2)./10)).*exp(i*(data(:,3)*(pi/180)));%Merubah magnitude dari dB ke linier (antilog)
htw=window.*mrec;%Perkalian magnitude linier dengan window
ht=abs(htw);
fs=2e+8;
t=1:n;
tm=t./fs;
mifft=ifft2(htw,n,1);%Proses transformasi domain frekuensi ke domain waktu
mabs=abs(mifft);
```

```
mabmax=max(mabs);
mabsdb=10*log10(mabs);%Magnitude dalam dB
for i=1:n;
mabn(i)=mabs(i)./mabmax(1);
mabdb(i)=10.*log10(mabn(i));
end
delaykabel=(20/(0.66*3e8))+2.4/3e8+((2*0.8824)/3e8);
tn=((tm-delaykabel)*1e9);
na=delaykabel/5e-9;

figure(1)
plot(f,data(:,2))
title('Magnitudo (dB) fungsi frekuensi')
xlabel('Frekuensi(MHz)');
ylabel('Magnitudo, dB')

figure(2)
plot(f,data(:,3))
title('Phase fungsi frekuensi')
xlabel('Frekuensi(MHz)');
ylabel('Phase, derajat')

figure(3)
plot(f,mrec)
title('Magnitudo Linier fungsi frekuensi')
xlabel('Frekuensi(MHz)');
ylabel('Magnitudo, watt')

figure(4)
plot(tn+40,mabs);grid on;hold on
title('Magnitudo linier hasil IFFT')
xlabel('Excess Delay (ns)');
ylabel('Magnitude, h(tau)')
axis([-50 100 0 1e-5]);

figure(5)
plot(tn+40,mabn);grid on;hold on
```

```
title('Magnitudo hasil IFFT yang dinormalisasi')
xlabel('Excess Delay (ns)');
ylabel('Magnitudo, h(tau)')
axis([-50 100 0 1]);

figure(6)
plot(window);grid on;hold on
title('Window Hamming')
xlabel('Frekuensi, MHz');
ylabel('Magnitudo')
axis([0 400 0 1]);

figure(7)
plot(tn+40,mabdb);grid on;hold on
title('Magnitudo hasil IFFT yang dinormalisasi')
xlabel('Excess Delay (ns)');
ylabel('Magnitudo, h(tau)')
axis([-50 500 -25 0]);
```

MODUL 11
PROSES REKURSI

I. Tujuan Instruksional Khusus:

- Setelah melakukan praktikum, mahasiswa diharapkan dapat membuat algoritma rekursi yang diimplementasikan untuk menghitung nilai Faktorial dan bilangan Fibonasi

II. Konsep Dasar Rekursi

Rekursif adalah salah satu metode dalam dunia matematika dimana definisi sebuah fungsi mengandung fungsi itu sendiri. Dalam dunia pemrograman, rekursi diimplementasikan dalam sebuah fungsi yang memanggil dirinya sendiri.

Dalam penyelesaian masalah matematika terdapat banyak kasus yang membutuhkan proses perhitungan berulang. Misalnya dalam menghitung nilai *n!* yang terdefinisi sebagai perkalian bilangan asli dari 1 sampai dengan n. Secara matematis hal tersebut ditulis sebagai $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$. Lebih lanjut $n!$ digunakan sebagai dasar untuk menghitung nilai permutasi dan kombinasi dimana dalam penerapannya untuk menentukan banyaknya titik sampel dalam suatu pengamatan data statistik peluang. Dalam kasus tersebut, parameter berhentinya proses perhitungan ditentukan dari banyaknya proses pengulangan, proses pengulangan inilah yang kemudian dikenal sebagai proses rekursi.

Misalnya untuk mendapatkan perhitungan 5! dilakukan proses pengulangan sebanyak 4 kali, yaitu 5 x 4 = 20, 20 x 3 = 60, 60 x 2 = 120 dan 120 x 1 = 120, sehingga 5! =120. Dalam bahasa pemograman komputer, apabila parameter berhentinya proses pengulangan ditentukan dari banyaknya proses pengulangan, maka bahasa program yang dapat digunakan adalah struktur *for*. Berikut dalam kegiatan di bawah ini, diberikan petunjuk algoritma komputasi untuk membangun bahasa pemograman perhitungan nilai faktorial, permutasi, kombinasi serta barisan dan deret Fibonacci dengan input bilangan asli menggunakan struktur *for*.

Fungsi factorial dari bilangan bulat positif n didefinisikan sebagai berikut:

n!= n.(n-1)! , jika n>1

n!= 1 , jika n=0, 1

Contoh :

3!= 3. 2!

3!= 3. 2. 1

3!= 6

Fungsi lain yang dapat diubah ke bentuk rekursif adalah perhitungan Fibonacci. Bilangan Fibonacci dapat didefinisikan sebagai berikut:

$f_n = f_{n-1} + f_{n-2}$ untuk $n > 2$

$f_1 = 1$

$f_2 = 1$

Berikut ini adalah barisan bilangan Fibonacci mulai dari n=1

1 1 2 3 5 8 13 21 34

III. Perangkat Yang Diperlukan

- PC multimedia yang sudah dilengkapi dengan OS Windows
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box DSP

IV. Langkah Percobaan

4.1. Nilai Faktorial

Buatlah program untuk menghitung nilai faktorial sebanyak banyaknya, sehingga tampak adanya algoritma rekursi didalamnya. Algoritma program sebagai berikut:

Step 1: Definisikan nilai *n*

Step 2: Untuk i = (*n-1*), (*n*-2), …, 2, 1

Step 3: Hitung nilai *n* = *n* x *i*

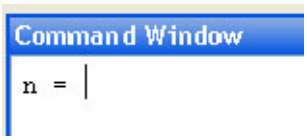
Step 4: Akhiri Step 2

Step 5: Cetak nilai *n*

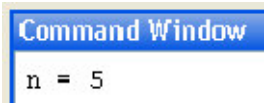
Penulisan program dari langkah tersebut diatas dengan matlab sbb:

```
1 -  n=input('n = ');
2 -  for i=n-1:-1:1
3 -      n=n*i
4 -  end
5 -  disp(['n! = ',num2str(n)])
```

Untuk menjalankan program tersebut, pada menu file, pilih **Debug ➔ Save and Run.** Selanjutnya simpan dengan nama **Kegiatan1_1** pada folder kerja anda. Setelah itu, anda menuju **command window** untuk memberikan inputan dari program yang telah dijalankan sebagai berikut .



Setelah muncul sebagaimana gambar di atas, anda kemudian memasukkan bilangan yang hendak dicari nilai faktorialnnya, misalnya akan dihitung nilai dari 5!, maka inputan n= 5 diberikan dan diperoleh hasil sebagai berikut.



Setelah itu kemudian **enter,** maka akan keluar hasil sebagai berikut.

```
n = 5

n =

    20


n =

    60


n =

   120


n =

   120

n! = 120
```

Mula-mula *n* bernilai 5, kemudian *n* diganti dengan *n* yang baru dengan nilai dari hasil perkalian 5 x 4 = 20, Selanjutnya *n* =20 diganti dengan *n* yang baru yang diperoleh dari hasil kali antara 20 x 3 = 60. Begitu seterusnya hingga nilai terakhir *n* = 120 x 1 = 120. Jika proses perhitungan tidak ingin di tampilkan, maka pada ujung step ke-3 anda akhiri dengan tanda titik koma (;), maka akan diperoleh eksekusi program sebagai berikut.

```
n = 5
n! = 120
```

4.2. Bilangan Fibonaci

a. Buatlah program membentuk barisan Fibonacci hingga suku ke-n, untuk barisan Fibonacci dengan pola 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,….

   Algoritma program sebagai berikut:

   Step 1: Definisikan n sebagai suku ke-n dari barisan fibonacci

   Step 2: Definisikan dua suku pertama, misal sebut sebagi F=[0 1]

   Step 3: Untuk *i* = 3:n

   Step 4: Bentuk barisan Fibonacci, yaitu F(i)=F(i-1)+F(i-2)

   Step 5: Akhiri Step

   Step 6: Cetak barisan Fibonacci

Konversikan algoritma di atas ke dalam bahasa Matlab dan dapatkan contoh output program sebagai berikut:

```
n = 8
F = 0   1   1   2   3   5   8  13
```

b. Buatlah program membentuk barisan Fibonacci hingga suku ke-n, untuk barisan Fibonacci dengan pola 0+1+1+2+3+5+8+13+21+34+…

Algoritma yang digunakan sebagai berikut.

Step 1: Definisikan nilai n

Step 2: Buat barisan Fibonacci, misal sebut sebagai F (Lihat Kegiatan 4)

Step 3: Buat tempat penyimpanan dari hasil penjumlahan tiap 2 suku, misal sebut sebagai JF, dengan JF = 0

Step 4: Untuk i=1:n

Step 5: Hitung nilai dari JF, yakni JF=JF+F(i)

Step 4: Cetak barisan Fibonacci (F) dan deretnya (JF)

Konversilah bahasa program tersebut ke dalam bahasa matlab, dan pastikan contoh output anda adalah sebagai berikut:

```
n = 5
F = 0  1  1  2  3
n = 8
F = 0   1   1   2   3   5   8  13
```

MODUL 11
PROSES REKURSI

I. Tujuan Instruksional Khusus:

- Setelah melakukan praktikum, mahasiswa diharapkan dapat membuat algoritma rekursi yang diimplementasikan untuk menghitung nilai Faktorial dan bilangan Fibonasi

II. Konsep Dasar Rekursi

Rekursif adalah salah satu metode dalam dunia matematika dimana definisi sebuah fungsi mengandung fungsi itu sendiri. Dalam dunia pemrograman, rekursi diimplementasikan dalam sebuah fungsi yang memanggil dirinya sendiri.

Dalam penyelesaian masalah matematika terdapat banyak kasus yang membutuhkan proses perhitungan berulang. Misalnya dalam menghitung nilai *n!* yang terdefinisi sebagai perkalian bilangan asli dari 1 sampai dengan n. Secara matematis hal tersebut ditulis sebagai $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$. Lebih lanjut $n!$ digunakan sebagai dasar untuk menghitung nilai permutasi dan kombinasi dimana dalam penerapannya untuk menentukan banyaknya titik sampel dalam suatu pengamatan data statistik peluang. Dalam kasus tersebut, parameter berhentinya proses perhitungan ditentukan dari banyaknya proses pengulangan, proses pengulangan inilah yang kemudian dikenal sebagai proses rekursi.

Misalnya untuk mendapatkan perhitungan 5! dilakukan proses pengulangan sebanyak 4 kali, yaitu 5 x 4 = 20, 20 x 3 = 60, 60 x 2 = 120 dan 120 x 1 = 120, sehingga 5! =120. Dalam bahasa pemograman komputer, apabila parameter berhentinya proses pengulangan ditentukan dari banyaknya proses pengulangan, maka bahasa program yang dapat digunakan adalah struktur *for*. Berikut dalam kegiatan di bawah ini, diberikan petunjuk algoritma komputasi untuk membangun bahasa pemograman perhitungan nilai faktorial, permutasi, kombinasi serta barisan dan deret Fibonacci dengan input bilangan asli menggunakan struktur *for*.

Fungsi factorial dari bilangan bulat positif n didefinisikan sebagai berikut:

n!= n.(n-1)! , jika n>1

n!= 1 , jika n=0, 1

Contoh :

3!= 3. 2!

3!= 3. 2. 1

3!= 6

Fungsi lain yang dapat diubah ke bentuk rekursif adalah perhitungan Fibonacci. Bilangan Fibonacci dapat didefinisikan sebagai berikut:

$f_n = f_{n-1} + f_{n-2}$ untuk $n > 2$

$f_1 = 1$

$f_2 = 1$

Berikut ini adalah barisan bilangan Fibonacci mulai dari n=1

1 1 2 3 5 8 13 21 34

III. Perangkat Yang Diperlukan

- PC multimedia yang sudah dilengkapi dengan OS Windows
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box DSP

IV. Langkah Percobaan

4.1. Nilai Faktorial

Buatlah program untuk menghitung nilai faktorial sebanyak banyaknya, sehingga tampak adanya algoritma rekursi didalamnya. Algoritma program sebagai berikut:

Step 1: Definisikan nilai *n*

Step 2: Untuk i = (*n-1*), (*n*-2), …, 2, 1

Step 3: Hitung nilai *n* = *n* x *i*

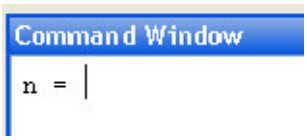
Step 4: Akhiri Step 2

Step 5: Cetak nilai *n*

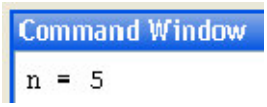
Penulisan program dari langkah tersebut diatas dengan matlab sbb:

```
1 -  n=input('n = ');
2 -  for i=n-1:-1:1
3 -      n=n*i
4 -  end
5 -  disp(['n! = ',num2str(n)])
```

Untuk menjalankan program tersebut, pada menu file, pilih **Debug ➔ Save and Run.** Selanjutnya simpan dengan nama **Kegiatan1_1** pada folder kerja anda. Setelah itu, anda menuju **command window** untuk memberikan inputan dari program yang telah dijalankan sebagai berikut .



Setelah muncul sebagaimana gambar di atas, anda kemudian memasukkan bilangan yang hendak dicari nilai faktorialnnya, misalnya akan dihitung nilai dari 5!, maka inputan n= 5 diberikan dan diperoleh hasil sebagai berikut.



Setelah itu kemudian **enter,** maka akan keluar hasil sebagai berikut.

```
n = 5

n =

    20


n =

    60


n =

   120


n =

   120

n! = 120
```

Mula-mula *n* bernilai 5, kemudian *n* diganti dengan *n* yang baru dengan nilai dari hasil perkalian 5 x 4 = 20, Selanjutnya *n* =20 diganti dengan *n* yang baru yang diperoleh dari hasil kali antara 20 x 3 = 60. Begitu seterusnya hingga nilai terakhir *n* = 120 x 1 = 120. Jika proses perhitungan tidak ingin di tampilkan, maka pada ujung step ke-3 anda akhiri dengan tanda titik koma (;), maka akan diperoleh eksekusi program sebagai berikut.

```
n = 5
n! = 120
```

4.2. Bilangan Fibonaci

a. Buatlah program membentuk barisan Fibonacci hingga suku ke-n, untuk barisan Fibonacci dengan pola 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,….

   Algoritma program sebagai berikut:

   Step 1: Definisikan n sebagai suku ke-n dari barisan fibonacci

   Step 2: Definisikan dua suku pertama, misal sebut sebagi F=[0 1]

   Step 3: Untuk *i* = 3:n

   Step 4: Bentuk barisan Fibonacci, yaitu F(i)=F(i-1)+F(i-2)

   Step 5: Akhiri Step

   Step 6: Cetak barisan Fibonacci

Konversikan algoritma di atas ke dalam bahasa Matlab dan dapatkan contoh output program sebagai berikut:

```
n = 8
F = 0   1   1   2   3   5   8  13
```

b. Buatlah program membentuk barisan Fibonacci hingga suku ke-n, untuk barisan Fibonacci dengan pola 0+1+1+2+3+5+8+13+21+34+…

Algoritma yang digunakan sebagai berikut.

Step 1: Definisikan nilai n

Step 2: Buat barisan Fibonacci, misal sebut sebagai F (Lihat Kegiatan 4)

Step 3: Buat tempat penyimpanan dari hasil penjumlahan tiap 2 suku, misal sebut sebagai JF, dengan JF = 0

Step 4: Untuk i=1:n

Step 5: Hitung nilai dari JF, yakni JF=JF+F(i)

Step 4: Cetak barisan Fibonacci (F) dan deretnya (JF)

Konversilah bahasa program tersebut ke dalam bahasa matlab, dan pastikan contoh output anda adalah sebagai berikut:

```
n = 5
F = 0  1  1  2  3
n = 8
F = 0   1   1   2   3   5   8  13
```

MODUL 12
TRANSFORAMSI-Z

I. Tujuan Instruksional Khusus:

- Setelah melakukan percobaan ini, diharapkan mahasiswa dapat menentukan persamaan tansformasi Z dari persamaan sebuah sinyal diskrit.

II. Teori Transformasi Z

Transfromasi-Z, seperti halnya Transformasi Laplace merupakan suatu metode atau alat matematis yang sangat bermanfaat untuk mendesain, menganalisa dan memonitoring suatu sistem. Transfromasi-Z mirip dengan Transformasi Laplace namun bekerja pada domain diskrit dan merupakan generalisasi dari transformasi Fourier dari fungsi khusus. Pengetahuan tentang Transformasi-Z sangat diperlukan sekali pada saat mempelajari filter digital dan sistem.

Di dalam matematika dan pengolahan sinyal, transforamsi-Z digunakan untuk mengkonversi suatu sinyal waktu diskrit yang terdiri dari sekuen real atau komplek menjadi seuatu representasi di dalam domain frekuensi. Transformasi-z bisa didefinisikan sebagai suatu transformasi satu sisi (*one-sided transform*)atau transformasi dua sisi (*two-sided transform*).

Transformasi Dua sisi

Transformasi dua sisi atau disebut juga sebagai *bilateral* atau *two-sided* Z-transform pada suatu sinyal waktu diskrit $x[n]$didefinisikan sebagai fungsi $X(z)$, dan keduanya memiliki hubungan sebagai berikut

$$X(z) = \mathrm{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \qquad (12\text{-}1)$$

Transformasi Satu Sisi

Ketika suatu kondisi dimana $x[n]$ hanya memilikinilai untuk $n \geq 0$, maka bentuk transformasi-z disebut sebagai transformasi satu sisi atau disebut juga sebagai *single-sided* atau *unilateral* Z-transform. Hubungan antara X(z) dengan x[n] bisa didefinisikan sebagai berikut.

$$X(z) = \mathrm{Z}\{x[n]\} = \sum_{n=0}^{\infty} x[n] z^{-n} \qquad (12\text{-}2)$$

Di dalam pengolahan sinyal, bentuk yang terakhir ini disebut sebagai sebuah kondisi causal. Suatu analisa filter analog bisa didekati dengan menggunakan transformasi laplace, sementara itu untuk filter digital atau filter recurssive bisa dibangun berdasarkan analisa transformasi-z.

Secara umum transformasi-Z dan transformasi Laplace memiliki beberapa kesamaan, termasuk di dalam proses transformasi terhadap sebuah fungsi.Secara umumkeduanya melakukan transformasi

dengan tahapan sebagai berikut: pengamatan respon impulse dengan sinusoida dan eskponensial untuk mendapatkan nilai-nilai pole dan zero sistem tersebut. Transformasi Laplace berkaitan dengan persamaan differential, domian-s dan bidang-s (*s-plane*). Sementara itu transformasi-z berkaitan dengan persamaan beda difference, domain-z, dan bidang–z (z-plane). Tetapi kedua teknik ini bukan merupakan suatu bentuk pencerminan satu dengan yang lain, sebab bidang-s (s-plane) di atur di dalam suatu sistem koordinat rectangular, sedangkan bidang-z (z-plane) menggunakan format sistem koordinat polar.

Filter digital rekurssive seringkali dirancang dengan memanfaatkan filter analog klasik seperti Butterworth, Chebyshev, atau elliptic. Sederetan persmaaan matematik yang merepresentasikan filter kemudian dikonversi untuk mendapatkan filter digital yang akan dirancang, dalamhal ini proses konversi menggunakan tansformasi-Z. Berikut ini adalah contoh proses transformasi-z dari sebuah sekuen $x[n] = x^2$. Langkah mendapatkan bentuk di dalam domain-z secara manual adalah sebagai berikut:

$$\frac{2z}{(z-a)^3} = \sum_{n=0}^{\infty} n(n-1)a^{n-2}z^{-n}$$
$$\frac{2z}{(z-1)^3} = \sum_{n=0}^{\infty} n(n-1)z^{-n}$$
$$n(n-1) = n^2 - n$$
$$n^2 = n(n-1) + n$$
$$Z[n^2] = Z[n(n-1)] + Z[n]$$
$$Z[n^2] = \frac{2z}{(z-1)^3} + \frac{z}{(z-1)^2} = \frac{z^2 + z}{(z-1)^3}$$

Transformasi-Z dengan Matlab

Kita pertimbangkan sebuah persamaan linear dengan koefisien-koefisien konstan yang dituliskan didalam bentuk polinomial seperti berikut yang sering digunakan untuk pemodelan hubungan antara sekuen input x[k] dengan respon output y[k] pada suatu sistem LTID (linear time invariant discrete).

$$a_n y[k] + a_{n-1} y[k-1] + \ldots\ldots\ldots + a_0 y[k-n] = b_m x[k] + b_{m-1} x[k-1] + \ldots\ldots\ldots + b_0 x[k-m]$$

Pada bagian ini kita fokuskan pada fungsi yang merepresentasikan fungsi transfer pada bentuk-z,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_m + b_{m-1}z^{-1} + \cdots\cdots + b_0 z^{-m}}{a_n + a_{n-1}z^{-1} + \cdots\cdots + a_0 z^{-n}}.$$

Kita juga bisa melakukan faktorisasi menjadi bentuk berikut ini

$$H(z) = \frac{Y(z)}{X(z)} = K\frac{(1 - z_0 z^{-1})(1 - z_1 z^{-1}) \cdot \cdots \cdot (1 - z_M z^{-1})}{(1 - p_0 z^{-1})(1 - p_1 z^{-1}) \cdot \cdots \cdot (1 - p_N z^{-1})}.$$

Pada Matlab diosumsikan bahwa bagian numerator (pembilang) dan denominator (penyebut) pada z-transfer function diekpresikan di dalam bentuk kenaikan berpangkat pada z^{-1}. Matlab menyediakan beberapa fungsi (M-files) untuk bisa digunakan di dalam proses transformasi-z. Dalam hal ini ada 5 fungsi tersebut cukup penting, yaitu: *residuez*, *residue*, *tf2zp*, *zp2tf*, dan *zplane*. Untuk mengilustrasikan cara bagaimana fusng-fungsi tersebut bekerja, anda dapat melakukannya dengan mengikuti langkah-langkah percobaan yang sudah diberikan.

III. Perangkat Yang Diperlukan

- PC multimedia yang sudah dilengkapi dengan OS Windows
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box DSP

IV. Langkah Percobaan

4.1. Fungsi Residuez

1. Untuk mengilustrasikan penggunaan sebuah fungsi ***residuez***, kita mulai dengan menghitung ekspansi parsial dari fungsi transfer bentuk z berikut ini.

$$H(z) = \frac{2z(3z + 7)}{(z - 1)(z^2 - 6z + 25)}$$

Ekpresikan fungsi trsnfer-z di dalam bentuk pangkat z^{-1} berikut ini

$$H(z) = \frac{6z^{-1} + 34z^{-2}}{1 - 7z^{-1} + 31z^{-2} - 25z^{-3}}$$

2. Anda buat program dengan Matlab untuk menentukan bentuk ekspansi pecah parsial seperti dibawah ini. Dalam hal ini anda menyusun dengan Matlab editor atau cukup dengan Matlab Command Line.

```
B = [0; 6; 34; 0];      % koef. numerator N(z)
A = [1; -7; 31; -25];   % koef. denominator D(z)
[R,P,K] = residuez(B,A) % Hitung partial fraction expansion
```

Coba anda lihat nilai-nilai R, P dan K dari perintah diatas, apakah nilainya seperti berikut ini

```
R = [-1.0000-1.2500j, -1.0000+1.2500j, 2.0000]
P = [3.0000+4.0000j, 3.0000-4.0000j, 1.0000]
K=[]
```

4.2. Fungsi Rezidue

Pada suatu kondisi diperlukan untuk membentuk sebuah pecah parsial di dalam terminologi polinomial z, bukan dalam bentuk z^{-1}. Dalam beberapa kasus, fungsi Matlab yang digunakan adalah ***residue***. Dalam hal ini kita coba untuk menyelesaikan persamaan dalam bentuk fungsi transfer berikut ini.

$$\frac{H(z)}{z} = \frac{6z^{-1} + 34z^{-2}}{z^3 - 7z^2 + 31z - 25z}$$

1. Anda buat program dengan Matlab untuk menentukan bentuk ekspansi pecah parsial seperti dibawah ini.

```
B = [0; 0; 6; 34];     % koef. numerator N(z)
A = [1; -7; 31; -25];  % koef. Denumerator D(z)
[R,P,K] = residue(B,A) % hitung  partial fraction expansion
```

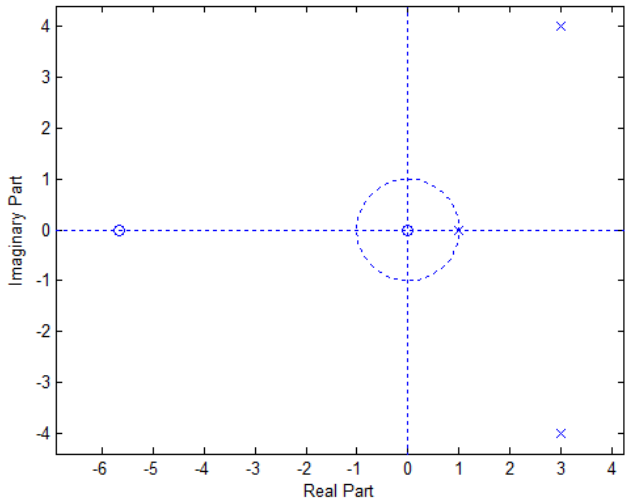
Hasilnya adalah seperti berikut

```
R = [-1.0000-1.2500j, -1.0000+1.2500j, 2.0000]
P = [3.0000+4.0000j, 3.0000-4.0000j, 1.0000]
K = [].
```

4.3. Menghitung Pole danZero dari Fungsi Transfer

Dengan memanfaatkan fungsi transfer yang ada pada bagian 4.1., anda buat sebuah program Matlab untuk menghasilkan nilai pole, zero dan posisinya pada bidang-z.

```
B = [0, 6, 34, 0];    % Koef. numerator N(z)
A = [1, -7, 31, -25]; % Koef. denominator D(z)
[Z,P,K] = tf2zp(B,A)  % Hitung poles dan zeros
zplane(Z,P) % plot poles dan zeros
```



Gambar 9.1. Posisi pole dan zero pada bidang-z

4.4. Mendapatkan nilai Fungsi Transfer dari nilai Pole dan Zero

Pada bagian ini kita berusaha mendapatkan nilai-nilai pembilang dan penyebut untuk dapat menyusun sebuh fungsi tranfer dari kondisi dimana nilai-nilai pole dan zero sudah diketahui. Misalnya pada suatu kasus diketahui bahwa nilai pole dan zero adalah sbb.

- Nilai Zero adalah: 0 dan -0 dan -5.666667
- Nilai Pole adalah: 3+4j , 3- 4 j dan 1

1. Anda dapat membuat program Matlab seperti berikut ini untuk mendapatkan nilai-nilai koefisien fungsi transfernya.

```
Z = [0; -5.666667];          % Zeros di dalam suatu vector kolom
P = [3+4 * j; 3-4 * j; 1];  % Poles di dalam suatu vector kolom
K = 6; % Gain pada numerator
[B,A] = zp2tf(Z,P,K);        % Proses penghitungan
```

Langkah ini akan memberikanhasil seperti berikut

```
B = [0 6 34 0]
A = [1 -7 31 -25]
```

V. Tugas

Sebuah fungsi tranfer digunakan untuk merepresentasikan sebuah sistem.

$$H(z) = \frac{2z^4 + 16z^3 + 44z^2 + 56z + 32}{3z^4 + 3z^3 + 15z^2 + 18z - 12}$$

Dengan melakukan faktorisasi, anda dapatkan nilai-nilai pole, zero, dan gambarkan posisinya pada bidang-z.

MODUL 12
TRANSFORAMSI-Z

I. Tujuan Instruksional Khusus:

- Setelah melakukan percobaan ini, diharapkan mahasiswa dapat menentukan persamaan tansformasi Z dari persamaan sebuah sinyal diskrit.

II. Teori Transformasi Z

Transfromasi-Z, seperti halnya Transformasi Laplace merupakan suatu metode atau alat matematis yang sangat bermanfaat untuk mendesain, menganalisa dan memonitoring suatu sistem. Transfromasi-Z mirip dengan Transformasi Laplace namun bekerja pada domain diskrit dan merupakan generalisasi dari transformasi Fourier dari fungsi khusus. Pengetahuan tentang Transformasi-Z sangat diperlukan sekali pada saat mempelajari filter digital dan sistem.

Di dalam matematika dan pengolahan sinyal, transforamsi-Z digunakan untuk mengkonversi suatu sinyal waktu diskrit yang terdiri dari sekuen real atau komplek menjadi seuatu representasi di dalam domain frekuensi. Transformasi-z bisa didefinisikan sebagai suatu transformasi satu sisi (*one-sided transform*)atau transformasi dua sisi (*two-sided transform*).

Transformasi Dua sisi

Transformasi dua sisi atau disebut juga sebagai *bilateral* atau *two-sided* Z-transform pada suatu sinyal waktu diskrit $x[n]$didefinisikan sebagai fungsi $X(z)$, dan keduanya memiliki hubungan sebagai berikut

$$X(z) = \mathrm{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] z^{-n} \qquad (12\text{-}1)$$

Transformasi Satu Sisi

Ketika suatu kondisi dimana $x[n]$ hanya memilikinilai untuk $n \geq 0$, maka bentuk transformasi-z disebut sebagai transformasi satu sisi atau disebut juga sebagai *single-sided* atau *unilateral* Z-transform. Hubungan antara X(z) dengan x[n] bisa didefinisikan sebagai berikut.

$$X(z) = \mathrm{Z}\{x[n]\} = \sum_{n=0}^{\infty} x[n] z^{-n} \qquad (12\text{-}2)$$

Di dalam pengolahan sinyal, bentuk yang terakhir ini disebut sebagai sebuah kondisi causal. Suatu analisa filter analog bisa didekati dengan menggunakan transformasi laplace, sementara itu untuk filter digital atau filter recurssive bisa dibangun berdasarkan analisa transformasi-z.

Secara umum transformasi-Z dan transformasi Laplace memiliki beberapa kesamaan, termasuk di dalam proses transformasi terhadap sebuah fungsi.Secara umumkeduanya melakukan transformasi

dengan tahapan sebagai berikut: pengamatan respon impulse dengan sinusoida dan eskponensial untuk mendapatkan nilai-nilai pole dan zero sistem tersebut. Transformasi Laplace berkaitan dengan persamaan differential, domian-s dan bidang-s (*s-plane*). Sementara itu transformasi-z berkaitan dengan persamaan beda difference, domain-z, dan bidang–z (z-plane). Tetapi kedua teknik ini bukan merupakan suatu bentuk pencerminan satu dengan yang lain, sebab bidang-s (s-plane) di atur di dalam suatu sistem koordinat rectangular, sedangkan bidang-z (z-plane) menggunakan format sistem koordinat polar.

Filter digital rekurssive seringkali dirancang dengan memanfaatkan filter analog klasik seperti Butterworth, Chebyshev, atau elliptic. Sederetan persmaaan matematik yang merepresentasikan filter kemudian dikonversi untuk mendapatkan filter digital yang akan dirancang, dalamhal ini proses konversi menggunakan tansformasi-Z. Berikut ini adalah contoh proses transformasi-z dari sebuah sekuen $x[n] = x^2$. Langkah mendapatkan bentuk di dalam domain-z secara manual adalah sebagai berikut:

$$\frac{2z}{(z-a)^3} = \sum_{n=0}^{\infty} n(n-1)a^{n-2}z^{-n}$$
$$\frac{2z}{(z-1)^3} = \sum_{n=0}^{\infty} n(n-1)z^{-n}$$
$$n(n-1) = n^2 - n$$
$$n^2 = n(n-1) + n$$
$$Z[n^2] = Z[n(n-1)] + Z[n]$$
$$Z[n^2] = \frac{2z}{(z-1)^3} + \frac{z}{(z-1)^2} = \frac{z^2 + z}{(z-1)^3}$$

Transformasi-Z dengan Matlab

Kita pertimbangkan sebuah persamaan linear dengan koefisien-koefisien konstan yang dituliskan didalam bentuk polinomial seperti berikut yang sering digunakan untuk pemodelan hubungan antara sekuen input x[k] dengan respon output y[k] pada suatu sistem LTID (linear time invariant discrete).

$$a_n y[k] + a_{n-1} y[k-1] + \ldots\ldots\ldots + a_0 y[k-n] = b_m x[k] + b_{m-1} x[k-1] + \ldots\ldots\ldots + b_0 x[k-m]$$

Pada bagian ini kita fokuskan pada fungsi yang merepresentasikan fungsi transfer pada bentuk-z,

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_m + b_{m-1}z^{-1} + \cdots\cdots + b_0 z^{-m}}{a_n + a_{n-1}z^{-1} + \cdots\cdots + a_0 z^{-n}}.$$

Kita juga bisa melakukan faktorisasi menjadi bentuk berikut ini

$$H(z) = \frac{Y(z)}{X(z)} = K\frac{(1 - z_0 z^{-1})(1 - z_1 z^{-1}) \cdot \cdots \cdot (1 - z_M z^{-1})}{(1 - p_0 z^{-1})(1 - p_1 z^{-1}) \cdot \cdots \cdot (1 - p_N z^{-1})}.$$

Pada Matlab diosumsikan bahwa bagian numerator (pembilang) dan denominator (penyebut) pada z-transfer function diekpresikan di dalam bentuk kenaikan berpangkat pada z^{-1}. Matlab menyediakan beberapa fungsi (M-files) untuk bisa digunakan di dalam proses transformasi-z. Dalam hal ini ada 5 fungsi tersebut cukup penting, yaitu: *residuez*, *residue*, *tf2zp*, *zp2tf*, dan *zplane*. Untuk mengilustrasikan cara bagaimana fusng-fungsi tersebut bekerja, anda dapat melakukannya dengan mengikuti langkah-langkah percobaan yang sudah diberikan.

III. Perangkat Yang Diperlukan

- PC multimedia yang sudah dilengkapi dengan OS Windows
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box DSP

IV. Langkah Percobaan

4.1. Fungsi Residuez

1. Untuk mengilustrasikan penggunaan sebuah fungsi ***residuez***, kita mulai dengan menghitung ekspansi parsial dari fungsi transfer bentuk z berikut ini.

$$H(z) = \frac{2z(3z + 7)}{(z - 1)(z^2 - 6z + 25)}$$

Ekpresikan fungsi trsnfer-z di dalam bentuk pangkat z^{-1} berikut ini

$$H(z) = \frac{6z^{-1} + 34z^{-2}}{1 - 7z^{-1} + 31z^{-2} - 25z^{-3}}$$

2. Anda buat program dengan Matlab untuk menentukan bentuk ekspansi pecah parsial seperti dibawah ini. Dalam hal ini anda menyusun dengan Matlab editor atau cukup dengan Matlab Command Line.

```
B = [0; 6; 34; 0];      % koef. numerator N(z)
A = [1; -7; 31; -25];   % koef. denominator D(z)
[R,P,K] = residuez(B,A) % Hitung partial fraction expansion
```

Coba anda lihat nilai-nilai R, P dan K dari perintah diatas, apakah nilainya seperti berikut ini

```
R = [-1.0000-1.2500j, -1.0000+1.2500j, 2.0000]
P = [3.0000+4.0000j, 3.0000-4.0000j, 1.0000]
K=[]
```

4.2. Fungsi Rezidue

Pada suatu kondisi diperlukan untuk membentuk sebuah pecah parsial di dalam terminologi polinomial z, bukan dalam bentuk z^{-1}. Dalam beberapa kasus, fungsi Matlab yang digunakan adalah ***residue***. Dalam hal ini kita coba untuk menyelesaikan persamaan dalam bentuk fungsi transfer berikut ini.

$$\frac{H(z)}{z} = \frac{6z^{-1} + 34z^{-2}}{z^3 - 7z^2 + 31z - 25z}$$

1. Anda buat program dengan Matlab untuk menentukan bentuk ekspansi pecah parsial seperti dibawah ini.

```
B = [0; 0; 6; 34];     % koef. numerator N(z)
A = [1; -7; 31; -25];  % koef. Denumerator D(z)
[R,P,K] = residue(B,A) % hitung  partial fraction expansion
```

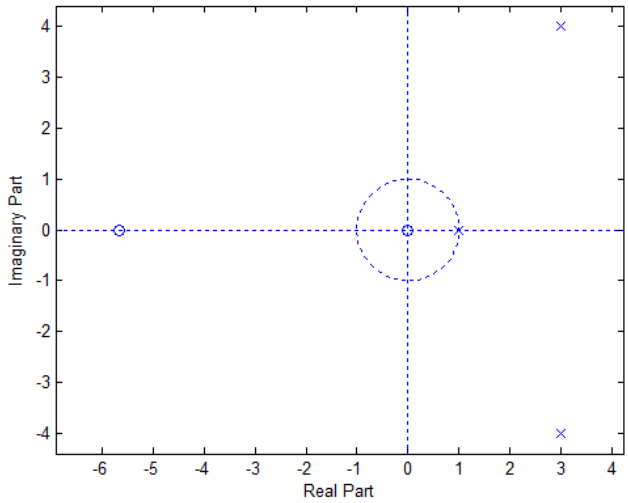
Hasilnya adalah seperti berikut

```
R = [-1.0000-1.2500j, -1.0000+1.2500j, 2.0000]
P = [3.0000+4.0000j, 3.0000-4.0000j, 1.0000]
K = [].
```

4.3. Menghitung Pole danZero dari Fungsi Transfer

Dengan memanfaatkan fungsi transfer yang ada pada bagian 4.1., anda buat sebuah program Matlab untuk menghasilkan nilai pole, zero dan posisinya pada bidang-z.

```
B = [0, 6, 34, 0];    % Koef. numerator N(z)
A = [1, -7, 31, -25]; % Koef. denominator D(z)
[Z,P,K] = tf2zp(B,A)  % Hitung poles dan zeros
zplane(Z,P) % plot poles dan zeros
```



Gambar 9.1. Posisi pole dan zero pada bidang-z

4.4. Mendapatkan nilai Fungsi Transfer dari nilai Pole dan Zero

Pada bagian ini kita berusaha mendapatkan nilai-nilai pembilang dan penyebut untuk dapat menyusun sebuh fungsi tranfer dari kondisi dimana nilai-nilai pole dan zero sudah diketahui. Misalnya pada suatu kasus diketahui bahwa nilai pole dan zero adalah sbb.

- Nilai Zero adalah: 0 dan -0 dan -5.666667
- Nilai Pole adalah: 3+4j , 3- 4 j dan 1

1. Anda dapat membuat program Matlab seperti berikut ini untuk mendapatkan nilai-nilai koefisien fungsi transfernya.

```
Z = [0; -5.666667];          % Zeros di dalam suatu vector kolom
P = [3+4 * j; 3-4 * j; 1];  % Poles di dalam suatu vector kolom
K = 6; % Gain pada numerator
[B,A] = zp2tf(Z,P,K);        % Proses penghitungan
```

Langkah ini akan memberikanhasil seperti berikut

```
B = [0 6 34 0]
A = [1 -7 31 -25]
```

V. Tugas

Sebuah fungsi tranfer digunakan untuk merepresentasikan sebuah sistem.

$$H(z) = \frac{2z^4 + 16z^3 + 44z^2 + 56z + 32}{3z^4 + 3z^3 + 15z^2 + 18z - 12}$$

Dengan melakukan faktorisasi, anda dapatkan nilai-nilai pole, zero, dan gambarkan posisinya pada bidang-z.

MODUL 13
TRANSFORAMSI LAPLACE

I. Tujuan Instruksional Khusus:

- Siswa mampu membangun sebuah program transofrmasi Laplace dengan menggunakan Matlab
- Penyelesaian masalah parsial komplek, memahami konstelasi pole dan zero, dan s-plane

II. Teori Transformasi Laplace

Transformasi Laplace pada suatu sinyal (fungsi) $f(t)$ bisa dituliskan sebagai $F=L(f)$, atau didefinisikan dengan persamaan berikut ini

$$F(s) = \int_0^\infty f(t)e^{-st}dt \qquad (13\text{-}1)$$

DimanaF adalah sebuah fungsi bernilai pada bilangan komplek. Variabel s disebut sebagai variabel frekuensi komlek, dengan satuan /sec, dan t adalah variabel waktu di dalam satuan detik. Dalam hal ini st menjadi tanpa satuan. Pada kondisi awal diasumsikan bahwa f tidak memiliki nilai impulse pada saat $t = 0$.

Contoh sederhana pada kasus berikut ini. Disini kita akan melakukan transformasi Laplace untuk fungsi $f(t) = e^t$. Dengan memanfatkan persamaan diatas,

$$\begin{aligned} F(s) &= \int_0^\infty f(t)e^{-st}dt \\ &= \int_0^\infty e^t e^{-st}dt \\ &= \int_0^\infty e^{(1-s)t}dt \\ &= \frac{1}{1-s}e^{(1-s)t}\Big|_0^\infty \\ &= \frac{1}{s-1} \end{aligned}$$

Sehingga kita bisa mendapatkan penyederhanaan di dalam transformasi Laplace sebuah hubungan seperti berikut

$$L\left(e^t\right) = \frac{1}{s-1} \qquad (13\text{-}1)$$

III. Perangkat Yang Diperlukan

- PC multimedia yang sudah dilengkapi dengan OS Windows
- Perangkat Lunak Matlab yang dilengkapi dengan Tool Box DSP

IV. Langkah Percobaan

4.1. Tranformasi Laplace untuk persamaan sederhana

1. Dapatkan tranformasi laplace dari persamaan berikut $f(t) = t^4$

   Anda dapat memperoleh bentuk Laplace dengan memanfaatkan kode Matlab berikut.

```
syms t;
f = t^4;
laplace(f)
```

   Hasilnya adalah

```
ans =
    24/s^5
```

2. Dapatkan tranformasi laplace dari persamaan berikut $f(t) = e^{-at}$

   Anda dapat memperoleh bentuk Laplace dengan memanfaatkan kode Matlab berikut.

```
syms t a x;
f = exp(-a*t);
laplace(f,x)
```

   Outputnya adalah

```
ans =
1/(a + x)
```

4.2. Tranformasi Laplace Invers untuk persamaan sederhana

1. Dapatkan invers Laplace dari persamaan berikut $f(s) = \frac{1}{s^2}$. Anda dapat memperoleh bentuk invers Laplace dengan memanfaatkan kode Matlab berikut

```
syms s;
f=1/s^2;
ilaplace(f)
```

   Outputnya adalah

```
ans = t
```

2. Dapatkan invers Laplace dari persamaan berikut $g(t) = \frac{1}{(t-a)^2}$. Anda dapat memperoleh bentuk invers Laplace dengan memanfaatkan kode Matlab berikut

```
syms a t;
g=1/(t-a)^2;
ilaplace(g)
```

   Outputnya adalah

```
ans =
```

x*exp(a*x)

3. Dapatkan invers Laplace dari persamaan berikut $f(u)=\frac{1}{u^2-a^2}$ Anda dapat memperoleh bentuk invers Laplace dengan memanfaatkan kode Matlab berikut

```
syms x u;
syms a real;
f=1/(u^2-a^2);
simplify(ilaplace(f,x))
```

Outputnya adalah

```
ans =
sinh(a*x)/a
```

4.3. Mendapatkan Pole dan Zero pada bidang-s

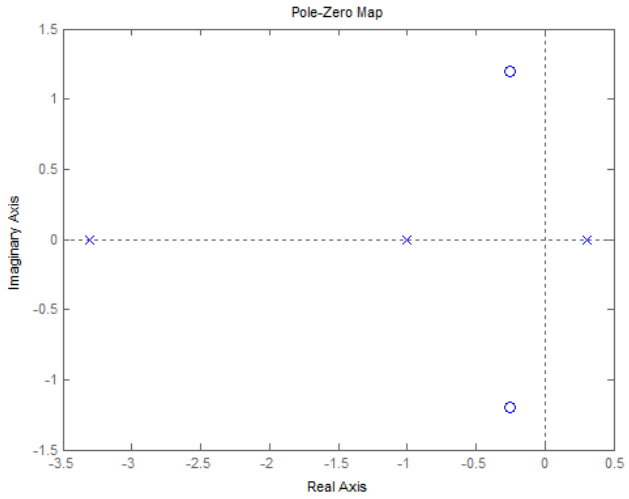
Pada bagian kita akan mendapatkan posisi pole dan zero pada bidang-s untuk sebuah sistem LTI yang memiliki fungsi transfer dalam polinomial-s seperti berikut:

$$G(s)=\frac{2s^2+s+3}{s^3+4s^2+2s-1}$$

1. Buat sebuah program Matlab sederhana berikut ini

```
clc;
sys = tf([0 2 1 3],[1 4 2 -1])
pzmap(sys)
```

Outputnya akan memberikan tampilan seperti gambar dibawah ini.



Gambar 13.1. Hasil pemetaan pole dan zero sebuah fungsi transfer domain-s

Pada kasus berikutnya kita akan mencoba menyelesaikan persoalan yang lebih komplek. Dapatkan hasil komputasi dengan tarnsformasi Laplace untuk sebuah fungsi $x(t) = e^{-t}u(t)$, dan sinyal termodulasi $y(t) = e^{-t}\cos(10t)u(t)$. Gambarkan bentuk sinyal, transformasi laplacenya, dan posisi pole dan zero pada bidang-s

2. Untuk itu anda dapat memanfaatkan kode Matlab berikut ini.

```
syms t
x = exp (-t);
y = x * cos(10 * t);
X = laplace(x)
Y = laplace(y)
% plotting of signals and poles/zeros
figure;
subplot(221)
ezplot(x,[0,5]);grid
axis([0 5 0 1.1]);title('x(t) = exp(-t)u(t)')
numx = [0 1];denx = [1 1];
subplot(222)
sys=tf(numx,denx);
pzmap(sys)
subplot(223)
ezplot(y,[-1,5]);grid
axis([0 5 -1.1 1.1]);title('y(t)= cos(10t)exp(-t)u(t)')
numy = [0 1 1];deny = [1 2 101];
sys2=tf(numy,deny)
subplot(224)
pzmap(sys2)
```

Transformasi Laplace memberikan nilai dalam domain-s seperti berikut

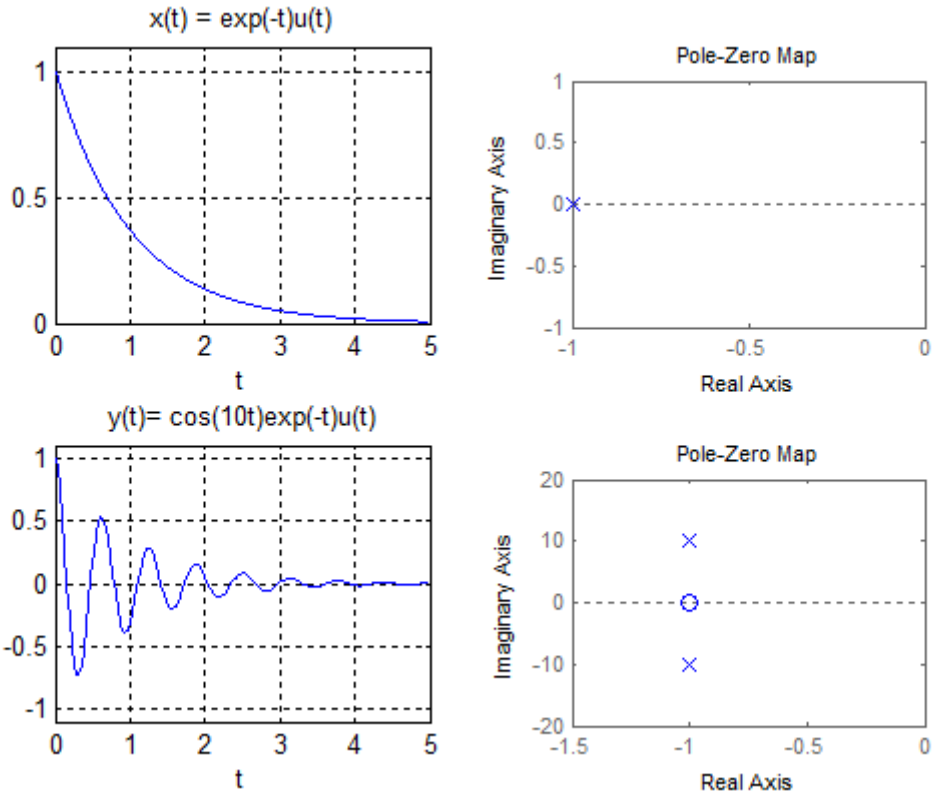
```
X =

 1/(s + 1)

Y =

(s + 1)/((s + 1)^2 + 100)
```

Sedangkan gambaran sinyal x(t), y(t) dan posisi pole dan zero bisa dilihat pada Gambar berikut ini.



Gambar 13.2. Gambaran sinyal dan posisi pole-zero yang direpresentasikan.

DAFTAR PUSTAKA:

1. Gabel and Roberts, "*Signal and Linier System*", 3rd ed. John Willey, 1987
2. Oppenheim, "*Signal and System*", Prentice Hall, 1983
3. Kwakernaak, H. dan Sivan, "*Modern Signal and System*", Prentice Hall Inc. 1991
4. Lathi, B.P, "*Signal Processing and Linear System*", 1991
5. Naresh K. Sinha, "*Limear Systems*", 1991
6. Simon Haykin & Barry Van Veen, "*Signals and Systems*", John Willey and Sons, 2003.
7. Luis F. Chaparro, "*Signals and Systems using Matlab*", Elsevier Academic Press, 2011.
8. Michael Corinthios,"*Signals, Systems, Transforms, and Digital Signal Processing with MATLAB*", Taylor and Francis Group, LLC, 2009.
9. Tadeusz A. Wysocki, Bahram Honary, and Beata J. Wysocki, "*Signal Processing for Telecommunications and Multimedia*", Springer, London, England, 2005.
10. K.R. Rao, D.N. Kim l, J.J. Hwang, "*Fast Fourier Transform: Algorithms and Applications*", Springer, New York, USA, 2010.